EPFL

Thèse n° 10 045

Interpretable Representation Learning and Evaluation for Abstractive Summarization

Présentée le 12 juin 2023

Faculté des sciences et techniques de l'ingénieur
Laboratoire de l'IDIAP
Programme doctoral en génie électrique

pour l'obtention du grade de Docteur ès Sciences

par

Andreas Thomas MARFURT

Acceptée sur proposition du jury

Prof. A. M. Alahi, président du jury
Prof. D. Gatica-Perez, Dr J. Henderson, directeurs de thèse
Prof. M. Lapata, rapporteuse
Dr S. Narayan, rapporteur
Prof. A. Popescu-Belis, rapporteur

École
polytechnique
fédérale
de Lausanne

2023

I'm a scientist; because I invent, transform, create, and destroy for a living,
and when I don't like something about the world, I change it.
— Rick Sanchez

Nobody exists on purpose.
Nobody belongs anywhere.
We're all going to die.
Come watch TV.
— Morty Smith

Acknowledgements

First, I would like to express my gratitude to my supervisor James Henderson. Without him taking a chance on me and providing continued support throughout the thesis I would have never made it. I am impressed by his never-ending curiosity and enjoyed discussing conceptual ideas in our weekly meeting which sometimes went on for way longer than the hour for which it was scheduled. I am grateful to have gotten along so well with my supervisor, thanks to his sincerity and good sense of humor. Thank you, Jamie!

Then, I would like to thank the people who made this thesis possible. Thanks to Daniel Gatica-Perez, who graciously spent his time dealing with administrative matters, showed genuine interest, and provided encouragement. Thanks to our collaborators from the Geneva Graduate Institute, especially David Sylvan and Ashley Thornton, who have met with me many times during my doctorate and who have always stayed curious, open-minded, and have been much more lenient with me than I deserved.

I am thankful for the great people in our NLU group at Idiap, that made the work at Idiap such a pleasant experience. Thank you, Florian, Rabeeh, Nikos, Navid, Lesly, Alireza, Melika, Andrei, Fabio, Molly, and Lonneke, for great conversations at our group lunches, interesting reading groups, honest discussions of ideas with a focus on identifying the good parts, and an unforgettable first offline conference after too many online ones. I would like to thank many more people at Idiap for the past 4.5 years, but especially those I will remember for their great companionship. Julian, Angelos, Pablo, Suhan, Pavel, Christian, Teja, Angel, Suraj, I hope I will hear from you, or otherwise please get famous, so that I can at least read of you in the news.

I would not have the same view of science today without the time at the Data Analytics Lab at ETH Zurich. Florian, Yannic, Carsten, Hadi, Kevin, Paulina, Jason, Jonas, Celestine, Aurelien, Gary, Octavian, and Thomas, you have shaped my scientific understanding. Thanks for two intense years and a pair of memorable conference experiences.

Finally, but most importantly, I am grateful to my family and friends for their unrelenting support. I hope I will be able to give it back and pass it on in the same way that all of you did!

Lausanne, May 23, 2023 Andreas Marfurt

Abstract

Abstractive summarization has seen big improvements in recent years, mostly due to advances in neural language modeling, language model pretraining, and scaling models and datasets. While large language models generate summaries that are fluent, coherent, and integrate the salient information from the source document well, there are still a few challenges. Most importantly, information that is either not supported by the source document (hallucinations) or factually inaccurate finds its way into the machine-written summaries. Moreover, and connected to this first point, knowledge retrieval and summary generation happen implicitly, which leads to a lack of interpretability and controllability of the models.

In this thesis, we contribute to solving these problems by working on making the summarization process more interpretable, faithful, and controllable. The thesis consists of two parts. In Part I, we learn interpretable representations that help with summary structure, faithfulness, and document understanding. First, we plan summary content at the sentence level, building a next sentence representation from the summary generated so far. Second, we integrate an entailment interpretation into standard text-encoding neural network architectures. In the last chapter of the first part, we use multiple object discovery methods from computer vision to identify semantic text units that should facilitate the extraction of salient information from source documents.

In Part II, we turn to the evaluation of summarization models, and also contribute annotated resources for our tasks. We start by using the attentions and probability estimates during summary generation to identify hallucinations. We then apply summarization models in a novel semi-structured setting, where the model is asked to generate an interpretation from a long source document. For this novel task, we develop an evaluation technique that allows efficient contrastive evaluation of generative models with respect to user-specified distinctions.

Keywords: abstractive summarization, text summarization, representation learning, interpretability, hallucination detection, text generation, evaluation, datasets, natural language understanding, natural language processing.

Zusammenfassung

Das Generieren von abstrahierten Zusammenfassungen hat in den letzten Jahren grosse Fortschritte gemacht, insbesondere durch verbesserte neuronale Sprachmodellierung, das Vortrainieren von Sprachmodellen, und das Skalieren von Modellen und Datensätzen. Grosse Sprachmodelle schreiben flüssig lesbare Zusammenfassungen, die kohärent sind und die wichtigen Informationen eines Ursprungsdokuments beinhalten. Aber es gibt weiterhin Herausforderungen. Immer wieder finden Informationen, die entweder nicht vom Ursprungsdokument abgeleitet werden können (Halluzinationen), oder gar falsch sind, ihren Weg in die Zusammenfassung. Eng verbunden mit diesem Punkt ist die fehlende Interpretationsmöglichkeit und Kontrollierbarkeit von Modellen. Sie entstehen, weil die Modelle sowohl das Abrufen von gelerntem Wissen als auch das Schreiben der Zusammenfassung implizit vornehmen.
In dieser Doktorarbeit tragen wir zur Lösung dieser Probleme bei, indem wir die Erstellung von Zusammenfassungen interpretierbarer, inhaltsgetreuer und kontrollierbarer machen. Die Arbeit besteht aus zwei Teilen. Wir beginnen den ersten Teil mit dem Erstellen eines Plans für die Zusammenfassung auf der Satzebene, indem wir eine Repräsentation für den nächsten Satz generieren, die auf der bisher erstellten Zusammenfassung beruht. Dann interpretieren wir die Repräsentationen von gängigen neuronalen Netzwerken als Vektoren, die Entailment-Beziehungen abbilden, und testen die daraus resultierenden Konsequenzen für die Anpassung der Netzwerkarchitekturen. Im letzten Kapitel des ersten Teils nutzen wir verschieden Methoden zur Beschreibung von Objekten, die im Feld der automatischen Bildbearbeitung entstanden sind. Mit ihnen wollen wir semantische Textbausteine identifizieren, welche das Extrahieren von wichtigen Informationen erleichtern sollen.
Im zweiten Teil wenden wir uns der Evaluierung von existierenden Modellen zu, und veröffentlichen zu diesem Zweck mehrere annotierte Datensätze. Im ersten Kapitel des zweiten Teils verwenden wir die Gewichte des Aufmerksamkeits-Mechanismus sowie die Wahrscheinlichkeitsverteilung über das nächste Wort zum Erkennen von möglichen Halluzinationen. Danach wenden wir Zusammenfassungs-Modelle auf ein neues Szenario an, in dem sie eine Interpretation eines langen Ursprungsdokuments erstellen sollen. Für diese neue Aufgabe entwickeln wir eine Evaluierung, die es uns ermöglicht, ein Modell auf effiziente Weise auf die Fähigkeit zur Unterscheidung von Benutzer-spezifizierten Kategorien zu testen.

Schlüsselwörter: Abstrahierte Zusammenfassung, Text-Zusammenfassung, Lernen von Repräsentationen, Interpretierbarkeit, Entdeckung von Halluzinationen, Text-Generierung, Evaluation, Datensätze, Verständnis natürlicher Sprache, Verarbeiten natürlicher Sprache.

Contents

Acknowledgements **i**

Abstract (English) **iii**

Abstract (Deutsch) **v**

1 Introduction **1**
1.1 Historical Context . . . 1
1.1.1 The Beginning of Deep Learning in NLP . . . 1
1.1.2 Advances in NLP During the Thesis . . . 2
1.2 Motivation . . . 3
1.3 Contributions . . . 4
1.3.1 Interpretable Representation Learning . . . 5
1.3.2 Evaluation . . . 6
1.4 Structure . . . 7

2 Background on Summarization **9**
2.1 Deep Learning in NLP . . . 9
2.1.1 Recurrent Neural Networks . . . 9
2.1.2 Sequence-to-Sequence Models . . . 11
2.1.3 Attention . . . 11
2.1.4 Transformer . . . 12
2.2 Extractive Summarization . . . 14
2.2.1 Inverted Pyramid Method . . . 14
2.2.2 Lead Baseline . . . 14
2.2.3 Influential Models . . . 15
2.3 Abstractive Summarization . . . 16
2.3.1 Task Definition . . . 16
2.3.2 Influential Models . . . 17
2.4 Evaluation . . . 19
2.4.1 Automatic Metrics . . . 19
2.4.2 Human Evaluation . . . 22
2.5 Datasets . . . 25
2.5.1 Single-Document Summarization . . . 25

2.5.2 Multi-Document Summarization . . . 28
2.5.3 Long-Document Summarization . . . 29
2.6 State of the Art . . . 31
2.6.1 Finetuned Models . . . 31
2.6.2 Zero- and Few-Shot Large Language Models . . . 34

I Interpretable Representation Learning 37

3 Summary Content Planning 39
3.1 Introduction . . . 40
3.2 Base Model . . . 41
3.3 Hierarchical Transformer Decoder . . . 42
3.3.1 Sentence Generator . . . 42
3.3.2 Word Generator . . . 44
3.4 Alternative Approaches . . . 44
3.5 Experimental Setup . . . 46
3.5.1 Datasets . . . 46
3.5.2 Metrics . . . 46
3.5.3 Implementation Details . . . 47
3.6 Results . . . 48
3.6.1 Results on Curation Corpus . . . 48
3.6.2 Attribution to Sentence Representation . . . 49
3.6.3 Model Ablation . . . 50
3.6.4 Number of Parameters . . . 50
3.6.5 SUM-QE Evaluation . . . 51
3.6.6 Results on CNN/DailyMail . . . 52
3.6.7 Human Evaluation . . . 53
3.6.8 Example Summaries . . . 54
3.7 Related Work . . . 54
3.8 Conclusion . . . 58
3.8.1 Higher Abstractiveness . . . 58

4 Entailment Representations 59
4.1 Entailment in Summarization . . . 60
4.2 Entailment Vector Framework . . . 60
4.3 Entailment-based Architectures . . . 61
4.3.1 Entailment LSTM . . . 61
4.3.2 Entailment GRU . . . 64
4.3.3 Entailment Transformer . . . 64
4.4 Experiments . . . 67
4.4.1 Datasets . . . 67
4.4.2 Training Details . . . 68

4.4.3 Natural Language Inference . . . 69
4.4.4 Language Modeling . . . 70
4.5 Results . . . 70
4.5.1 Natural Language Inference . . . 70
4.5.2 Language Modeling . . . 71
4.6 Architecture Ablation . . . 72
4.6.1 Number of Model Parameters . . . 72
4.6.2 LSTM Architecture . . . 75
4.6.3 Transformer Architecture . . . 75
4.7 Related Work . . . 76
4.8 Conclusion . . . 78

5 Salient Information Extraction and Representation 79
5.1 Semantic Text Units . . . 80
5.2 Object Detection and Representation in Computer Vision . . . 81
5.2.1 Capsule Networks . . . 81
5.2.2 GLOM . . . 83
5.2.3 Slot Attention . . . 84
5.3 Slot Attention for Text . . . 84
5.3.1 Slot Initialization . . . 85
5.3.2 Softmax Normalization . . . 87
5.3.3 Slots as an Information Bottleneck . . . 87
5.3.4 Guiding Attention Patterns . . . 88
5.3.5 Experiments . . . 88
5.4 Pretraining Slot Attention on Extractive Summarization . . . 96
5.4.1 Modeling Adaptations . . . 96
5.4.2 Experiments . . . 98
5.5 Object Representations from Bottom-Up Attention . . . 103
5.5.1 Generating Multi-Level Object Representations . . . 104
5.5.2 Minimizing Mutual Information with Adversarial Training . . . 107
5.5.3 Experiments . . . 109
5.6 Related Work . . . 112
5.7 Conclusion . . . 118
5.7.1 Representation Learning for Encoder-Decoder Models . . . 118

II Evaluation 119

6 Hallucination Detection 121
6.1 What are Hallucinations? . . . 122
6.2 Hallucination Detection on XSum . . . 123
6.3 Hallucination Detection Methods . . . 123
6.3.1 QG-QA Models . . . 123

6.3.2 Dependency-arc Entailment . . . 123
6.3.3 Token-level Prediction with an External Model . . . 124
6.4 Unsupervised Hallucination Detection . . . 124
6.4.1 Motivation . . . 124
6.4.2 Initial Alignment . . . 125
6.4.3 Context Voting . . . 125
6.4.4 Classifying Aligned Tokens . . . 125
6.4.5 Classifying Unaligned Tokens . . . 126
6.4.6 Converting Scores to Probabilities . . . 128
6.5 Experiments . . . 128
6.5.1 Datasets . . . 128
6.5.2 Model Details . . . 131
6.5.3 Baselines . . . 131
6.6 Results . . . 133
6.6.1 Precision-Recall Results . . . 133
6.6.2 Extrinsic Hallucinations . . . 134
6.6.3 Intrinsic Hallucinations . . . 135
6.6.4 Ablation Study . . . 135
6.6.5 Maximum Possible Hallucination Recall . . . 135
6.6.6 Maximum Recall of (Un)aligned Tokens . . . 136
6.6.7 ROC Results . . . 137
6.6.8 Hallucination Examples . . . 137
6.7 Related Work . . . 139
6.8 Conclusion . . . 140
6.8.1 Hallucination Definition . . . 140
6.8.2 Transfer to Other Models . . . 140
6.8.3 Transfer to other datasets. . . . 141
6.8.4 Prevalence of sports topics in hallucinations. . . . 141

7 Semi-Structured Annotations for Interpretation 143
7.1 Learning to Interpret . . . 144
7.2 Semi-Structured Annotations . . . 145
7.2.1 Standardizing Annotations . . . 145
7.2.2 Converting Annotations to Text . . . 145
7.2.3 Sequence-to-sequence Task . . . 145
7.2.4 The FOMC Dataset . . . 146
7.3 Evaluation Modes . . . 150
7.3.1 Full Prediction . . . 150
7.3.2 Completion of a Marked Span . . . 150
7.4 Equivalence Classes Evaluation . . . 150
7.4.1 Definition . . . 150
7.4.2 Creating Evaluation Instances . . . 151

7.4.3 Model Evaluation . . . 152
7.4.4 In-Depth Analysis . . . 153
7.5 Experiments . . . 153
7.5.1 Equivalence Classes . . . 153
7.5.2 Standard Text Generation Metrics . . . 154
7.5.3 Filtering Source Documents . . . 155
7.5.4 Generative Models . . . 155
7.6 Results . . . 156
7.6.1 Equivalence Classes Evaluation . . . 157
7.6.2 Text Generation Metrics . . . 157
7.6.3 In-Depth Analysis . . . 158
7.6.4 Ablation Study . . . 160
7.6.5 Hallucinations . . . 161
7.7 Related Work . . . 162
7.8 Conclusion . . . 163
7.8.1 Value of Annotations for Language Models . . . 163
7.8.2 Subjectivity of the Annotation Process . . . 163
7.8.3 Application to Other Domains . . . 164
7.8.4 Equivalence Classes Creation . . . 164
7.8.5 Syntactic Structure of Equivalence Class Members . . . 164

8 Conclusion **165**
8.1 Summary of Contributions . . . 165
8.2 Inspirations for Future Work . . . 166
8.3 Limitations . . . 168
8.4 Ethical Considerations . . . 169

Bibliography **171**

Curriculum Vitae **197**

1 Introduction

In this introduction, we motivate this thesis from multiple points of view. First, we take a historical perspective, quickly surveying the developments in the field of NLP. Then, we view it from a technical angle by connecting the individual steps a summarization model has to perform with our chapters. Next, we highlight our contributions to the area of summarization and the broader field of natural language generation that we make in this thesis. We finish by giving an overview of the chapters that this thesis comprises.

1.1 Historical Context

Abstractive summarization in its basic form is the task of generating a summary given a source document. Performance on the task has increased dramatically in recent years and now is at a point where summaries are typically of high quality. Nevertheless, some error modes remain, especially concerning hallucinated content, i.e. pieces of information in the summary that are not supported by the source document.

1.1.1 The Beginning of Deep Learning in NLP

In natural language processing (NLP), neural networks are the most widely used class of models that are employed to produce summaries. One specific architecture has come to dominate not just the task of summarization. It is called the *Transformer* (Vaswani et al., 2017), and processes the input units, typically *byte-pair encoding* (BPE) (Sennrich et al., 2016) tokens, across many layers, refining the representation of each input token in the context of the remaining tokens. The increasing number of processing layers has given this branch of machine learning its name, *deep learning*. Achieving its breakthrough in computer vision in 2013 (Krizhevsky et al., 2012), it also picked up steam in NLP soon after. The first popular architectures were based on *recurrent neural networks* (RNN) (Rumelhart et al., 1986). The introduction of attention in 2015 brought the field a step forward (Bahdanau et al., 2015). In 2017, the Transformer architecture was introduced and has since taken the top spot from

RNNs as the most used neural network architecture to process language.

1.1.2 Advances in NLP During the Thesis

The period of the thesis was an exciting time to work in NLP. When I started my thesis in November 2018, the Transformer had been around for a year and it became clear that it was a powerful model that challenged the state of the art in all areas of NLP. Just when the thesis started, BERT (Devlin et al., 2019) appeared and revolutionized NLP by beating specialized state-of-the-art models in a variety of natural language understanding tasks, as measured on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018a). After the successes of pretraining a Transformer encoder for text classification, pretraining the decoder for text generation was a natural next step. Two variants emerged. Decoder-only models, as in the GPT series (Radford et al., 2018, 2019; Brown et al., 2020), just employ a Transformer decoder, while the full model is used by jointly pretrained encoder-decoder models (BART, Lewis et al., 2020; PEGASUS, Zhang et al., 2020a; T5, Raffel et al., 2020). Datasets grew steadily, but more than anything else, the model sizes exploded. While the largest version of T5 consists of 11B parameters, GPT-3 has 175B and the Megatron-Turing NLG (Smith et al., 2022) and PaLM models (Chowdhery et al., 2022) have 530B and 540B, respectively. These large language models (LLM) opened up an interesting operation mode. For some tasks, no further training was necessary, and the knowledge learned during pretraining could be queried either directly (*zero-shot*) or by providing a few examples as context first (*few-shot*). Providing examples as context preceding the actual query has also been called *in-context learning*, due to LLMs seemingly adapting to the task and format of the context. Naturally, different forms of querying the model (also called *prompting*) have emerged, and *prompt engineering* tries to find the best possible way to query LLMs. For reasoning tasks, the currently best working query mode is called *chain-of-thought prompting* (Wei et al., 2022), where the few-shot examples come together with a textual description of the reasoning steps to arrive at the solution. LLMs can also be finetuned to understand different prompt forms, such as instructions (Ouyang et al., 2022). Finetuning encoder-decoder models still provides the best results on language understanding tasks with plenty of supervised data (Tay et al., 2022), but the capabilities of decoder-only LLMs become more and more impressive (see Section 2.6). It will be exciting to see where NLP research will venture next.

Having said that, this period was also filled with uncertainties and new challenges for researchers. At which point should one abandon the proven approaches and adopt the new paradigms? Will reviewers still consider publications that did not include the Transformer, pretraining, or large language models? In a time where the state of the art was dominated by scaling dataset, compute, and model size, how can one compete without the necessary resources? And if one decides not to chase leaderboard scores, would results from smaller models stay relevant when they cannot be tested on the large models that were kept a company's secret, for justified economic reasons?

At the same time, scaling the models made it harder to understand the contributions of the individual parts of a neural network's architecture. Different studies reached their best results with different components (e.g. activation functions), and while these still had an impact on the final performance, other factors such as the depth of the network, the amount of data (Brown et al., 2020), or the number of training steps (Hoffmann et al., 2022) had a bigger impact. Consequently, the field as a whole moved away from tweaking architectures; almost all current studies use the Transformer architecture, a general method that effectively leverages computation (Sutton, 2019).

This thesis is a reflection of the developments in the field. Neural network architecture and representation learning was arguably the most studied subject in NLP up to the introduction of pretrained encoder-decoder (Lewis et al., 2020; Raffel et al., 2020) or decoder-only models (Radford et al., 2018, 2019). Naturally, the thesis concerns itself with representation learning in Part I. Chronologically, we started the thesis with our work on bringing an entailment interpretation to neural architectures (see Chapter 4). At the time, it was not yet clear that the Transformer would be the last major architectural innovation for an extended period, which now has stretched up until the publication of this thesis. We then turned to devise a special-purpose representation for the decoder (Chapter 3), and afterward to general-purpose encoder representations (Chapter 5) that could benefit a randomly initialized decoder as well as the interpretability of summary generation. Towards the end of that project, it became obvious that pretraining the decoder jointly with the encoder provided large performance benefits. Changing the internal structure of a pretrained model would only disturb the coordination between the encoder and decoder that was so carefully built up during pretraining. This was especially true as models and pretraining datasets became larger and larger. As a result, large parts of the research community shifted to devising tasks and evaluating models, and so did we in 2022. In Part II, we analyze the internal representations of BART and utilize them to detect hallucinations (see Chapter 6). We also employ BART for a novel task and devise an evaluation that allows evaluating language models in precisely defined scenarios. Personally, I see a lot of opportunities in this development. As long as researchers and companies continue to openly discuss and share their trained models (Scao et al., 2022; Zhang et al., 2022a; Touvron et al., 2023) instead of keeping them under wraps (OpenAI, 2023), more analysis and evaluations will give us more insight into the strengths and limitations of our language understanding models.

1.2 Motivation

The task of abstractive summarization can be divided into multiple steps. We propose the following six subtasks that a summarization model should perform (potentially implicitly) to generate an abstractive summary:

1. Understanding the source document
2. Identifying the salient information
3. Organizing it, recognizing dependencies (e.g. concerning time or entities)

4. Selecting which information to include in the summary (potentially depending on the desired aspect or output length)
5. Planning the summary, for example at the sentence level
6. Writing the summary words in a fluent and grammatical style, with the desired length

In a classic encoder-decoder sequence-to-sequence model, the first three subtasks are typically the responsibility of the encoder, and the latter three of the decoder (for extractive summarization, step 4 is the final step, and performed by the classifier).

This thesis addresses all of these subtasks in individual chapters. The only exception is document understanding (step 1), which is implicitly handled at the pretraining stage of the models.

- Step 2, identifying salient information, is investigated in Chapter 5, where we try to unsupervisedly cluster the words of the source document into phrases, and then use an information bottleneck to only keep salient information.
- Step 3, relating the salient information, is done in Chapter 4, where we learn representations with an entailment interpretation that let us identify the dependencies between words in the source document.
- Step 4, the selection of which information to include, is viewed from the perspective of saliency in Chapter 5. In Chapter 6, the model is evaluated for hallucinations in the information it copied from different parts of the source document.
- Step 5, summary content planning, is the main focus of Chapter 3. Our hierarchical decoder predicts a latent sentence representation for the next summary sentence, on which it conditions the word-by-word generation.
- Step 6, writing the summary, is evaluated in Chapter 6, where uncertainty in the generation process is used to identify likely hallucinations. In Chapter 7, we predict the aspects of an interpretation in text form.

With the thesis touching on all of the subtasks of abstractive summarization, we hope to present a diverse view of the task. Chapter 7 further formulates the task of interpretation as a step beyond summarization. The models are trained to mimic the human process of abstraction combined with speculating on the actors' reasons and motives in the historical context. We now state our contributions.

1.3 Contributions

This thesis is structured into two parts on interpretable representation learning and evaluation. In the first part, our goal is to improve the performance and interpretability of summarization models by introducing inductive biases that let us learn representations for specific subtasks of summarization. In the second part, we evaluate existing sequence-to-sequence models on their faithfulness and generation capabilities in well-defined contexts. We put a special focus

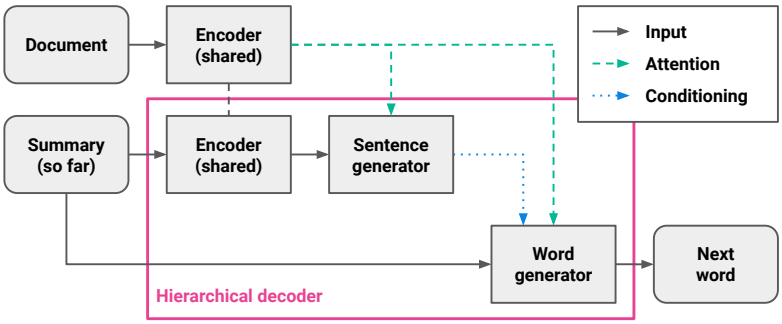


Figure 1.1: We introduce a hierarchical decoder in the sentence planner, where a sentence generator predicts an outline for the next summary sentence.

on the efficiency of our evaluations, both in construction and execution. To support further research, we open-source our code and models. Additionally, we release new evaluation datasets and an annotated corpus on the interpretation of monetary policy documents.

1.3.1 Interpretable Representation Learning

In Chapter 3, we start by introducing the *sentence planner* model that uses hierarchy on the decoder side to predict a continuous sentence representation for the next summary sentence and then conditions the generation on it (Marfurt and Henderson, 2021). Our model is shown in Figure 1.1. Integrating hierarchy in the Transformer decoder's generation process is nontrivial, and we present an effective conditioning method that outperforms using attention to the higher-level sentence plan. Our evaluation on the Curation Corpus and CNN/DailyMail is extensive and includes an attribution analysis, an ablation, and a comparison to the baseline with an increased number of parameters. Our model consistently produces more abstractive summaries while retaining high ROUGE scores, two objectives that are in opposition.

In Chapter 4, we adapt the two most common neural network architectures for learning contextual word embeddings, RNN and Transformer, to generate representations with an entailment interpretation, according to the entailment vector framework of Henderson and Popa (2016). We devise a number of architecture changes as inductive biases and evaluate the entailment representations on natural language inference and language modeling.

The last chapter in Part I, Chapter 5, introduces an ambitious plan for unsupervised clustering of words into semantic units, phrases that should be of use to general language understanding. To achieve this, we adapt object discovery and representation algorithms from computer vision to text. We also devise our own *objecter* model with adversarial training that minimizes the mutual information between object representations. We analyze the attention patterns between different model components to get an understanding of how (and if) the model uses these semantic text units. Our experiments are comprehensive, evaluating both abstractive and extractive summarization on the Curation Corpus and CNN/DailyMail.

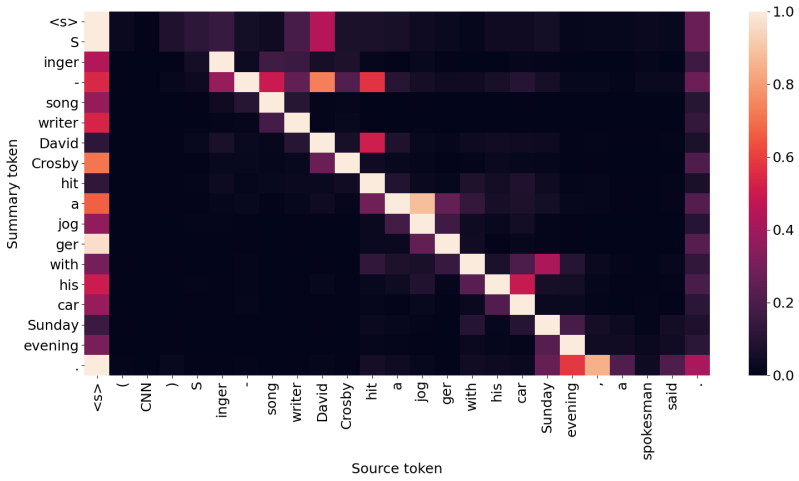


Figure 1.2: BART cross-attentions help us identify copied source segments, which we use to find hallucinations.

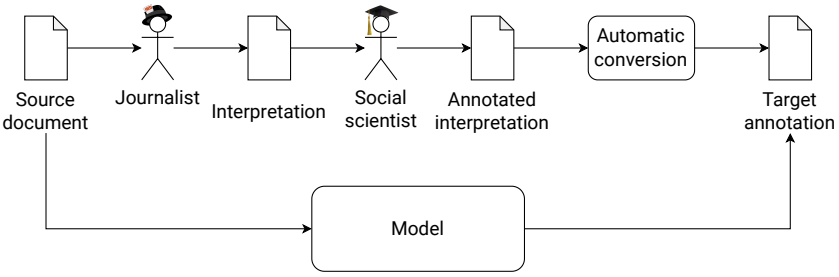


Figure 1.3: In our interpretation task, the model imitates a journalist's interpretation of a source document. The relevant details of the interpretation are carefully extracted and annotated by social scientists.

1.3.2 Evaluation

We start Part II in Chapter 6 with an analysis of BART's behavior when generating summaries for CNN/DailyMail (Marfurt and Henderson, 2022). We find that we can use the model-internal by-products of summary generation (attentions and decoding probabilities) to detect hallucinations (see Figure 1.2). The detection is very efficient to run since it does not require training or running an external model. In contrast to prior work, we classify every summary word and extend the study of hallucinations on the XSum dataset to CNN/DailyMail. We release two evaluation datasets with word-level annotations to facilitate future research. One is adapted from a factuality dataset and the other is a product of our human annotation.

In the final chapter of Part II, Chapter 7, we apply sequence-to-sequence models in the novel setting of interpreting documents (Marfurt et al., 2022). We present a new task, where a model learns to imitate the interpretation process of a journalist (see Figure 1.3). We release a carefully annotated corpus on interpreting the monetary policy of the US central bank.

y_{prefix}	`[REFERENCE START] Last week [REFERENCE END] , the [ACTOR START] Federal Reserve [ACTOR END] [ACT START]`
$a^{(\text{pos})}$	`left interest rates unchanged (Did not raise rates) [ACT END]`
$a^{(\text{neg})}$	`decided to raise interest rates (Did raise rates) [ACT END]`

Figure 1.4: Equivalence classes evaluation example of an act annotation. The model has to distinguish the true continuation $a^{(\text{pos})}$ to the prefix y_{prefix} from a distracting continuation $a^{(\text{neg})}$.

The semi-structured annotations enable a fine-grained evaluation of model generations for the individual annotation categories. This evaluation technique, called *equivalence classes evaluation*, is efficient to construct from a clustering of annotated values by domain experts (see an example in Figure 1.4). Our evaluation shows that the structure of our annotations is a natural fit for language models, and allows the models to pick up on important details even with small training data. This chapter is a product of our interdisciplinary collaboration with political and economic scientists at the Geneva Graduate Institute and an interesting showcase of how NLP models could be leveraged for social sciences.

1.4 Structure

After this introductory chapter, we give background information on the task of abstractive summarization, including an overview of models and methods, evaluation, datasets, and a review of the state of the art. The main thesis is divided into two parts. In Part I, we experiment with several ways in which learned representations can improve interpretability and support summarization models in the various subtasks that they need to solve. In Chapter 3, we devise a hierarchical decoder that generates a representation for the next summary sentence, based on the previously generated summary. In Chapter 4, we train and evaluate neural architectures that output representations with an entailment interpretation. In the final chapter of the first part, Chapter 5, we use object discovery methods to find semantic text units to improve the interpretability of summary generation. In Part II, we use by-products of summary generation to detect hallucinations in Chapter 6. In Chapter 7, we apply summarization models on a novel task of interpreting policy announcements of the US central bank, and devise an efficient and fine-grained evaluation of their capabilities for narrowly defined scenarios. Finally, in Chapter 8, we conclude by summarizing our contributions, suggesting directions for future work, and reflecting on the limitations and ethical considerations of this thesis.

2 Background on Summarization

Text summarization is the task of presenting the important information from one or more documents in less space. It can be tackled from different angles, e.g. according to Radev et al. (2002):

> *Extraction* is the process of identifying important material in the text, *abstraction* the process of reformulating it in novel terms, *fusion* the process of combining extracted portions, and *compression* the process of squeezing out unimportant material. The need to maintain some degree of grammaticality and coherence plays a role in all four processes.

Extractive and abstractive summarization are the two main approaches, which we will briefly survey in the following.

First, however, we start with an overview of deep neural network architectures for natural language processing.

2.1 Deep Learning in NLP

Quickly after the successes of deep neural networks in computer vision (Krizhevsky et al., 2012), their popularity in NLP increased as well. In contrast to computer vision, convolutional neural networks (LeCun et al., 1998) were not the dominant architecture. Instead, recurrent neural networks (Rumelhart et al., 1986) with their inductive bias for processing sequences proved to be a good fit for processing language.

2.1.1 Recurrent Neural Networks

Recurrent neural networks process one element of a sequence at each time step t. They keep a hidden state vector h_t with the sequence information up to t. The hidden state is updated with a combination of information from the current input x_t and the previous hidden

state h_{t-1}. The hidden state after the final element of the input sequence is used as the representation for the entire sequence. A common definition of the network architecture is the Elman RNN (Elman, 1990):

$$h_t = \sigma(W x_t + U h_{t-1} + b) \tag{2.1}$$

with an activation function σ (e.g. the sigmoid function), and learnable weights W, U and bias b.

A lot of variations of RNN architectures exist. In the following, we present the two most common, LSTM and GRU.

Long Short-Term Memory (LSTM). The LSTM cell (Hochreiter and Schmidhuber, 1997) is defined by the following equations at a time step t:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2.2}$$
$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2.3}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{2.4}$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{2.5}$$
$$h_t = o_t \circ \tanh(c_t) \tag{2.6}$$

where f_t is the forget gate, i_t is the input gate, and o_t is the output gate. σ is the sigmoid function, $\circ$ is the element-wise product, W and U are weight matrices on the input x_t and the previous hidden state h_t, respectively, and b are biases. Finally, c_t is the cell state that serves as the memory for the LSTM, whereas h_t is the hidden state that is output at time step t. Various modifications exist, such as coupling the forget and the input gate, or peephole connections (Gers and Schmidhuber, 2000).

Gated Recurrent Unit (GRU). The Gated Recurrent Unit (Cho et al., 2014b) is an attempt to simplify the LSTM architecture while keeping its modeling capacity. In a comparison of character-level languague models, the GRU has even been found to make use of longer contexts than the LSTM (Madsen, 2019). The GRU is defined by:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{2.7}$$
$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{2.8}$$
$$\tilde{h}_t = \tanh(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h) \tag{2.9}$$
$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \tilde{h}_t \tag{2.10}$$

where z_t is the update gate and r_t the reset gate. In comparison to the LSTM, the cell state and the output gate have been removed, and the forget and input gate have been coupled in the update gate z_t. The reset gate r_t allows select which information of the previous hidden state

h_{t-1} to use in the computation of the update $\tilde{h}_t$.

2.1.2 Sequence-to-Sequence Models

For text generation tasks (e.g. machine translation, summarization) the goal is to produce a sequence output for a given input sequence. Sutskever et al. (2014) propose to solve the task by using two separate networks. One network encodes the input sequence x into a latent intermediate representation z and therefore is called the *encoder*. Then, a second network, the *decoder*, generates the output sequence $\hat{y}$ from the intermediate representation z.

$$z = \text{Encoder}(x) \tag{2.11}$$

$$\hat{y} = \text{Decoder}(z) \tag{2.12}$$

Typically, and throughout this thesis, the summary is generated autoregressively by predicting the next token $\hat{y}_i$ based on the latent representation z and the previous tokens $y_{<i} = y_1, \ldots, y_{i-1}$.

$$\hat{y}_i = \text{Decoder}(z, y_{<i}) \tag{2.13}$$

This setup is generally applicable to all tasks that can be formulated as sequence-to-sequence transformations. The encoder as well as the decoder can be any neural network. In the original paper, an LSTM architecture is used for both.

2.1.3 Attention

Since the intermediate representation z is the only information that flows from encoder to decoder; it is an information bottleneck (Tishby et al., 1999). In text generation tasks, different output elements require information from different parts of the input. When the capacity of the representation z is limited, some of that information may be lost. Instead of a single representation containing all of the information, Bahdanau et al. (2015) propose to compute a context vector that is specific to each generated output. The context vector c_i for decoder position i is computed as the attention-weighted sum of the encoder outputs (hidden states in RNNs) $h_j^{(\text{enc})}$, with input length n. The attention weights α_{ij} between decoder position i and encoder position j are probabilities computed from the outputs e_{ij} of the attention function f_{att}.

$$c_i = \sum_{j=1}^{n} \alpha_{ij} h_j^{(\text{enc})} \tag{2.14}$$

$$\alpha_{ij} = \text{softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})} \tag{2.15}$$

$$e_{ij} = f_{\text{att}}\left(h_{i-1}^{(\text{dec})}, h_j^{(\text{enc})}\right) \tag{2.16}$$

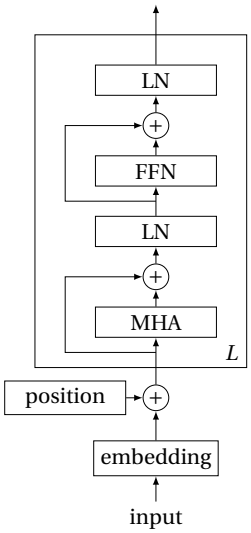


Figure 2.1: Transformer encoder architecture. MHA is multi-head attention, FFN is a feed-forward network, and LN is layer normalization.

For the attention function, additive attention (Equation 2.17) was proposed in the original paper. The simpler multiplicative attention (Equation 2.18) was also found to give good results (Luong et al., 2015).

$$f_{\text{att}}\left(h_{i-1}^{(\text{dec})}, h_j^{(\text{enc})}\right) = v^\top \tanh\left(W h_{i-1}^{(\text{dec})} + U h_j^{(\text{enc})}\right) \tag{2.17}$$

$$f_{\text{att}}\left(h_{i-1}^{(\text{dec})}, h_j^{(\text{enc})}\right) = h_{i-1}^{(\text{dec})} W h_j^{(\text{enc})} \tag{2.18}$$

Key-value attention. Separation of concerns is introduced in Daniluk et al. (2017), which uses the idea to split hidden state dimensions into computation (keys) and memory (values) from memory networks (Weston et al., 2015). The keys are used to compute the attention weights α_{ij}, which then get multiplied by the values to compute the context vector.

2.1.4 Transformer

The Transformer (Vaswani et al., 2017) is a neural network architecture based on attention. Among other innovations, it develops key-value attention into query-key-value attention, where queries and keys are used to determine the attention weights, and values contain the data for further processing. The Transformer no longer computes a context vector to update a hidden state. Instead, the result of the attention computation is the layer's output.

The Transformer architecture does not possess recurrent parts, making it easier to parallelize. It was designed for sequence-to-sequence tasks and therefore comprises both an encoder and a decoder. Figure 2.1 shows the Transformer encoder's architecture. The boxed area represents a single Transformer layer, which is stacked L times. The input sequence is embedded as for

the previous architectures, but since the Transformer does not process the input in sequence, it needs to know about the input tokens' positions. The position information is therefore added to the input embeddings, before being passed to the first layer. Each encoder layer consists of two sublayers, multi-head attention (MHA) and a feed-forward network (FFN), which are both equipped with a skip connection and subsequent layer normalization (LN). We formalize one layer of the Transformer encoder as follows:

$$a_t^l = \mathrm{LN}(h_t^{l-1} + \mathrm{MHA}(h_t^{l-1})) \tag{2.19}$$

$$h_t^l = \mathrm{LN}(a_t^l + \mathrm{FFN}(a_t^l)) \tag{2.20}$$

where t is the current time step, l is the layer, a_t^l is the output of the attention sublayer, and h_t^l the output of the feed-forward network. We set $h_t^0 = x_t$ for the input to the first attention sublayer. We further specify the attention sublayer as:

$$q, k, v = x_t W^q, x_t W^k, x_t W^v \tag{2.21}$$

$$q_i, k_i, v_i = f_{\text{divide}}(q), f_{\text{divide}}(k), f_{\text{divide}}(v), \quad \text{with } 1 \le i \le H \tag{2.22}$$

$$\mathrm{MHA}_i(q_i, k_i, v_i) = \mathrm{softmax}\left(\frac{q_i k_i}{\sqrt{d_k}}\right) v_i \tag{2.23}$$

$$\mathrm{MHA}(x) = f_{\text{merge}}(\mathrm{MHA}_i(q_i, k_i, v_i))\ W + b, \quad \text{with } 1 \le i \le H \tag{2.24}$$

where q, k, v are the queries, keys, and values, respectively. $W^{\{q,k,v\}}$ are their projection matrices. f_{divide} is a function that splits q, k and v along the model dimension d to be processed by H individual attention heads MHA_i. $d_k = d/H$ is the resulting dimensionality of the key (equivalently for q, v). After the per-head attention weights have been computed with dot-product attention and multiplied with the values, the merge function f_{merge} concatenates the results back together, such that $\mathrm{MHA}(x) \in \mathbb{R}^d$. A final linear layer projects the outputs with weights W and bias b.

The feed-forward network is a simple 2-layer MLP with a ReLU nonlinearity in between:

$$\mathrm{FFN}(x) = \mathrm{ReLU}(xW_1 + b_1)W_2 + b_2 \tag{2.25}$$

where we have weight matrices $W_{\{1,2\}}$ and biases $b_{\{1,2\}}$. The input and output dimensions are d, but the inner dimension is $4d$. For language model pretraining, the GELU activation function (Hendrycks and Gimpel, 2016) is sometimes used instead of a ReLU (Radford et al., 2018).

In the Transformer decoder, an additional attention sublayer is inserted. The first sublayer of multi-head attention (same as Equation 2.19) is also called *self-attention*, as it only performs attention on its input sequence. The second attention sublayer, called *cross-attention*, differs from self-attention by replacing Equation 2.22. The queries are still constructed from the previous sublayer's outputs a_t^l, but the keys and values come from the last encoder layer

outputs h^L_{enc}:

$$q, k, v = a^l_t W^q, h^L_{\text{enc}} W^k, h^L_{\text{enc}} W^v \tag{2.26}$$

The output of cross-attention becomes the input to the feed-forward sublayer (Equation 2.20). A layer of the Transformer decoder looks as follows:

$$a^l_t = \text{LN}(h^{l-1}_t + \text{SelfAttention}(h^{l-1}_t)) \tag{2.27}$$
$$b^l_t = \text{LN}(a^l_t + \text{CrossAttention}(a^l_t, h^L_{\text{enc}})) \tag{2.28}$$
$$h^l_t = \text{LN}(b^l_t + \text{FFN}(b^l_t)) \tag{2.29}$$

2.2 Extractive Summarization

Before automatic text generation saw big improvements with the introduction of deep neural networks, extractive summarization was the go-to method for creating acceptable summaries. In extractive summarization, the task is to select the most important text from the source document. With a few exceptions, extractive approaches operate at the sentence level.

2.2.1 Inverted Pyramid Method

As we will see in Section 2.5, news summarization is probably the most studied genre of text summarization. The two main reasons are its usefulness and, maybe even more importantly, the availability of data. News summarization has its peculiarities, however. For once, certain topics are over-represented compared to other genres, for example, war, crime, and sports. Moreover, the articles themselves are written according to the inverted pyramid method. It is a writing style that puts the most important information at the top of an article, then proceeds to add content of steadily decreasing importance. The origins of the inverted pyramid method are found in nineteenth-century American journalism (Pöttker, 2003). The writing style has important implications for news summarization methods.

2.2.2 Lead Baseline

For a long time, the method of selecting the first n sentences in the news article as the extractive summary, called the *lead* strategy, was a hard-to-beat baseline, typically choosing $n = 3$. This can be attributed to the inverted pyramid writing style, which defines that the most important information should come at the beginning of an article. For genres other than news summarization, this does not necessarily hold.

2.2.3 Influential Models

In the following, we briefly present some influential extractive summarization models. Early approaches were based on position in the document, as well as keywords and key phrases (Radev et al., 2002).

Sentence scoring. A successful early approach represents the sentences in a source document as a graph. LexRank (Erkan and Radev, 2004) computes the graph's adjacency matrix as the cosine similarity of sentences' tf-idf scores. They then apply PageRank (Page et al., 1999) to extract the sentences with the most influence in the graph.

To improve the interpretability of its selection, the model can award each sentence a score per feature, and then include the highest-scoring ones in its extractive summary. SummaRuNNer (Nallapati et al., 2017) computes scores for saliency (as the alignment of a sentence with the document), novelty as negative redundancy (the alignment of a sentence with already selected summary sentences), content, and position.

Hierarchical encoders. Another commonly used technique is to use hierarchical encoders to compute sentence representations from the representations of their words. SummaRuNNer (Nallapati et al., 2017) and NeuSum (Zhou et al., 2018) both use a bidirectional word-level RNN whose outputs are pooled and then input to a bidirectional RNN at the sentence level.

A different hierarchical encoder extracts sentence embeddings from their words with a CNN, followed by an RNN that creates a document embedding (Cheng and Lapata, 2016; Narayan et al., 2018b). A sentence extractor scores the sentences with an attention-based RNN that is conditioned on the document representation and the previously labeled sentences. A sentence's score can predict a rule-based matching to reference summary sentences (Cheng and Lapata, 2016), or how often it participates in the top-scoring extractive candidate summaries, according to their ROUGE scores with the reference (Narayan et al., 2018b).

Tree induction. Liu et al. (2019b) induce a dependency discourse tree for a document, which captures the dependencies between sentences. The root nodes of each dependency tree constitute the summary. Structured attention is employed on the output of a hierarchical Transformer encoder (word and sentence level), and that structure is iteratively refined.

Text matching. Extractive summarization has also been cast as a semantic text matching problem (Zhong et al., 2020). This approach first generates candidate summaries by filtering the non-salient sentences from the source document and then constructing all possible combinations for a given maximum number of sentences. Saliency is computed by an external model, BERTSUMEXT (Liu and Lapata, 2019b). It then computes representations for the source document and candidate summaries from a pretrained BERT model (Devlin et al., 2019) and

selects the highest-scoring candidate according to cosine similarity.

2.3 Abstractive Summarization

With better neural text generation models, the quality of abstractive summaries started to surpass the extractive ones.

2.3.1 Task Definition

Abstractive single-document summarization is a sequence-to-sequence task, where given an input sequence x one has to predict the target sequence y. The default training setting is maximum likelihood estimation (MLE). In MLE, the model predicts the most likely next summary token $\hat{y}_t$ from the vocabulary $\mathcal{V}$ given the source document x and the previous tokens $y_{<t} = y_1, \ldots, y_{t-1}$:

$$\hat{y}_t = \underset{v \in \mathcal{V}}{\operatorname{argmax}}\, p(v|x, y_{<t}) \tag{2.30}$$

During training with *teacher forcing* (Williams and Zipser, 1989), the previous tokens are taken from the reference summary. This leads to a discrepancy between the training and inference modes since in inference the model predicts continuations from its generations. This phenomenon is known as *exposure bias* (Ranzato et al., 2016).

The target sequence is supposed to capture the important information of the source document in a concise text. What is deemed important can be subjective. We now present an information-theoretic view that tries to formalize importance.

Information-theoretic model of importance. In Peyrard (2019a), importance is defined as a combination of redundancy, relevance, and informativeness. It assumes that summaries are built of semantic units $\omega \in \Omega$, which are atomic pieces of information.

The redundancy of a summary S is defined as the maximally possible entropy minus the summary's entropy:

$$\text{Red}(S) = H_{\max} - H(S) \tag{2.31}$$

With a vocabulary Ω, the maximum entropy $H_{\max} = \log|\Omega|$ is independent of the summary S. We can ignore it when we minimize the summary's redundancy and write:

$$\text{Red}(S) \propto -H(S) \tag{2.32}$$

$$= \sum_{\omega_i} P_S(\omega_i) \log P_S(\omega_i) \tag{2.33}$$

for the summary's probability distribution P_S over the semantic units ω_i.

The relevance of a summary S to its source document D is defined by the amount of information they have in common. This is naturally expressed by the cross-entropy:

$$\begin{aligned} \text{Rel}(S, D) &= -H(S, D) && (2.34)\\ &= \sum_{\omega_i} P_S(\omega_i) \log P_D(\omega_i) && (2.35) \end{aligned}$$

Relevance and redundancy combined have an interpretation as the KL divergence D_{KL}:

$$\begin{aligned} D_{\text{KL}}(S, D) &= H(S, D) - H(S) && (2.36)\\ &\propto -\text{Rel}(S, D) + \text{Red}(S) && (2.37) \end{aligned}$$

Thus, by minimizing the KL divergence, we maximize relevance while minimizing redundancy.

For a summary to be informative, we want it to provide us with new information, and therefore overlap as little as possible with our prior knowledge K. This is again expressed as the cross-entropy, this time of the summary with the prior knowledge:

$$\begin{aligned} \text{Inf}(S, K) &= H(S, K) && (2.38)\\ &= -\sum_{\omega_i} P_S(\omega_i) \log P_K(\omega_i) && (2.39) \end{aligned}$$

To achieve high informativeness, we want the distributions P_S and P_K to be different, and therefore the cross-entropy between them to be high. Whereas redundancy and relevance are commonly used in the evaluation of summaries, prior knowledge is rarely included.

2.3.2 Influential Models

The vast majority of abstractive summarization models are attention-based sequence-to-sequence models. Early approaches like Rush et al. (2015) and Nallapati et al. (2016) use an encoder RNN to process the source text. The decoder then performs attention (Bahdanau et al., 2015) over the encoder hidden states for every generation step.

Content selection. The pointer mechanism (Vinyals et al., 2015) is used to copy words from the source document. This is especially helpful for rare words or domain-specific technical terms. In the pointer-generator network (See et al., 2017), the model computes a probability for predicting the next output by the generator or copying from the source document. With a coverage loss, the network avoids repetition by penalizing attention to the same locations in the source document as in previous decoding steps.

Hierarchical attention. Hierarchical encoders are also commonplace in abstractive summarization. Nallapati et al. (2016) use hierarchical attention in the encoder with a word- and a sentence-level RNN. The attention weights at the word level are re-weighted by the

sentence-level attention weights. Celikyilmaz et al. (2018) divide the document into paragraphs, which are encoded separately by agents. Each agent performs attention within its paragraph, and the decoder attends to the agents. Gehrmann et al. (2018) first employ a content selector at the word level to decide which words are candidates for copying. They then use a pointer-generator network with just the admissible tokens to generate the summary.

Reinforcement learning. Reinforcement learning is a popular method to introduce flexible auxiliary rewards, sometimes discrete or coming from external metrics, and still retain differentiability for training with gradient descent. The MLE loss can be combined with a reward for high ROUGE scores to encourage generations that are similar to the reference summary and reduce exposure bias (Paulus et al., 2018). Auxiliary rewards have also been defined for the inclusion of salient words and phrases, or entailment of the summary by the source document, both determined by external classifiers (Pasunuru and Bansal, 2018). In another study, an extractive filtering network is combined with an abstractive rewriter. The sentences to extract as well as the rewritten abstractive sentences are compared to the references with ROUGE-L, which serves as the reward (Chen and Bansal, 2018). A different technique trains a reward prediction model from human preferences, i.e. a ranking of candidate solutions (Christiano et al., 2017). The learning model then performs actions and gets rewards directly from the reward model, without further human interaction. This strategy has been used for book summarization. The model learns to summarize paragraphs, then recursively combines the summaries of adjacent texts until a summary for the entire book remains (Wu et al., 2021).

Encoder pretraining. The introduction of the Transformer paired with self-supervised pretraining on large amounts of text has had a profound impact on the field of natural language processing. Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) is a Transformer encoder pretrained on the task of reconstructing masked input tokens. Liu and Lapata (2019b) initialize their encoder with a pretrained BERT model and the decoder with a randomly initialized Transformer decoder. They first finetune the encoder on extractive summarization before finetuning the full model on abstractive summarization.

Joint encoder-decoder pretraining. Jointly pretrained encoder-decoder models perform especially well on the summarization task, as the model benefits from the language understanding capabilities of the encoder and the language generation of the decoder. BART (Lewis et al., 2020) is pretrained on the task of reconstructing masked input text spans and bringing shuffled sentences back into their original order. A different pretraining objective is used by PEGASUS (Zhang et al., 2020a), which reconstructs entire masked sentences. The text-to-text transfer Transformer (T5) (Raffel et al., 2020) compares different pretraining objectives. It investigates the type of objective (language modeling, reconstructing masked tokens, unshuffling), corruption strategies (masking, replacement, deletion), the rate of corruption as well as the length of the corrupted span. UL2 (Tay et al., 2022) pretrains on a mixture of

denoising objectives and activates the relevant mode for a specific downstream task with a special control token. All of these approaches generate high-quality summaries.

T5 and UL2 also explore scaling up the model parameters. The largest T5 model has 11B parameters, which arguably started a trend of models growing bigger and bigger. As a result of scaling, new model capabilities have started to emerge in language models exclusively from pretraining (Brown et al., 2020).

Zero- and few-shot summarization. Large language models (LLM) are a class of models that achieve high task performance solely based on their pretraining, without further finetuning on the specific task under evaluation. In zero-shot evaluation, the model is directly presented with the example to solve, in the few-shot setting it receives one or more examples of input and desired output pairs. The phrasing of the input, called the *prompt*, can have a big impact on the performance of LLMs (Wei et al., 2022; Suzgun et al., 2022). For abstractive summarization, instruction tuning (Ouyang et al., 2022) was found to significantly increase summary quality compared to standard self-supervised pretraining (Zhang et al., 2023).

In Zhang et al. (2023), it is shown that the suboptimal quality of reference summaries negatively impacts (a) models finetuned on these references, (b) LLMs prompted with them in the few-shot setting, and (c) automatic reference-based evaluations. We look at evaluations next.

2.4 Evaluation

Properly evaluating summaries is a notoriously difficult problem, for multiple reasons. First and foremost, no single ground truth summary exists that we could compare candidates to. If we pick one summary as our canonical ground truth, automatic metrics would have to perfectly understand and judge the semantic similarity of the candidate with the reference. For human evaluations, we run into problems of subjectivity, reproducibility, evaluation protocols, and cost. Unsurprisingly, summarization (and, in general, natural language generation) evaluation remains an active field of study.

Additional complexities for evaluation are introduced through the sourcing process of our datasets, where collected references were not necessarily created with the goal of providing a summary. We will elaborate on these in the description of the respective dataset in Section 2.5.

2.4.1 Automatic Metrics

Compared to human evaluation, automatic evaluation metrics are cheap, fast, and scalable. Therefore, a multitude of automatic evaluation metrics has been developed. We survey the most widely used in the following.

BLEU. Developed for machine translation, BLEU stands for bilingual evaluation understudy (Papineni et al., 2002). It operates on sentence pairs and compares a candidate to a reference translation. Evaluation is based on lexical overlap of word n-grams, and typically BLEU-1 to BLEU-4 are reported (and sometimes averaged). An additional brevity penalty punishes candidates that are too short compared to the reference.

ROUGE. The standard metric to automatically evaluate summarization systems is Recall-Oriented Understudy for Gisting Evaluation (ROUGE, Lin, 2004). It measures textual overlap between the generated candidate and the reference summaries, after tokenization and stemming. There exist multiple variants of ROUGE, but it is common to report unigram and bigram overlap (ROUGE-1, ROUGE-2), as well as the longest common subsequence (ROUGE-L). Overlap computation results in precision P and recall R scores, which are combined with the harmonic mean into the F1 score, which is usually reported.

$$\text{F1} = \frac{2PR}{P+R} \tag{2.40}$$

ROUGE was found to prefer longer summaries over shorter ones (Sun et al., 2019). This can be solved by either comparing only summaries of equal (or similar) length or by normalizing with a random summary's score of the same length.

METEOR. Although less frequently used, the Metric for Evaluation of Translation with Explicit ORdering (METEOR, Banerjee and Lavie, 2005) is sometimes also reported. It combines unigram precision and recall with subsequence matching. For unigrams, it applies stemming and matches synonyms. Recall is weighted 9 times higher than precision in its weighted F-score. A penalty discourages fragmented matches of the subsequences of the reference summary.

BLEURT. Bilingual Evaluation Understudy with Representations from Transformers (BLEURT, Sellam et al., 2020) evaluates the quality of a candidate text with respect to a reference. It is trained in multiple stages to eventually predict human ratings. Starting from a pretrained BERT model, it then trains on synthetic sentence pairs before being finetuned to predict human annotations from the WMT Metrics shared task of the years 2017 to 2019.

BERTScore. BERTScore (Zhang et al., 2020b) is a semantic similarity metric between candidate and reference summary. It uses a pretrained BERT (Devlin et al., 2019) or RoBERTa (Liu et al., 2019a) model to obtain contextual representations for each input token, then constructs a cosine similarity matrix between each token of the candidate and reference. From the similarity matrix one can compute precision (as the highest similarity for each candidate token) and recall (highest similarity per reference token). The F1 score computed from precision and recall is the output BERTScore. If multiple references are present, the authors propose to

report the maximum BERTScore to any one of them.

BARTScore. As can be told from the name, BARTScore (Yuan et al., 2021) is using a pre-trained BART model (Lewis et al., 2020) to evaluate natural language generations. BARTScore computes the log probability of a text, conditioned on a different text. It can thereby compute faithfulness (probability of summary given source document), precision (probability of summary given reference), and recall (probability of reference given summary). The latter two can again be combined into an F1 score for semantic similarity.

Novel n-grams. The fraction of novel n-grams in the summary that do not appear in the source document measures its abstractiveness. It is most common to use $n = 2$ and report the fraction of novel bigrams, as unigrams are often too noisy (they include accidental matches), and for $n > 2$ the number of partial matches increases and is not accounted for.

Extractive fragments. Grusky et al. (2018) define *extractive fragments* as the shared sequences between summary and source document. They are greedily determined by stepping through the summary and finding the longest matching sequence in the source text if such a match exists for the current summary token. They compute two measures from the set of extractive fragments. *Extractive fragment coverage* is the fraction of words that are part of an extractive fragment, similar to novel n-grams above. *Extractive fragment density* is a measure for the length of the extractive fragments. It is the sum of quadratic lengths, divided by the number of summary tokens. Thus, it is disproportionally impacted by long extractive fragments.

Compression ratio. A straightforward summarization metric is the compression ratio, which divides the number of words in the source document by the words in the summary (Grusky et al., 2018).

Problems with Automatic Evaluation

Several problems with automatic evaluation methods exist. First, as already mentioned, there is no canonical summary for a source document. Many good summaries can be written, and humans may disagree on what the best summary is. Second, reference-based metrics depend on good reference summaries. The comparison to a reference with low coherence (CNN/DailyMail) or hallucinations (XSum) will underestimate the quality of good model summaries. Third, some metrics are based on the surface forms instead of the semantic meaning of a sentence. ROUGE tries to alleviate this by stemming, METEOR by taking synonyms into account. However, there remain many ways to express the same concepts using different words that are not synonyms. Fourth, methods such as BERTScore and BARTScore capture semantic meaning with a model-based approach, but the semantic similarity they measure can only

ever be as good as the model they use. The proxy target for a summarization model becomes to get good scores by the evaluation model, not to generate a good summary. Fifth, these metrics are vulnerable to being exploited by methods that choose to optimize them without consideration for the summary's quality or readability. There are even methods that directly optimize these metrics during training with reinforcement learning – since they are otherwise non-differentiable. Sixth, prior work has found that the metrics currently in use have been designed for a scoring range that is below what current models achieve (Peyrard, 2019b). As a result, the metrics are not well calibrated at higher scores. They disagree on the summaries produced by the current state-of-the-art models and should be re-calibrated. Finally, the best choice of automatic evaluation metric (as measured by correlation with human judgments) can vary between datasets (Bhandari et al., 2020).

2.4.2 Human Evaluation

In light of all these shortcomings of automatic evaluation, human evaluation is still the gold standard of summarization evaluation. There are many qualities that can be evaluated separately by asking humans to rate them in a given summary.

Human Evaluation Dimensions

For the dimensions of human evaluation, change is the only constant. As summarization models have improved over time, previously used criteria have become obsolete, and new ones have been added. Consequently, there is no standard set of dimensions along which studies evaluate, not even those that are conducted at the same time. Nevertheless, the Document Understanding Conferences (DUC) of the years 2004 and 2005 have contributed to at least some standardization of evaluation criteria and their definition. In DUC 2004 (Over and Yen, 2004), questions for grammaticality, referential clarity, redundancy (repetition), conciseness, and coherence as posed. In DUC 2005 (Dang, 2005), conciseness is removed but focus is added. More recently, several studies propose to use the four dimensions of coherence, consistency, fluency, and relevance (Kryscinski et al., 2019; Fabbri et al., 2021).

Grammaticality. Grammatical summaries should not include bad formatting, capitalization, or grammatical errors. In the DUC guidelines, grammaticality is linked to fluency in that ungrammatical summaries are mentioned to be hard to read.

Referential clarity. The reader should be able to identify who or what nouns and pronouns refer to. Additionally, an entity's relation to the rest of the summary should be clear.

Redundancy. Sometimes also termed repetition, or framed as non-redundancy (so higher scores are better), this criterion evaluates if words and phrases appear multiple times in

summaries. This is closely related to a language model's tendency to get stuck in generating repetitions (Holtzman et al., 2020).

Conciseness. Conciseness measures how succinct, or to the point, the important information is conveyed. Typically, it is synonymous with a shorter summary length. It thus has connections to the compression ratio from Grusky et al. (2018).

Coherence. A well-structured and well-organized summary achieves high coherence. The individual summary sentences should build on each other and collectively make sense, instead of being a series of unconnected facts. Due to their construction, CNN/DailyMail reference summaries score low on coherence (see Section 2.5.1).

Focus. The entire summary should talk about a specific event, and not move from topic to topic. This criterion is related to coherence, but the two nevertheless appeared as separate evaluation dimensions in DUC 2005.

Consistency. A good summary should be consistent, and neither contradict the source document nor itself. This criterion also prohibits hallucinations, i.e. summary content that is not supported by the source document.

Fluency. Fluency mostly evaluates the readability of text, which is also influenced by grammaticality. Fluency is sometimes understood a bit more generally, as it can also penalize text that is grammatical but does not flow nicely or contains uncommon words. In Zhang et al. (2023), fluency is no longer used as an evaluation criterion. The authors argue that all current models are mostly fluent, and it is no longer useful to evaluate.

Relevance. Sometimes also termed informativeness, relevance measures if only the important information from the source document is included in the summary.

Coverage. With coverage, we measure how much of the important information in the source document was reproduced in the summary. If relevance is important information's precision, coverage is its recall.

Abstractiveness. The more abstractive a summary is, the less it copies from the source document and instead rephrases the information in novel terms. It is debatable whether this in itself is a desirable property of a summary, but it certainly allows for summarizing certain information more concisely.

Faithfulness. Faithfulness determines whether the summary can be inferred from the source document. If a statement cannot be inferred, we call it a hallucination. Hallucinations can either occur when information from the source document has been combined in the wrong way, or when information not present in the source document is generated.

Factuality. Factuality measures the amount of information in a summary that is factually correct. This is different from faithfulness. A fact can be hallucinated when it is not present in the source document, but it can still be factually accurate. With pretrained language models, this is often the case, as factually correct world knowledge has been obtained by the model during pretraining and enters summary generation.

Problems with Human Evaluation

The motivation for automatic evaluation is also the most obvious limitation of human evaluation: it is expensive, time-consuming, and not easily scaled when controlling for quality, even with modern crowd-sourcing platforms. The literature on best practices and pitfalls in human evaluation is vast, so we only mention a few prominent issues here. Crowd annotators were found to conflate individual dimensions with the overall score (Fabbri et al., 2021), so evaluating the individual dimensions of a summary becomes more difficult. A possible solution is to properly train evaluators with examples for calibration, but subjects can still focus on surface-level and fluency-related aspects of quality after training, and give contradictory reasons for their judgment (Clark et al., 2021). Reaching high inter-annotator agreement is hard (Goyal et al., 2022), and agreement between experts and crowd workers is low (Fabbri et al., 2021). Even between experts, agreement was found to be low without mediation meetings (Iskender et al., 2021). Sometimes, expert annotators can continue to disagree even after discussion (Fabbri et al., 2021).

In general, the subjectivity underlying the task of human evaluation coupled with the inconsistency of human nature makes human evaluation very difficult to reproduce. To increase the reliability of human evaluation, a stricter evaluation protocol as well as a more structured evaluation have been proposed (Clark et al., 2021; Liu et al., 2022a). We now look at one such structured evaluation, the Pyramid method.

Pyramid Scores

Pyramid scores are an attempt to make human evaluation of relevance in summarization more comparable and reliable (Nenkova and Passonneau, 2004). It requires multiple reference summaries to be present, something that can not always be guaranteed (see Section 2.5). Annotators compare multiple reference summaries for the same source document and extract summarization content units (SCU). The SCUs correspond to the independent facts that summaries (and their individual sentences) consist of. The importance of an SCU is defined by

counting the reference summaries in which it appears. The most important SCUs will have a high count, but there will only be few SCUs that appear in (almost) all summaries. In contrast, many unimportant SCUs will have counts of 1 or 2. If we assign the SCUs with importance 1 to the bottom level of a pyramid, the SCUs with importance 2 to the second level, and so on, the number of SCUs will decrease with every level. This gives the pyramid its characteristic shape, and the method its name. A candidate summary is now evaluated with the help of the pyramid by finding the SCUs it contains, and then summing the corresponding importance weights of the SCUs. A candidate's score is normalized by the score of an optimal summary with the same number of SCUs. This optimal summary is constructed by picking SCUs from the top of the pyramid until the target number has been reached. Therefore, the pyramid score of each candidate is between 0 and 1. In a later work, the same authors note that normalizing by an optimal summary with the candidate's SCU count measures precision (Nenkova et al., 2007). To measure recall instead, one can normalize by an optimal summary with the mean number of SCUs in the reference summaries. They call this the *modified pyramid score*.

2.5 Datasets

In the following, we present text summarization datasets, with a focus on the task of single-document summarization (§ 2.5.1). We also briefly present multi-document (§ 2.5.2) and long-document summarization (§ 2.5.3) afterwards.

2.5.1 Single-Document Summarization

The standard setting for abstractive summarization is to summarize a single document. Consequently, most available datasets fall under this category. Due to the availability of data, most datasets are from the news domain. The most widely used datasets are CNN/DailyMail and XSum, due to their large number of examples. We present the datasets in chronological order and collect dataset statistics in Table 2.1.

DUC 2003/2004. The DUC 2003 and 2004 tasks (Over and Yen, 2003, 2004) provide small datasets for short and very short summarization of single and multiple documents. The individual tasks contain less than 1,000 examples. For very short summarization, the target length is 10 words, and for short summarization 100 words. The DUC dataset is used for testing since it is too small for training but multiple reference summaries exist and careful human annotations (e.g. Pyramids) have been performed. These are used to measure correlation with human judgments. The data can be obtained from the 2003[1] and 2004[2] task homepages.

[1] https://duc.nist.gov/duc2003/tasks.html
[2] https://duc.nist.gov/duc2004/

Dataset	Examples	Mean document words	Mean summary words
Single-document summarization			
DUC 2003/2004	<1,000	500	10–100
Gigaword	3,995,559	31	8
New York Times	654,759	800	46
CNN/DailyMail	312,085	685	52
XSum	226,711	431	23
Newsroom	1,321,995	659	27
Curation Corpus	39,911	504	83
WikiHow	204,004	580	62
SAMSum	16,369	94	20
Multi-document summarization			
WikiSum	2,332,000	1,840	78
Multi-News	56,216	2,103	264
WCEP	2,390,000	3,866	32
Long-document summarization			
PubMed	133,215	3,016	203
arXiv	215,913	4,938	220
BigPatent	1,341,362	3,573	117
Novels (chapters)	6,288	5,165	373
BookSum (chapters)	12,630	5,102	505
BookSum (books)	405	112,885	1,167
SQuALITY	625	5,200	237

Table 2.1: Dataset statistics for single-, multi- and long-document summarization.

Gigaword. The Annotated English Gigaword (Graff et al., 2003; Parker et al., 2011; Napoles et al., 2012) was created by pairing the first sentence of an article with its title, for the task of headline generation. In preprocessing, a large number of examples is removed according to heuristics (Rush et al., 2015). The remaining dataset contains 3.8M training, 190k validation, and 1,951 test pairs. The average word count is very low with 31 for source documents and 8.3 for summaries. The preprocessed corpus is available on GitHub.[3]

New York Times. The New York Times Annotated Corpus (Sandhaus, 2008) contains 1.8 million articles with 655k abstractive summaries written in the years 1987 to 2007. Different versions are in use for summarization with either 111k (Durrett et al., 2016) or 655k (Paulus et al., 2018) examples. For the latter version, average documents consist of 800 words, and summaries of 46 words (Narayan et al., 2018a). According to the corpus homepage[4] it cannot be freely accessed.

CNN/DailyMail. One of the most widely used summarization corpora is based on news articles from the CNN and Daily Mail websites. It was originally proposed as a question answering

[3] https://github.com/harvardnlp/sent-summary
[4] https://catalog.ldc.upenn.edu/LDC2008T19

dataset (Hermann et al., 2015), but then slightly adapted for summarization (Nallapati et al., 2016). The summary sentences are a concatenation of human-generated highlights: bullet points that accompany the original news articles. In some cases, these highlights can also contain links to related articles or standings of particular sports leagues. The corpus's source documents contain on average 685 words and the summaries 52 words. In the original paper, entities were replaced with per-document integer ids (anonymized version), but the standard version keeps the entities from the original articles (See et al., 2017). Scripts for generating the corpus can be found on GitHub.[5] We use the preprocessed version from Liu and Lapata (2019b) in this thesis.

XSum. The extreme summarization dataset (abbreviated as *XSum*) is created from BBC news articles (Narayan et al., 2018a). The summaries are taken to be the first sentence of the article (appears in bold on the website), and the source document is the rest of the article. The title is dropped. Choosing the lead sentence as the summary is motivated by the inverted pyramid writing style (Pöttker, 2003), in which that sentence is supposed to summarize the most important information of the article, answering the questions of who, when, where, what, and sometimes why. XSum specializes in creating a very short abstractive summary that is still a well-formed sentence, as opposed to headline generation. The news articles appeared between 2010 and 2017. The dataset was randomly split into 204k training pairs, and 11k validation and test pairs each. The average word count in the source documents is 431, and 23 for the summaries. The corpus is available on GitHub.[6]

Newsroom. The NEWSROOM dataset consists of 1.3 million examples written between 1998 and 2017 (Grusky et al., 2018). It collects these from 38 different major news publications, to combine different writing and summarization styles in a single dataset. The summaries are taken from the articles' HTML metadata used by search engines and social media, so are provided by the publishers themselves. Source documents are on average 659 words long, and summaries 27 words. Compared to other news summarization datasets, the compression ratio (see Section 2.4.1) is rather high. The data can be requested on the dataset website.[7]

Curation Corpus. The Curation Corpus (Curation, 2020) is a dataset of professionally written summaries of news articles. The corpus is an order of magnitude smaller than CNN/DailyMail, and its articles and summaries have fewer but longer sentences (see Table 2.1). The 40k articles have an average sentence length of 504 words, while the summaries span on average 83 words. The Curation Corpus is the only freely available news summarization dataset with references that were written for the purpose of summarizing the article. We therefore consider this a very interesting dataset in spite of its smaller size. The data can be obtained by following the

[5] https://github.com/abisee/cnn-dailymail
[6] https://github.com/EdinburghNLP/XSum
[7] https://lil.nlp.cornell.edu/newsroom/index.html

instructions on the dataset's GitHub.[8]

WikiHow. The WikiHow dataset was constructed from a knowledge base of how-to articles, explaining how to solve a task (Koupaee and Wang, 2018). Each task description is a series of multiple steps, and each step starts with a bold line summarizing that step, followed by a detailed step description. The summary is formed from the concatenation of the step summaries, and the articles from the concatenation of step descriptions. The dataset consists of 204k examples with 580 mean source document words, and 62 mean summary words. The dataset is available on GitHub.[9]

SAMSum. Abstractive dialogue summarization is the task of the Samsung Abstractive Messenger Summarization (SAMSum) dataset (Gliwa et al., 2019). It is constructed to resemble the chats of a mobile messenger app. Each dialogue is written by a single linguist, can be formal or informal, and potentially contains slang, emoticons or typos. Around 75% of dialogues are between two participants, and the number of utterances per dialogue ranges from 3 to 30. The summaries are created to be short and relevant, contain the names of the participants, and are written in the third person. The dataset consists of 16k total examples. It can be downloaded from the ancillary files website of the paper's arXiv preprint.[10]

2.5.2 Multi-Document Summarization

In multi-document summarization, a summary of multiple source documents reporting on the same topic or event has to be written. We look at the three best-known multi-document summarization datasets, before giving a short glimpse at different strategies that have been proposed to tackle this task.

WikiSum. In the WikiSum dataset (Liu et al., 2018), the lead paragraph of a Wikipedia topic is used as a target summary, and the union of references in the Wikipedia article and Google search results (from the article's title) are used as the source documents. The total number of articles-summary pairs (called clusters) is 2.3 million. The median cluster has a summary length of 78 words, 2 Wikipedia references, and 26 search results. The total number of source words in a typical cluster is 52k, with a mean of 1,148 words for Wikipedia references and 1,893 words for search results. Instructions on how to generate the dataset are on GitHub.[11]

Multi-News. The Multi-News dataset (Fabbri et al., 2019) consists of 56k clusters, crawled from a news aggregation website. The sources are they diverse; they come from more than

[8] https://github.com/CurationCorp/curation-corpus
[9] https://github.com/mahnazkoupaee/WikiHow-Dataset
[10] https://arxiv.org/src/1911.12237v2/anc
[11] https://github.com/tensorflow/tensor2tensor/tree/master/tensor2tensor/data_generators/wikisum

1,500 different news sites. The clusters have 2 – 10 source documents, with an average of 2.3. Gholipour Ghalandari et al. (2020) note that the curation of relevant sources before writing the summary is a special setting. The WikiSum and WCEP datasets instead request to summarize an uncurated collection of source documents. The concatenation of source documents has on average 2.1k words, and the summaries have 264 words (10 sentences), which is comparably long. The dataset is available with different amounts of preprocessing on GitHub.[12]

WCEP. The Wikipedia Current Events Portal (WCEP) dataset (Gholipour Ghalandari et al., 2020) contains 10k clusters with short summaries of news events. The original summary cites 1.2 articles on average, so the sources are extended with articles from Common Crawl News, selected with heuristic features. According to the authors' manual inspection, about 18% of these additional articles are unrelated to the summary. The dataset comes in two versions: a full version, and one where random articles from Common Crawl News are added to the cited articles until 100 source documents are reached. The typical cluster has 78 articles and 29 summary words in a single summary sentence. The dataset can be downloaded from GitHub.[13]

Multi-document summarization strategies. As became apparent from the dataset descriptions, the challenge of multi-document summarization is to handle the sheer size of input text that has to be analyzed and summarized, especially in WikiSum and WCEP. Several approaches divide this task into two stages, a retrieval and a summarization stage.

In the retrieval stage, models usually operate on paragraphs. If a proxy for the target summary is available (in WikiSum the title of the target Wikipedia article is given), the similarity of a paragraph to the proxy can be determined from the cosine similarity of tf-idf vectors of title and paragraph (Liu et al., 2018), or a similarity score from a neural network (Liu and Lapata, 2019a). If no such proxy is known, the best set of paragraphs can be determined with determinantal point processes (Cho et al., 2019) or submodularity (Lin and Bilmes, 2010).

In the summarization stage, multiple paragraphs can be concatenated and summarized as in single-document summarization. If the concatenation contains too many words (up to 11k tokens in WikiSum), operating on sliding windows (Liu et al., 2018), compressing internal representations (Liu et al., 2018), employing hierarchy by encoding paragraphs first (Liu and Lapata, 2019a), or using graph representations (Li et al., 2020b) have been proposed.

2.5.3 Long-Document Summarization

Summarizing long inputs is not just encountered in the multi-document setting, but is also a dedicated task. Long-document summarization is especially challenging for the Transformer

[12] https://github.com/Alex-Fabbri/Multi-News
[13] https://github.com/complementizer/wcep-mds-dataset

architecture since its self-attention scales quadratically with the input length. Several models have been proposed to reduce self-attention complexity (Dai et al., 2019; Beltagy et al., 2020; Katharopoulos et al., 2020; Jaegle et al., 2021; among others). We describe these general approaches in more detail in Section 4.7. Next, we describe the most common long-document summarization datasets.

PubMed and arXiv. PubMed and ArXiv are collections of scientific articles from the PubMed and arXiv repositories, respectively (Cohan et al., 2018). The papers' abstracts serve as the target summary. The PubMed dataset contains 133k examples, with an average source document length of 3,016 words, and a summary length of 203 words. The arXiv dataset consists of 215k pairs, with source documents averaging 4,938 words, and summaries 220 words. Both datasets are available on GitHub.[14]

BigPatent. The BIGPATENT dataset is a collection of 1.3 million US patent documents across nine technical areas (Sharma et al., 2019). The target summaries are the patent abstracts. Patents were filed between the years 1971 and 2019. The dataset's documents contain an average of 3,573 words, and its summaries 117. The data can be downloaded from the dataset website.[15]

Novels. An unnamed dataset on summarizing novel chapters has been proposed in Ladhak et al. (2020). They pair the chapters of novels from Project Gutenberg with summaries from five online study guides. The dataset consists of 6,288 chapter-summary pairs, with a mean chapter length of 5,165 words, and a summary length of 373 words. Instructions on how to obtain the dataset for yourself are on GitHub.[16]

BookSum. The BOOKSUM dataset is composed of literature from Project Gutenberg (novels, plays, and stories) where copyrights have expired (Kryscinski et al., 2022). The dataset provides summaries at three levels: paragraph, chapter, and full-text. The chapter and full-text summaries were retrieved online. Paragraph summaries were constructed by aligning chapter summary sentences according to the similarity of their embeddings. The dataset comprises 147k paragraph-level, 12.6k chapter-level, and 405 book-level examples. It is interesting to note that the mean chapter summary consists of 505 words, while for the full text, it is 1,167 words. This is in contrast to related work that kept summary lengths at different levels stable (Wu et al., 2021). The source documents and instructions on how to obtain the summaries are on GitHub.[17]

[14] https://github.com/armancohan/long-summarization
[15] https://evasharma.github.io/bigpatent/
[16] https://github.com/manestay/novel-chapter-dataset
[17] https://github.com/salesforce/booksum

SQuALITY. The SQuALITY dataset consists of summaries of Project Gutenberg short stories written between 1930 and 1970 (Wang et al., 2022). Each short story is summarized by four writers (freelancers and undergraduates), who also answer four additional story-specific questions, which results in 4 general and 16 aspect-oriented summaries per short story. The dataset contains summaries for 625 source documents. The stories have on average 5,200 words, and the summaries 237. The data can be downloaded from GitHub.[18]

What is the current state of long-document summarization? Koh et al. (2022) compare models that reduce the complexity of self-attention to models that reduce the input length in the first step and then summarize the shorter input. For the second variant, they use an oracle reduction of the input, so it should be considered an upper bound for the method's performance. Considering this, they find that both approaches achieve high ROUGE scores, with reduce-then-summarize models generating more relevant but less factual summaries. They speculate that the reduced factuality comes from incoherent texts that are output by the initial filtering stage.

When considering the factuality at the summary level, the best model generates 21% factually inconsistent summaries for the arXiv dataset, and 60% inconsistent summaries on the Gov-Report (Huang et al., 2021) dataset. The two most common factuality error types are wrong primary arguments (like entities) of the predicate, and mistakes in linking multiple statements in the discourse. The authors conclude that the models are not sufficiently robust to be used in practice.

2.6 State of the Art

In the final section of this chapter, we want to look at the state of the art on abstractive summarization, as of the writing of this thesis in March 2023. We present the best results known to us on the two largest summarization benchmarks, CNN/DailyMail and XSum. We first report on finetuned encoder-decoder models that have historically shown strong performance on summarization. Afterward, we show the results of a very recent study that suggests that zero- and few-shot large language models have surpassed finetuned models in summary quality.

2.6.1 Finetuned Models

We get an overview of available models from the resources and benchmarking platform *Papers with Code*.[19] In Table 2.2, we present ROUGE-1/2/L of the most influential models in our eyes, together with the best-performing models. ROUGE scores were selected as the performance metric since they are the most widely accepted automatic metric, and scores are available for

[18] https://github.com/nyu-mll/SQuALITY
[19] https://paperswithcode.com/

Dataset	ROUGE-1	ROUGE-2	ROUGE-L
CNN/DailyMail			
MoCa	48.88	24.94	45.76
SLiC	47.97	24.18	44.88
BRIO	47.78	23.55	44.57
SummaReranker	47.16	22.55	43.87
MatchSum (extractive)	44.41	20.86	40.55
PEGASUS	44.17	21.47	41.11
BART	44.16	21.28	40.90
T5-11B	43.52	21.55	40.69
BERTSUMEXTABS	42.13	19.60	39.18
Pointer-Generator	39.53	17.28	36.38
XSum			
SLiC	49.77	27.09	42.08
MoCa	49.32	25.91	41.47
BRIO	49.07	25.59	40.40
SummaReranker	48.12	24.95	40.00
PEGASUS	47.12	24.56	39.25
BART	45.14	22.27	37.25
BERTSUMEXTABS	38.81	16.50	31.27
Pointer-Generator	29.70	9.21	23.24
MatchSum (extractive)	24.86	4.66	18.41

Table 2.2: State-of-the-art results on CNN/DailyMail and XSum.

all models. We present the performance as reported in the original papers.

The extractive MatchSum (Zhong et al., 2020) achieves good scores on the CNN/DailyMail dataset, for which reference summaries are known to be highly extractive (Grusky et al., 2018). On XSum, which is defined to be especially abstractive, an extractive method cannot reach high scores. The Pointer-Generator (See et al., 2017) is a representative of RNNs with attention. We see that the gap to the pretrained models is larger on XSum than on CNN/DailyMail, most likely due to the differences in language generation capabilities between RNNs and (pretrained) Transformers. The next iteration of approaches uses different pretraining objectives to train a strong language understanding model, that can then be finetuned on a target task. BERTSUMEXTABS (Liu and Lapata, 2019b) pretrains only the encoder, while T5-11B (Raffel et al., 2020), BART (Lewis et al., 2020) and PEGASUS (Zhang et al., 2020a) pretrain an encoder-decoder model, before finetuning on summarization. PEGASUS's pretraining objective of masking and reconstructing important sentences is tailored to the summarization task, and it consequently achieves the highest scores of these models on both benchmarks.

The current best-performing models achieve another boost in ROUGE scores by addressing exposure bias (see Section 2.3.1). Since model generations during inference can diverge from the ones the model sees during training, recent approaches train a model that selects the best

of multiple generated candidate summaries or calibrate the model's sequence likelihood to accurately rank its generations. SummaReranker (Ravaut et al., 2022) reranks 30 candidate summaries generated by PEGASUS with beam search and diverse beam search (Vijayakumar et al., 2018) decoding. BRIO (Liu et al., 2022b) trains a model to both generate summaries and rank its generations, according to their ROUGE score. It uses BART for CNN/DailyMail and PEGASUS for XSum to generate 16 candidate summaries with diverse beam search, then calibrates the model's log probability with a contrastive loss between candidate pairs. Momentum Calibration (MoCa, Zhang et al., 2022b) uses a very similar approach but employs the same model with separate parameters for ranking and generation. The ranking model is trained as in BRIO. The generation model's parameters are updated with a weighted average of its own and the ranking model's parameters. This slow update avoids a collapse of training or too fast convergence of the ranking model. Finally, *sequence likelihood calibration* (SLiC, Zhao et al., 2023) introduces a new training objective to align the model's likelihood of summaries if they are similar in the model's latent space. The similarity is computed as in BERTScore (see Section 2.4.1), taking into account matching spans of more than one token. The embeddings are taken from the decoder output representations of the two sequences conditioned on the article. The calibration loss then is a ranking loss that trains the model to have a higher likelihood for candidate summaries with higher latent space similarity to the reference. Unlike the previous reranking and calibration methods, SLiC does not use ROUGE as a target signal. For the results in Table 2.2, it uses a larger base model with around 2B parameters, compared to the other approaches, whose models are about four times smaller. In general, the differences between the model's scores are small; they all achieve very high ROUGE scores on both datasets.

Are higher ROUGE scores still desirable? According to a recent human evaluation of large language models in Zhang et al. (2023), LLMs now generate higher-quality summaries than the references of the aforementioned datasets. Consequently, higher lexical similarity with these references from ROUGE scores no longer implies higher quality for LLMs. There are two possible remedies. First, getting higher-quality reference summaries could resolve the problem. A new human evaluation would have to be conducted to compare the improved reference summaries with LLM generations. It would be interesting to see this evaluation be performed on other existing datasets first, for example, the Curation Corpus with its human-generated summaries for the purpose of summarization. However, aside from the difficulty of conducting a conclusive and reproducible human evaluation, better references will most likely be matched by better generated summaries soon. The second approach is to change the default automatic evaluation metric. This is an inevitable next step in summarization evaluation in our opinion and will be a continued focus of research in natural language generation in general.

2.6.2 Zero- and Few-Shot Large Language Models

Large language models have not been systematically evaluated against the entire CNN/DailyMail or XSum test set. Instead, we here report the results of Zhang et al. (2023), who have run a careful human evaluation on 100 examples from LLMs in the zero- and few-shot setting, two well-performing finetuned models, and the datasets' references. We present the results in Table 2.3. The evaluated LLMs are: GPT-3 (Brown et al., 2020), InstructGPT (Ouyang et al., 2022), Anthropic-LM (Bai et al., 2022), Cohere XL[20], and OPT (Zhang et al., 2022a). The finetuned language models are PEGASUS (Zhang et al., 2020a) and BRIO (Liu et al., 2022b), which we have already presented before. The human evaluation asks three annotators per example to rate whether summaries are faithful (binary score of 0 or 1), coherent, and relevant (1 to 5 Likert scale).

The main findings of the human evaluation are that (a) reference summaries are judged worse than (strong) model summaries, (b) LLMs outperform finetuned models when flawed reference summaries are used for finetuning, a verdict with stronger reference summaries is still outstanding, (c) instruction tuning improves zero-shot performance by a lot, and few-shot performance by a smaller but substantial margin. This last finding is certainly influenced by summarization being one of the tasks used for instruction tuning (4.2% in Ouyang et al., 2022). The best model from the human evaluation (zero-shot InstructGPT) is then compared against a newly collected set of summaries written by freelance writers for 50 articles from CNN/DailyMail and XSum, each. The summaries of the model and freelance writers are rated very similarly, with half the annotators preferring either. InstructGPT uses a more extractive summarization style, whereas freelance writers abstract and generalize a lot more. The difference in preference between annotators could originate from different preferences for summarization style. As a result, the study's authors suggest moving from an intrinsic evaluation of summary quality to judging a summary's usefulness in downstream applications, in line with Clark et al. (2021).

[20] https://docs.cohere.ai/docs/generation-card

Dataset	Faithfulness	Coherence	Relevance
CNN/DailyMail			
Reference summaries	0.84	3.20	3.94
PEGASUS	0.97	3.93	4.38
BRIO	0.94	3.94	4.40
GPT-3 (zero-shot)	0.76	2.65	3.50
GPT-3 (few-shot)	**0.99**	3.95	4.34
InstructGPT (zero-shot)	**0.99**	**4.15**	**4.60**
InstructGPT (few-shot)	**0.98**	**4.13**	4.49
Anthropic-LM (few-shot)	0.94	3.88	4.33
Cohere XL (few-shot)	**0.99**	3.42	4.48
OPT (few-shot)	0.96	3.64	4.33
XSum			
Reference summaries	0.37	4.13	3.00
PEGASUS	0.57	4.73	3.85
BRIO	0.58	4.68	3.89
GPT-3 (zero-shot)	0.80	2.78	3.52
GPT-3 (few-shot)	0.69	4.69	4.03
InstructGPT (zero-shot)	**0.97**	4.41	**4.28**
InstructGPT (few-shot)	0.77	**4.83**	**4.33**
Anthropic-LM (few-shot)	0.70	**4.77**	4.14
Cohere XL (few-shot)	0.63	**4.79**	4.00
OPT (few-shot)	0.67	**4.80**	4.01

Table 2.3: Results of the human evaluation of three quality criteria from Zhang et al. (2023). Few-shot prompts include 5 training examples from the respective dataset that fit into the prompt context window. Scores are in the interval $[0,1]$ for faithfulness and $[1,5]$ for coherence and relevance. Results not statistically significantly different from the best are bolded.

Part I Interpretable Representation Learning

3 Summary Content Planning

Chapter Summary

Neural text generation models generate a summary word by word, but a summary would optimally be planned at a higher level (e.g. the sentence) before being realized at the word level. Related work has focused on hierarchy on the encoder side, introduced a coverage loss to avoid repeatedly summarizing the same content, or planned the sequence of entities that should be covered in the summary. Instead, we introduce hierarchy on the decoder side and predict a continuous sentence representation of the next summary sentence. This representation is then used to write the next summary sentence by the word generator. Our model is called the *sentence planner*. On two common datasets, it improves ROUGE scores and abstractiveness compared to an – at the time state-of-the-art – baseline model.

Publication in this Chapter

This chapter builds on the material in our publication:
Marfurt, A., and Henderson, J. (2021). Sentence-level Planning for Especially Abstractive Summarization. In *Proceedings of the Third Workshop on New Frontiers in Summarization* (pp. 1-14).
PDF: https://aclanthology.org/2021.newsum-1.1.pdf
Code: https://github.com/idiap/sentence-planner
Video: https://screencast-o-matic.com/watch/cr6XqEVXO5n

In this chapter, we experiment with giving the model the ability to outline the next summary sentence before creating it. We posit that humans also benefit from structuring their thoughts first before writing down the words of a text. We expect a so-created text to be more coherent, and also more abstractive. On the basis of this idea, we devise a hierarchical decoder that generates a plan before the words of a summary sentence.

3.1 Introduction

In abstractive summarization, we aim for a summary to be fluent and coherent, and at the same time synthesize the source document's important information (see Section 2.4). State-of-the-art sequence-to-sequence models successfully write fluent summaries (Liu and Lapata, 2019b; Lewis et al., 2020; Zhang et al., 2020a). To achieve high ROUGE scores, however, they also heavily rely on copy mechanisms such as the pointer-generator network (See et al., 2017) or attention to the source document (Bahdanau et al., 2015; Rush et al., 2015). This comes at the cost of the abstractiveness[1] and coherence of the resulting summaries. The effect is further exacerbated by datasets that were constructed from sources that are not fully aligned with the goal of summarization. The highlights in CNN/DailyMail, for example, were not meant to be read as a human-written summary, but as bullet points presenting a quick overview to the reader, and sometimes contain additional information and links to related articles (see Section 2.5.1). As a result, the concatenation of these sentences is not always a coherent summary of the article.

In this chapter, we conjecture that adding a planning step at the sentence level alleviates these problems without sacrificing high ROUGE scores. By first producing an outline for the next summary sentence, we give the model more capacity for abstraction. As a result, the model has to rely less on copying the input and thereby generates more abstractive summaries. Our model, the *sentence planner*, is an adaptation of the typical encoder-decoder architecture. The encoder is initialized from pretrained BERT weights. The decoder is hierarchical and consists of a sentence generator that plans an outline for the summary at the sentence level, and a word generator that is conditioned on this outline when generating the summary's words. Both generators attend to the source document to condition their predictions on the input. The sentence planner is trained end-to-end to predict the words of the target summary, with an additional guidance loss that encourages the sentence generator to produce the encoder's embedding for the target next sentence.

We extensively evaluate our model on the highly abstractive Curation Corpus (see Section 2.5.1) and the established but more extractive CNN/DailyMail. We show that the sentence planner generates more abstractive summaries while improving the ROUGE scores of a base model without a hierarchical decoder. We use gradient attribution to quantify the impact of the

[1] More abstractive methods generally attain lower ROUGE scores. To see why, consider the case where the reference summary and the model copy from the document. The generated summary is guaranteed to get an exact match and high ROUGE. In the opposite case, where both the reference summary and the model generate novel text, there is a good chance that the choice of words is not exactly the same, resulting in low ROUGE.

sentence generator on the model's predictions as well as how much information from the document it captures. Moreover, we verify the effectiveness of our model components with an ablation study and show that simply increasing the baseline's decoder parameters does not bring it up to par with the hierarchical decoder. Our automatic evaluations are confirmed in a human evaluation study, where the sentence planner improves upon its strong baseline in each of the six quality categories.

3.2 Base Model

We use BERTSUMEXTABS (Liu and Lapata, 2019b) as our base model. It adapts the pretrained BERT model (Devlin et al., 2019) to summarization by applying a few changes, detailed below.

Encoding multiple sentences. BERT is built for encoding one or two segments (with possibly multiple sentences). A document to be summarized is a single segment with multiple sentences. To be able to get a representation for each of these sentences, a CLS token is inserted at the start of every sentence. In addition, the segment embeddings (in BERT they are called A for the first segment and B for the second) are interleaved for consecutive sentences, such that all odd sentences have segment embedding A, while all even sentences have B.

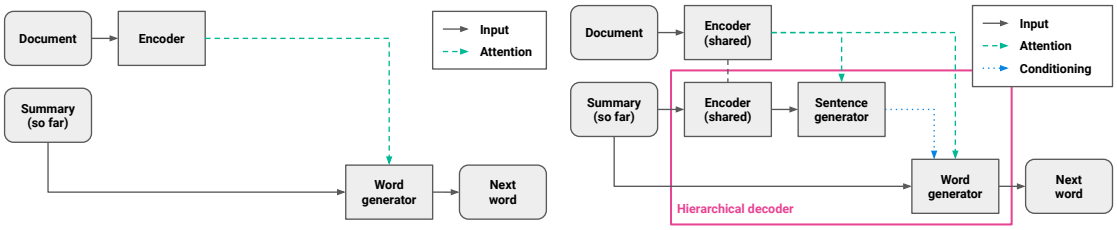
Extractive summarization. For the task of extractive summarization, the model needs to classify sentences. To get a representation for each sentence, two additional Transformer layers are added on top of the pretrained BERT model. They operate on the output representations at the positions of the CLS tokens. The output of the two inter-sentence Transformer layers is used to decide for each sentence whether it is in the extractive summary or not. A sigmoid classifier is used:

$$\hat{y}_i = \sigma(W h_i^L + b) \tag{3.1}$$

where i is the sentence index and L the last layer. The loss is binary classification entropy. The (randomly initialized) inter-sentence Transformer layers are finetuned jointly with the pretrained BERT model. This model is called BERTSUMEXT.

Abstractive summarization. In the abstractive setting, the pretrained BERT model is used as the encoder without the inter-sentence Transformer layers. A 6-layer Transformer decoder is added and randomly initialized. The encoder and decoder are trained jointly. However, the decoder has a higher learning rate and a shorter warmup period than the encoder. This model is named BERTSUMABS.

Extractive then abstractive training. A two-stage approach called BERTSUMEXTABS finetunes the encoder first on the extractive task, then removes the inter-sentence Transformer



(a) BERTSUMEXTABS (b) Sentence planner

Figure 3.1: (a) BERTSUMEXTABS model. An encoder encodes the document, and a word generator generates the next word given previous words while paying attention to the document. (b) Sentence planner model. A shared encoder separately encodes the document and each sentence of the summary generated so far. The sentence generator takes the summary sentence embeddings and predicts the next sentence embedding, which the word generator is then conditioned on. Both generators integrate document information through attention.

layers and finetunes the abstractive model.

3.3 Hierarchical Transformer Decoder

Our approach builds on the BERTSUMEXTABS model. Their model consists of an encoder initialized with an extractive summarization model, which in turn was initialized with a BERT model and a randomly initialized Transformer decoder.[2] We keep the encoder the same. We replace the decoder with a hierarchical version by introducing a sentence generator that develops a high-level plan for the summary, and a word generator that is conditioned on this plan. A model diagram is shown in Figure 3.1. Section 3.3.1 describes how the sentence generator develops the outline for the summary, and Section 3.3.2 shows how the word generator makes use of it.

3.3.1 Sentence Generator

The sentence generator is a two-layer Transformer decoder. It receives as inputs the sentence representations of completed summary sentences and generates a sentence representation for the next summary sentence.

Inputs. The inputs to the sentence generator are a sequence of representations of already completed summary sentences. These are computed by the same encoder that computes representations for the document tokens. For each individual previous summary sentence, the encoder computes its contextualized token embeddings. We use the contextual embedding

[2] Even stronger results have been achieved when pretraining an entire sequence-to-sequence model on a task closer to summarization (BART (Lewis et al., 2020), PEGASUS (Zhang et al., 2020a)). In this chapter, we restrict ourselves to encoder initializations with the BERT model and do not consider other pretraining approaches.

of the end-of-sentence token as a representation of the sentence.[3] When generating the first summary sentence, there are no completed sentences, so we use a single zero vector as input to the sentence generator.

During training with teacher forcing, we use the previous portion of the reference summary as input to the encoder. Since the entire summary is known in advance, we can compute all inputs to the sentence generator in parallel.

Self-attention. The sentence generator's self-attention operates at the sentence level, which means the sequence length n for our Transformer decoder is very small (between 2 and 4 on average, see Section 3.6). As a result, the self-attention computation, which is quadratic in the sequence length, becomes extremely cheap. As in regular Transformer decoders, a causal mask prevents attention to future sentences.

Cross-attention. In the cross-attention, the sentence generator pays attention to the encoded document. Through this connection, the sentence generator can compare the already generated summary to the document and identify missing information that should appear in the next sentence.

Output. The output of the sentence generator is a representation r_{sent} for the next summary sentence. Section 3.3.2 describes how we condition the word generator on this sentence representation.

Guidance loss. We provide the sentence generator with an additional loss term for guidance. Since during training, we know the ground truth next summary sentence and can compute its encoding r_{gold}, we penalize the (element-wise) mean squared error between the gold and the predicted next sentence representation.

$$\mathscr{L}_{\text{MSE}} = \frac{1}{d} \sum_{i=1}^{d} ||r_{\text{gold}}^{(i)} - r_{\text{sent}}^{(i)}||_2^2 \tag{3.2}$$

where d is the representations' dimension. This loss term is added to the regular cross-entropy loss with a scaling hyperparameter λ, although we found $\lambda = 1$ to work well in practice.

We do not backpropagate the guidance loss's gradients from the sentence generator into the encoder to avoid a collapse to a trivial solution. Otherwise, the encoder might output the same representation for every sentence so that the sentence generator can perfectly predict it.

[3] We found that this performed better than alternative encodings of the summary, as discussed in Section 3.4.

3.3.2 Word Generator

Our word generator is also a Transformer decoder. The regular Transformer decoder consists of layers l with self-attention, cross-attention, and feed-forward sublayers. They are defined as follows:

$$s^l = \text{LN}(h^{l-1} + \text{SelfAttention}(h^{l-1})) \tag{3.3}$$

$$c^l = \text{LN}(s^l + \text{CrossAttention}(s^l, r_{\text{enc}})) \tag{3.4}$$

$$h^l = \text{LN}(c^l + \text{FFN}(c^l)) \tag{3.5}$$

where LN is layer normalization (Ba et al., 2016), r_{enc} are the encoder outputs, and FFN is the feed-forward sublayer consisting of two fully-connected layers with an intermediate nonlinearity.

In our word generator, we condition on the sentence representation by replacing Equation 3.4 with

$$c^l = \text{LN}(s^l + \text{CrossAttention}(s^l, r_{\text{enc}}) + r'_{\text{sent}}) \tag{3.6}$$

where r'_{sent} is the sentence representation obtained from the sentence generator, passed through a fully-connected and a dropout layer. We do not differentiate between layers and add the same sentence representation in every layer and to every token.

We experimented with various ways to use attention in the word generator to integrate the sentence representation. However, the conditioning method presented above substantially outperforms the attention-based integrations of the sentence representation. We further discuss this topic in Section 3.4.

At the end of a sentence, the word generator either outputs a special sentence separator symbol, prompting the sentence generator to generate the next sentence representation, or an end-of-summary symbol, stopping generation.

3.4 Alternative Approaches

In the following, we discuss alternative approaches which we tried but did not achieve as good results as the proposed model.

Separate encoders for document and summary. We conjectured that encoding a document for cross-attention in the word generator, and encoding a summary for generating the next summary sentence representation require extracting different pieces of information. We therefore added a second encoder for the summary generated so far and initialized it with BERT. This change did not improve over sharing the encoder weights for the article and the summary. However, it introduced many additional parameters, so we discarded this idea.

Same preprocessing for the summary. BERTSUMEXTABS uses different preprocessing formats for the source document and the summary. For the document, every sentence is surrounded by a leading `CLS` token and a trailing `SEP` token. The summary is preceded by a *beginning of summary* token, the summary sentences are separated by a sentence separator token and the end is marked with an *end of summary* token.

We tried homogenizing the preprocessing formats for the document and the summary, such that the encoder does not need to deal with different inputs. We surround every sentence with a `CLS` and `SEP` token. The end of the summary is still marked with an *end of summary* token to tell the decoder to stop.

We did not reach the results of the preprocessing used in BERTSUMEXTABS with this format. Interestingly, the generated summaries consistently contained fewer sentences on average. We conjecture that this could be an artifact of decoding with beam search, but cannot substantiate this presumption.

Contextual sentence representations. In our model, we encode summary sentences individually, without self-attention to the surrounding sentences. It is not possible to allow representations to see future ground-truth sentences, as that would serve as a shortcut for the model and prevent proper learning of the task. While it is possible for the sentence representations to encode information from previous summary sentences, experiments showed no improvements with this change.

Attention to the sentence representation. A different way to integrate the sentence representation in the word generator is to perform attention over it. We experimented with two methods. On the one hand, we specialized an attention head to exclusively look at the sentence representation, while the others attend to the source document. This method performed slightly worse than the base model on ROUGE scores. On the other hand, we concatenated the sentence representation to the encoder outputs, and jointly attended to it in the word generator's cross-attention. When analyzing the attention weights, we realized that the sentence representation was mostly ignored. As a remedy, we separated training into two phases. In the first phase, we trained our model without attention to the document, such that the sentence planner gets a chance to learn meaningful sentence representations and is not ignored from the start. We then finetuned the model with attention to the document. While this increased the attention weights of the sentence representation substantially, the results did not improve over the baseline with the same number of total training steps (pretraining and finetuning combined).

Dataset	Examples	Mean doc length		Mean summary length		Novel bigrams	Corefs
		words	sentences	words	sentences		
CNN/DailyMail	312085	685.12	30.71	52.00	3.88	54.33%	0.105
Curation Corpus	39911	504.26	18.27	82.63	3.46	69.22%	0.441

Table 3.1: Dataset statistics.

3.5 Experimental Setup

We now describe our selection of datasets (§ 3.5.1) and metrics (§ 3.5.2) that we use to evaluate our model and give implementation details (§ 3.5.3) to replicate our experiments.

3.5.1 Datasets

For our experiments, we choose an established and well-studied dataset, CNN/DailyMail (Hermann et al., 2015; Nallapati et al., 2016), and a more recently introduced high-quality dataset, the Curation Corpus (Curation, 2020). A description of both datasets is given in Section 2.5.1. The datasets' statistics are shown in Table 3.1.

We selected these two datasets since in contrast to other widely used datasets (Gigaword (Rush et al., 2015), XSum (Narayan et al., 2018a)), the summaries span multiple sentences, which is a prerequisite for our approach. We use the preprocessed data from Liu and Lapata (2019b) and describe the preprocessing for the Curation Corpus next.

Preprocessing the Curation Corpus. We follow the instructions in the Curation Corpus GitHub repository[4] to download the 40000 article-summary pairs. After filtering examples where either the article or the summary is empty, we are left with 39911 examples. We split them into train/validation/test sets as 80/10/10 to arrive at split sizes of 31929/3991/3991.

Since the text extractor from the HTML websites inserts a lot of newlines (probably due to the website layout), we replace them with spaces to avoid splitting sentences in the middle.

We use the NLTK tokenizer (Bird et al., 2009) to split the article text into sentences. We then preprocess the data in the same way as Liu and Lapata (2019b) processed the CNN/DailyMail corpus, except that we do not filter examples based on the number of tokens in the article or summary, but instead keep them irrespective of their length.

3.5.2 Metrics

We use ROUGE and novel bigrams as automatic evaluation metrics (see Section 2.4.1). Additionally, we use a coreference resolution-based metric, described in the following.

[4] https://github.com/CurationCorp/curation-corpus

Corefs. Inspired by Iida and Tokunaga (2012), we evaluate discourse coherence with a coreference resolution model. We count the number of coreference links across sentence boundaries as a proxy for the coherence of a summary, i.e. whether the sentences build upon information in the preceding ones. Since summaries with more sentences could be favored by this count, we normalize by the number of sentences. To extract coreferences from the generated summaries, we use NeuralCoref[5] with spaCy[6]. The mean number of coreference links across sentence boundaries for the datasets' reference summaries is 0.441 for the Curation Corpus and 0.105 for CNN/DailyMail. This shows that the summaries in the Curation Corpus are written in a much more coherent style than the ones from CNN/DailyMail. Specifically, the bullet point style summaries in CNN/DailyMail do not foster summaries whose sentences build on each other. However, this is a quality we would expect from human summaries, which is yet another reason to focus our analysis on the Curation Corpus.

3.5.3 Implementation Details

We use the code from BERTSUMEXTABS[7] for our experiments. For the decoder, they have their own Transformer implementation while we employ the popular Hugging Face library (Wolf et al., 2020). In our experiments, we control for the possible discrepancy between these two implementations by reporting BERTSUMEXTABS's performance with a Hugging Face Transformer as well.

We use the hyperparameters from BERTSUMEXTABS where not specified otherwise. For our implementation, a grid search found a learning rate of 0.001 for the BERT-initialized encoder and 0.02 for the randomly initialized Transformer(s) to work best. We use a fixed batch size of 3 with gradient accumulation over 5 batches. The hyperparameters for our implementation of BERTSUMEXTABS and our model are the same, and we only tune the hyperparameters of the sentence generator with a grid search.

Our sentence generator is a 2-layer Transformer with 12 heads, a hidden size of 768, an intermediate dimension of 3072 for the feed-forward sublayer, and dropout of 0.1 for attention outputs. We do not apply dropout to the outputs of linear layers.

Curation Corpus. All our models are trained for 40,000 training steps, with a learning rate warmup of 2,500 steps. We did not see an improvement from initializing the encoder with a pretrained extractive model and therefore initialize from BERT weights. We average the results from 5 models, trained with seeds 1 to 5, and also report the standard deviation.

CNN/DailyMail. Our models are trained for 200,000 training steps, with 20,000 warmup steps for the pretrained encoder, and 10,000 warmup steps for the randomly initialized

[5] https://github.com/huggingface/neuralcoref, version 4.0.0
[6] https://spacy.io, version 2.1.0
[7] https://github.com/nlpyang/PreSumm

Model	ROUGE		
	R-1	R-2	R-L
BSEA (Liu and Lapata, 2019b)	42.95 (0.14)	17.67 (0.19)	37.46 (0.21)
BSEA (our implementation)	43.37 (0.37)	17.92 (0.17)	37.73 (0.31)
Sentence planner	**44.40** (0.14)	**18.31** (0.13)	**38.69** (0.10)

Table 3.2: Comparison of generated with reference summaries on Curation Corpus. Mean and std (in brackets) over 5 runs. Best result in bold.

Transformer(s), following Liu and Lapata (2019b). We also use their model checkpoint of BERTSUMEXT to initialize the encoder in all our models.

3.6 Results

We now turn to the evaluation of our method. First, we show the results on Curation Corpus (§ 3.6.1). With attribution techniques (§ 3.6.2) and an ablation study (§ 3.6.3), we uncover how the model uses the sentence generator component. Increasing the number of parameters of BERTSUMEXTABS (BSEA) does not provide the same improvements as our approach (§ 3.6.4). The SUM-QE model-based evaluation uncovers differences between the datasets but not between the models (§ 3.6.5). On the CNN/DailyMail dataset, our model generates more abstractive summaries while retaining high ROUGE scores (§ 3.6.6). Following our automatic evaluations, we validate the results with a human evaluation (§ 3.6.7). Finally, we show and comment on two example summaries (§ 3.6.8).

3.6.1 Results on Curation Corpus

Table 3.2 shows the results of comparing the generated to the reference summaries on the Curation Corpus. The sentence planner substantially improves ROUGE scores compared to BERTSUMEXTABS. The relative difference is between 2.2% and 2.5% for the different ROUGE variants. A noticeable difference also exists between the ROUGE scores of the two base model implementations, which is why we continue reporting the scores for both in the following.

In Table 3.3 we see that the sentence planner's summaries are more abstractive than those of BERTSUMEXTABS, as indicated by the number of novel bigrams. However, there is still a large gap to the reference summaries displayed on the first line. The sentence planner generates substantially more sentences than BERTSUMEXTABS on average, moving it closer to the gold summaries. The mean number of words within those sentences stays close to the reference statistic. [8]

[8]The mean number of sentences and (to a lesser extent) their average length can be influenced by a length penalty hyperparameter α, which is set between 0.6 and 1 (Liu and Lapata, 2019b). BERTSUMEXTABS with no penalty ($\alpha = 1$) produces the same number of sentences and words as the sentence planner with the largest penalty

Model	Sentences		Novel Bigrams	Corefs
	Number	Length		
Gold summaries	*3.46*	*28.0*	*69.22%*	*0.441*
BSEA (Liu and Lapata, 2019b)	2.73 (0.09)	27.3 (0.5)	36.77% (0.94%)	0.267 (0.011)
BSEA (our implementation)	2.76 (0.10)	28.5 (0.8)	37.29% (1.32%)	0.283 (0.026)
Sentence planner	3.15 (0.11)	28.2 (0.5)	**39.29%** (2.00%)	**0.289** (0.023)

Table 3.3: Properties of generated summaries on Curation Corpus. Mean and std (in brackets) over 5 runs.

The mean number of coreferences across sentence boundaries, normalized by the number of sentences, is similar for all models, with the best score achieved by the sentence planner. This number is lower than for the reference summaries but substantially higher than for references and generated summaries from the CNN/DailyMail corpus (see Section 3.6.6).

3.6.2 Attribution to Sentence Representation

A natural question to ask is whether the sentence representation r_{sent} is actually used by the word generator. We therefore compare the attribution of the model predictions to r_{sent} with the attribution to the output of the cross-attention. We use the Integrated Gradients (IG) algorithm (Sundararajan et al., 2017) with respect to these intermediate representations. We choose the zero vector as a baseline r_0, but taking the mean of r_{sent} over the test examples as a baseline provides similar results. We then integrate along the path from r_0 to r_{sent}

$$(r_{\text{sent}} - r_0) \int_{\eta=0}^{1} \frac{\partial F(x, r_0 + \eta(r_{\text{sent}} - r_0))}{\partial r_{\text{sent}}} \tag{3.7}$$

for a given input x. In practice, we discretize the integral and sum over 50 integration steps with linearly spaced η values. The case for the attribution to the cross-attention output is analogous. We report the relative attribution to r_{sent} in Table 3.4. The result is averaged over the first 100 examples in our test set. It shows that the attribution to r_{sent} with the sentence generator alone is about a quarter, while three quarters are attributed to the cross-attention. This is already a substantial amount, considering that the alternative is to directly look at the document. r_{sent}'s attribution share further increases to more than a third with the addition of the guidance loss $\mathscr{L}_{\text{MSE}}$, making r_{sent} even more useful.

While we expect that the sentence representation is mostly used as an outline for the next summary sentence, we are curious to see how much information from the source document is

($\alpha = 0.6$), but a large gap in ROUGE-(1/2/L) remains: (0.7/0.6/0.6). Consistent with Sun et al. (2019), we find that ROUGE scores increase with length and α, but we also find that novel bigrams decrease. In order to not favor one side of the trade-off over the other, we stick with the setting of $\alpha = 0.95$ from Liu and Lapata (2019b) for both models.

Model	IG	Conductance
BSEA	-	-
+ Sentence generator	25.1%	32.3%
+ $\mathscr{L}_{\text{MSE}}$ (= Sentence planner)	36.6%	29.1%

Table 3.4: Attribution study. IG: Attribution of the model predictions to r_{sent} vs. to cross-attention. Conductance: Attribution of the predictions to the article via r_{sent} vs. via cross-attention.

Model	ROUGE		
	R-1	R-2	R-L
BSEA (our implementation)	43.37 (0.37)	17.92 (0.17)	37.73 (0.31)
+ Sentence generator	43.97 (0.30)	18.28 (0.11)	38.32 (0.22)
+ $\mathscr{L}_{\text{MSE}}$ (= Sentence planner)	**44.40** (0.14)	**18.31** (0.13)	**38.69** (0.10)

Table 3.5: Ablation study showing ROUGE scores on Curation Corpus when adding the individual components of our model. Mean and std (in brackets) over 5 runs.

present in r_{sent}. We use the conductance (Dhamdhere et al., 2019) via r_{sent} with respect to the encoder outputs, and compare it to the conductance via the cross-attention. We ignore the encoder's computation as it is the same for both paths. Since it is computationally expensive to compute gradients over every neuron in r_{sent}, we sum over just 5 integration steps and average the result over the first 10 examples of the test set. From Table 3.4, we see that almost a third of the document's information is passed through the sentence representation. The addition of the guidance loss decreases this number, which means that r_{sent} serves more as an outline than an additional condensed representation of the document.

3.6.3 Model Ablation

Table 3.5 shows an ablation study for the two components we introduced in the hierarchical decoder. Both the sentence generator network and the guidance loss provide a steady increase in ROUGE performance as well as a reduction in variance. This demonstrates the efficacy of our additions.

3.6.4 Number of Parameters

To verify that the improved performance of the sentence planner is not just a result of the increased number of parameters, we perform an experiment where we increase the base model's capacity. BERTSUMEXTABS consists of a 12-layer Transformer encoder and a 6-layer decoder. Our model has additional parameters in the 2-layer Transformer that serves as the sentence generator. We therefore increase the BERTSUMEXTABS decoder's parameters such

Model	Params	ROUGE			Novel Bigrams
		R-1	R-2	R-L	
BSEA (Liu and Lapata, 2019b, $L_{\text{dec}} = 6$, $\text{ff}_{\text{dec}} = 2048$)	180M	43.13	17.80	37.63	36.83%
BSEA (our implementation, $L_{\text{dec}} = 6$, $\text{ff}_{\text{dec}} = 2048$)	182M	43.21	17.69	37.54	37.12%
BSEA (our implementation, $L_{\text{dec}} = 6$, $\text{ff}_{\text{dec}} = 3072$)	191M	43.12	17.84	37.53	37.34%
BSEA (our implementation, $L_{\text{dec}} = 8$, $\text{ff}_{\text{dec}} = 2048$)	198M	43.41	17.91	37.79	37.09%
BSEA (our implementation, $L_{\text{dec}} = 8$, $\text{ff}_{\text{dec}} = 3072$)	210M	43.68	18.06	38.06	37.77%
Sentence planner	208M	**44.40**	**18.31**	**38.69**	**39.29%**

Table 3.6: Number of parameters of each model (M = million) together with ROUGE scores and novel bigrams on Curation Corpus.

that the total model sizes match. Specifically, we increase the number of layers L_{dec} and the inner dimension of the feed-forward sublayer ff_{dec}. The comparison is shown in Table 3.6. While increasing the number of parameters improves BERTSUMEXTABS's ROUGE scores, they are still far behind the sentence planner's scores. Similarly, the share of novel bigrams rises a bit with additional parameters. However, it still stays behind the abstractiveness of the sentence planner, showing that the inductive bias of our hierarchical decoder is very effective.

3.6.5 SUM-QE Evaluation

In line with our evaluations, SUM-QE (Xenouleas et al., 2019) evaluates the linguistic quality of a summary. In particular, the two qualities *focus* and *coherence* are desired properties for natural summaries. However, we found the metric to give non-discriminative scores to all summaries (including reference summaries). We therefore only provide the results for completeness.

SUM-QE automatically evaluates summaries with regard to linguistic quality questions asked in the DUC-05/06/07 tasks. We select the qualities regarding focus and coherence, described in Dang (2005) as follows:

> *Q4: Focus.* The summary should have a focus; sentences should only contain information that is related to the rest of the summary.
>
> *Q5: Structure and Coherence.* The summary should be well-structured and well-organized. The summary should not just be a heap of related information but should build from sentence to sentence to a coherent body of information about a topic.

The raters were asked to judge summaries on an integer scale of 1 to 5, which is normalized to $[0,1]$ by the SUM-QE model. It is trained on the raters' judgments and achieves high correlations on a held-out test set. We use the model trained on DUC-05/06 (and evaluated on DUC-07) with the "multi-task-5" setting, producing one output per linguistic quality.

Dataset / Model	Focus	Coherence
CNN/DailyMail	*0.654*	*0.298*
Curation Corpus	*0.848*	*0.563*
BSEA (Liu and Lapata, 2019b)	0.838	0.547
BSEA (our implementation)	0.850	0.563
Sentence planner	0.859	0.562

Table 3.7: Focus and coherence scores of SUM-QE. Models are trained and evaluated on Curation Corpus. Mean over 5 runs.

Model	ROUGE			Sentences		Novel Bigrams	Corefs
	R-1	R-2	R-L	Num	Len		
Gold summaries	-	-	-	*3.88*	*14.08*	*54.33%*	*0.105*
BSEA (Liu and Lapata, 2019b, theirs)	42.16	19.49	39.16	3.33	19.1	7.40%	0.124
BSEA (Liu and Lapata, 2019b, ours)	41.17	18.82	38.27	3.07	18.5	8.14%	0.126
BSEA (our implementation)	41.48	18.86	38.41	2.99	19.6	7.18%	0.104
Sentence planner	41.87	19.37	39.02	3.82	17.8	**10.65%**	**0.132**

Table 3.8: Results on CNN/DailyMail. Best result with our own training underlined.

Table 3.7 holds the SUM-QE scores for the reference summaries of CNN/DailyMail and Curation Corpus. There is an evident difference in scores between the two datasets, with Curation Corpus's summaries being judged more focused and coherent by the model. When comparing the scores of Curation Corpus's reference summaries with the models' scores, there are only minimal differences. The same holds true for a comparison between models. We therefore cannot draw conclusions with respect to the quality of different models from the SUM-QE evaluation.

3.6.6 Results on CNN/DailyMail

For comparison with previous work, we now report the results on the more extractive CNN/DailyMail corpus. Table 3.8 shows the results for BERTSUMEXTABS and the sentence planner. The first line evaluates the model checkpoint that Liu and Lapata (2019b) provide. When we train both the extractive initialization and the abstractive model ourselves with the hyperparameters suggested, we are not quite able to achieve the same results. With our implementation of the decoder, we are able to close the gap in ROUGE scores somewhat. The sentence planner performs best out of the models we trained ourselves. As on the Curation Corpus, it is also much more abstractive than BERTSUMEXTABS. This could well account for the remaining difference in ROUGE scores.

The mean number of generated sentences by the sentence planner is almost identical with the reference summaries, and again a lot larger than for BERTSUMEXTABS. The generated

Quality	BSEA	SP	p-value
Non-redundancy	4.05	**4.08**	0.408
Fluency	3.70	**3.75**	0.343
Structure/coherence	3.68	**3.85**	0.102
Informativeness	3.57	**3.77**	0.069
Abstractiveness	3.45	**3.65**	0.047
Semantic similarity	2.98	**3.18**	0.043

Table 3.9: Mean score for each quality in the human evaluation for BSEA and the sentence planner (SP). Scores range from 1 (worst) to 5 (best). The p-value is determined with a paired bootstrap test.

sentences are also shorter, in line with the references. The number of coreference links across sentence boundaries is similar across models, with the sentence planner producing those links most often. We conclude that even on the more extractive CNN/DailyMail corpus, the sentence planner generates more abstractive and coherent summaries at high ROUGE.

3.6.7 Human Evaluation

We perform a human evaluation to verify the results found by our automatic metrics. We compare outputs of BERTSUMEXTABS (our implementation) with the sentence planner. The annotators are presented with the source article, the reference summary as well as the candidate summaries for both systems. The systems are labeled *1* and *2*, and their order is randomized for each example. For each candidate summary, the annotators then have to select a score from 1 to 5 for six qualities, which are presented with a descriptive question (in brackets). The qualities are non-redundancy (*Is information stated only once?*), fluency (*Is the summary grammatical and good to read?*), structure/coherence (*Do the sentences build on each other?*), informativeness (*Is the important information captured?*), abstractiveness (*How much of the summary is rephrased (instead of copied)?*), and semantic similarity (*How semantically similar is the candidate summary to the gold summary?*).

We randomly draw 20 examples from the Curation Corpus test set. We limit the number of words of source articles to be above 100 and below 700 (includes 70% of examples), to remove extreme examples and keep the workload for annotators reasonable. We divide our 6 annotators, which are all NLP experts, into two groups, who review 10 examples each, resulting in 3 annotations per example, of which we take the mean. The results are reported in Table 3.9. The sentence planner is evaluated favorably compared to BERTSUMEXTABS in all categories. The non-redundancy and fluency categories show a smaller gap. This is expected, as we did not change the word generator, which impacts these categories the most. In the other categories, the sentence planner achieves larger improvements, showing that the introduction of a hierarchical decoder improves the planning capabilities of the model.

To determine the statistical significance of the results, we follow the guidelines in Dror et al.

(2018) and select the non-parametric paired bootstrap test (Efron and Tibshirani, 1994). We find that the two models are not significantly different for the first four categories, while they are for the abstractiveness and semantic similarity categories when selecting a threshold of $p = 0.05$. Additionally, we quantify the inter-annotator reliability with the intraclass correlation coefficient (ICC), according to Shrout and Fleiss (1979). The reliability is moderate with an ICC of 0.56 and a 95% confidence interval of [0.46, 0.65]. Given the moderate annotator agreement and our relatively small sample size of the human evaluation, it is possible that a more extensive (and therefore expensive) human evaluation could show a significant difference in informativeness and structure/coherence.

Finally, we are curious whether the Corefs evaluation can serve as an automatic evaluation of the structure/coherence category. We therefore compute the Pearson ρ for the correlation between the human and the metric's scores. The correlation is weak at 0.098 (p-value: 0.549). Thus there seems to be a mismatch between what the metric measures (discourse coherence by counting the number of coreference links across sentence boundaries) and the open way the question was formulated in the human evaluation (*Do the sentences build on each other?*). Nevertheless, the Corefs metric showed its value by very clearly distinguishing the CNN/DailyMail's summaries from the Curation Corpus's summaries. We therefore leave its optimal use for future work.

3.6.8 Example Summaries

Tables 3.10 and 3.11 show example summaries from the Curation Corpus validation set for the sentence planner and BERTSUMEXTABS (our implementation), alongside the source article and the reference summary.

3.7 Related Work

Several aspects of our method can be found in prior and subsequent work. We mention them in the following and point out the differences to the sentence planner.

Hierarchical attention. Nallapati et al. (2016) use hierarchical attention in the encoder with a word- and a sentence-level RNN. The attention weights at the word level are re-weighted by the sentence-level attention weights. Celikyilmaz et al. (2018) divide the document into paragraphs, which are encoded separately by agents. Each agent performs attention within its paragraph, and the decoder attends to the agents. Gehrmann et al. (2018) first employ a content selector at the word level to decide which words are candidates for copying. They then use a pointer-generator network with just the admissible tokens to generate the summary. Miculicich et al. (2018) use hierarchical attention networks (Yang et al., 2016) to encode the context of previous sentences, which is used to inform the translation of the next word. In contrast to these methods, we employ hierarchy on the decoder side and generate a sentence

Source article Theresa May's plans for a post-Brexit trade deal with the US will be put at risk if she retains EU protections for food and drink such as Champagne and Parma Ham, a senior ally of Donald Trump has warned. The Telegraph has learned that Liam Fox, the International Trade Secretary, has written to David Davis, the Brexit Secretary, warning him not to concede over the issue during negotiations with Brussels. During a recent visit to the US he was told by Paul Ryan, a senior Republican and Speaker of the House of Representatives, that the UK must be able to "diverge" from EU protected status standards to reach a free trade deal. The US produces its own Feta, Parmesan and Champagne and has strongly resisted attempts to ban the sale of American products in the past. Its refusal to compromise on the issue led to the collapse of a major trade deal between the EU and the US. However Michel Barnier, the EU's chief Brexit negotiator, is demanding that Britain must recognise 3,300 protected food and drink products after Brexit. The products are protected under a system of "geographical indications", meaning that they cannot be produced elsewhere.
Reference summary A post-Brexit trade deal with the US may be jeopardised if the UK continues to recognise EU protected status standards for food and drink. The US has resisted calls to adopt protections for products such as feta, Parmesan and Champagne, and would expect the UK to also diverge from them. However, the EU's chief Brexit negotiator, Michel Barnier, says Britain must retain the protections.
Candidate summary (sentence planner) uk prime minister theresa may ' s plans for a post - brexit trade deal with the us will be placed at risk if she retains eu protections for food and drink products such as champagne and parma ham , according to unnamed sources . european trade secretary liam fox has written to david davis , the eu ' s chief brexit negotiator michel barnier , to call for britain to recognise 3 , 300 protected food and drinks products after brexit . the uk produces its own feta , parmesan and champagne imports , and called for the uk to " diverge " from eu protected status standards .
Candidate summary (BERTSUMEXTABS, our implementation) brexit negotiator liam fox has written to david davis , the uk ' s brexit negotiator , calling for britain to recognise 3 , 300 protected food and drink products after brexit . the uk produces its own feta , parmesan and champagne and has strongly opposed attempts to ban the sale of us products in the past . michel barnier , the eu ' s chief brexit negotiator for brexit negotiator michel barnier is calling for the uk to recognise three , 300 products following brexit .

Table 3.10: Hard example from the Curation Corpus. The sentence planner correctly calls "a senior ally of Donald Trump" an "unnamed source". It nicely includes the Speaker of the House's demand to "diverge" from EU standards as a call by the US. It gets confused with the International Trade Secretary, the Brexit Secretary, and the EU's chief Brexit negotiator. It also mistakes the US for the UK when talking about a country producing its own products. BERTSUMEXTABS does these same mistakes, but gets even more confused with the Brexit negotiator. It repeats the call to recognize the protected products by the Brexit negotiator and misses the main point of the article, namely that this issue jeopardizes the post-Brexit trade deal.

Source article FILE PHOTO: U.S. President Donald Trump talks to reporters as he heads to the Marine One helicopter to depart the White House for travel to Florida in Washington, U.S. October 8, 2018. REUTERS/Jonathan Ernst/File PhotoWASHINGTON (Reuters) - President Donald Trump's administration on Thursday for a second time asked the U.S. Supreme Court to put the brakes on a lawsuit filed by young activists who have accused the U.S. government of ignoring the perils of climate change. In the lawsuit, 21 activists, ages 11 to 22, said federal officials violated their rights to due process under the U.S. Constitution by failing to adequately address carbon pollution such as emissions from the burning of fossil fuels. The lawsuit was filed in 2015 against former President Barack Obama and government agencies in a federal court in Eugene, Oregon. Both the Obama and Trump administration have failed in efforts to have the lawsuit thrown out. On July 30, the high court rejected an earlier application by the Trump administration, saying it was premature. Eugene, Oregon-based federal judge Ann Aiken on Monday issued another ruling allowing the case to move forward to trial on Oct. 29 if either the high court or the San Francisco-based 9th U.S. Circuit Court of Appeals do not intervene. Reporting by Lawrence Hurley; Editing by Will Dunham
Reference summary President Trump's administration on 18 October failed in its attempt to halt a climate change lawsuit filed in 2015 by 21 activists, aged between 11 and 22, which accuses the US government of ignoring the risks of climate change. The Obama administration had also attempted unsuccessfully to have the lawsuit thrown out, while the high court rejected a previous application by the Trump administration to have the case dismissed on 30 July. The latest ruling from the Supreme Court means the case can proceed to trial on 29 October if neither the high court or Court of Appeals intervenes.
Candidate summary (sentence planner) president donald trump ' s administration has asked the us supreme court to put the brakes on a lawsuit filed by young activists who have accused the us government of ignoring the perils of climate change . in 2015 , 21 activists claimed federal officials violated their rights to due process under the constitution by failing to address carbon pollution , including emissions from the burning of fossil fuels . on 30 july , the high court rejected an earlier application by the trump administration , saying it was premature .
Candidate summary (BERTSUMEXTABS, our implementation) president donald trump ' s administration for a second time has asked the us supreme court to put the brakes on a lawsuit filed by 21 activists who have accused the us government of ignoring the perils of climate change . the case , which was filed in 2015 against former president barack obama and government agencies in a federal court in oregon , is being brought forward by the high court on 30 july .

Table 3.11: Example from the Curation Corpus. The sentence planner manages to get all facts correct, and summarizes the important content very well by removing phrases such as "on Thursday for a second time", "U.S." in "U.S. Constitution" and "adequately" in "adequately address". It also uses the information that the lawsuit was filed in 2015 from a later sentence to include in the sentence about the origin of the lawsuit. BERTSUMEXTABS also nicely fuses information in its first generated sentence. In the second one, however, it mistakenly believes that the case will be handled on July 30, instead of October 29. It is again a bit shorter on information compared to the sentence planner.

representation for the *next* sentence.

Summary planning. Tan et al. (2017) use word- and sentence-level RNNs in both encoder and decoder. They also predict a next sentence embedding but use a graph model for the importance of the encoded sentences instead of attention. The word-level decoder RNN is conditioned by initializing the first hidden state with the sentence embedding. Perez-Beltrachini et al. (2019) use a CNN word encoder/decoder and an LSTM sentence decoder for multi-document summarization. They predict a next sentence embedding with attention, which they add to the input of each convolutional decoder layer. An auxiliary loss pushes sentence embeddings to be close to LDA topics of summary sentences. Both models do not employ Transformers, and consequently, their conditioning is very different from ours.

Another RNN-based method that can be considered summary planning is presented in See et al. (2017). Their coverage mechanism can be seen as a plan of what *not* to write next. They compute a coverage vector as the sum of attention to the source document in previous decoder time steps, then use it as an input to the attention computation and introduce a penalty if it resembles the next step's attention weights. A similar idea is used for extractive summarization in Narayan et al. (2020), where the already extracted summary is added to the input of (hierarchical) structured transformers to avoid extracting further similar sentences.

Discrete summary planning with entity chains is employed in Narayan et al. (2021). Their main focus is to stay faithful to the source document. To that effect, they split the generation of the summary into two phases. In the first phase, their model generates a chain of entities (named entities, dates, and numbers) that should appear in the summary, i.e. a discrete plan of the entities that should be mentioned. In the second stage, it generates the summary based on the source document and the previously generated entity chain. By restricting the entity chain to only contain entities that appear in the source document, they can increase the faithfulness of the resulting summary to the source document. In Narayan et al. (2022), this idea is extended by sampling diverse entity chains for generation, instead of sampling the summary words directly. The sampled discrete plan can then be used to generate the summary with beam search. Guiding generation with an entity chain is more effective at restricting the output vocabulary (and therefore extrinsic hallucinations) compared to other planning methods, such as plans based on translation, summarization, or question answering (Razumovskaia et al., 2022).

Sentence-level language modeling. Ippolito et al. (2020) pick the most likely continuation from a set of candidate sentences. Their task provides a context of four sentences and requires them to pick a single following sentence. A pretrained BERT model generates a target sentence representation, and the candidate with the highest cosine similarity is selected. Huang et al. (2020) address the task of sentence infilling, where context on both sides of the missing sentence is provided. They learn sentence representations with a denoising autoencoder, predict the representation of the missing sentence with a separate Transformer, and then

use the autoencoder's decoder to generate the missing sentence from that representation. Deutsch and Roth (2019) propose the *summary cloze* task. Given the beginning of a summary, the topic, and the reference document, their model has to continue with a single sentence supported by the reference document. These approaches only predict a single sentence and are given substantial context. In our approach, we generate sentence representations with variable context (or no context for the first summary sentence).

Hua and Wang (2020) receive a prompt and a set of keyphrases, which they position and then fill in the gaps around them. Similarly, Jhamtani and Berg-Kirkpatrick (2020) generate a keyword per target sentence and then generate its left and right context. In contrast to these approaches, our sentence generator outputs a latent representation r_{sent} for the entire sentence, which is used to condition the word generator. We do not tie this representation to specific words.

3.8 Conclusion

We presented the sentence planner, an encoder-decoder model with a hierarchical decoder, consisting of a sentence and a word generator. Our sentence generator computes a plan for the next summary sentence. The word generator is then conditioned on this plan when generating the sentence's words. An additional loss term, which guides the sentence planner towards producing the embedding of the target next sentence, improves the sentence generator's plan. When comparing the sentence planner to a state-of-the-art model without a hierarchical decoder, it generates more abstractive and coherent summaries at higher ROUGE scores.

3.8.1 Higher Abstractiveness

Neither manual inspection of generated summaries nor the analysis of the most frequent novel bigrams of either method showed any systematic explanation as to why the sentence planner generates more abstractive summaries. As stated earlier, higher abstractiveness does not in general increase ROUGE scores, and it was also not part of the training objective (cross-entropy loss with the reference summary). Our assumption therefore is that the conditioning on the next sentence representation introduces an alternative to copying from the input. For next word decisions where the model is not certain, this could then lead the model to generate a novel word instead, which then also changes the words that follow as it completes the sentence.

4 Entailment Representations

Chapter Summary

Contextualized word embeddings are great general-purpose representations. Unfortunately, we do not understand what the individual dimensions of the vectors represent, or how their values relate to those of embeddings for other words. This kind of interpretability is offered by the *entailment vector framework* (Henderson and Popa, 2016). Entailment vectors allow us to determine whether a summary is entailed by the source document, i.e. whether its information is supported by the source document and therefore not hallucinated. In this chapter, we use the entailment vector framework to interpret neural network architectures for contextualized word embeddings and arrive at a set of architectural modifications. We evaluate the resulting representations on the task of recognizing textual entailment (natural language inference). We also assess their usefulness as general-purpose representations on language modeling. Our experiments show no substantial deviations in performance from the base algorithms.

A summary's goal is to present the important information of the source document in a concise and coherent text. A summary should not contain information not present in the source document, even if including it would make the summary itself more fluent and coherent. Nevertheless, such unsupported information is present in some of the summaries that models are trained on, presumably teaching them to hallucinate (see also Chapter 6). We conjecture that with better interpretability of our models, we could decrease the occurrence of hallucinations. In this chapter, we propose methods to learn interpretable contextual representations that are designed to surface the relations between words of the source document and summary.

4.1 Entailment in Summarization

Entailment is useful for abstractive summarization at two levels. At the summary level, a source document should entail its summary (Pasunuru et al., 2017). This includes the relations between entities, possibly across summary sentence boundaries. Additionally, each individual phrase should be supported (occur or be implied) by the source document. A summary constructed in this way avoids hallucinations, which is a challenge for current summarization models (Maynez et al., 2020).

We aim to alleviate this shortcoming by learning contextual entailment vectors according to the framework of Henderson and Popa (2016). These are word vectors where the individual dimensions correspond to information that can be known or unknown. The framework enables computing the entailment relationship between vectors. If we can compute the entailment relationship between the source and summary words, we can mitigate the problem of hallucination, at least at the phrase level. Next, we describe the entailment vector framework, before we present our adjustments to neural network architectures.

4.2 Entailment Vector Framework

In Henderson and Popa (2016), the entailment vector framework is proposed. It is used to interpret the non-contextualized distributed representations of word2vec (Mikolov et al., 2013a) as an approximation of entailment relations between a word and its context words. In their entailment vectors, every vector dimension models if a fact is known or unknown. Each dimension corresponds to a different fact. In the binary version, a 1 in a dimension means that this fact is known about the current word, a 0 that it is unknown. For the probabilistic version, the value is the probability of the fact being known. The entailment relations between known and unknown facts, with their binary equivalents, are defined as follows:

$$\begin{aligned} \text{unknown} &\models \text{unknown} & 0 &\models 0 \\ \text{unknown} &\not\models \text{known} & 0 &\not\models 1 \\ \text{known} &\models \text{unknown} & 1 &\models 0 \\ \text{known} &\models \text{known} & 1 &\models 1 \end{aligned}$$

where $\models$ stands for "entails", and $\not\models$ for "does not entail".

They go on to derive approximate inference operators over entailment constraints. The feature vectors x, y contain the log-odds of the fact at dimension i being known, under a mean-field approximation of the posterior probability. Assuming that feature vector y entails feature vector x, i.e. $y \models x$, they can infer y from x with the *backward inference operator* (termed after the inference direction in the entailment constraint), and x from y with the *forward inference operator*:

$$y \succ x \equiv \sigma(-y) \cdot \log\sigma(-x) \qquad \text{(backward inference)} \tag{4.1}$$

$$y \prec x \equiv \sigma(x) \cdot \log\sigma(y) \qquad \text{(forward inference)} \tag{4.2}$$

In their interpretation of word2vec as entailment feature vectors, there exists a latent vector y for each middle word vector x_m and context word vectors X_c, such that y entails both x_m and each vector in X_c, i.e. the shared facts in the given context should be captured in y. The latent vector y can therefore be inferred from the observed middle and context word vectors with the backward inference operator. We are using the same mechanism in our entailment-based architectures to incorporate update information into contextualized representations.

4.3 Entailment-based Architectures

We aim to learn a representation for a sequence of n discrete tokens (words in our case) $w_i, 1 \leq i \leq n$. We assume that we are given a continuous representation x_i for each such token, where $x_i \in \mathbb{R}^d$ and d is the dimensionality of this representation. In the case of word representations, one typically uses pretrained word vectors from either word2vec (Mikolov et al., 2013a,b), GloVe (Pennington et al., 2014) or fastText (Bojanowski et al., 2017). The common architectures to learn contextual word embeddings are either RNNs, used by ELMo (Peters et al., 2018), or the Transformer, employed by BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019a), among others.

4.3.1 Entailment LSTM

Our Entailment LSTM is an adaptation of the LSTM from Hochreiter and Schmidhuber (1997), described in Section 2.1.1. In the Entailment LSTM, we accumulate evidence in the cell state c_t. It is a function of c_{t-1}, h_{t-1} and x_t (see Equation 2.5). The evidence of the previous time step is inherited from c_{t-1} if it is not invalidated by the forget gate f_t. In the Entailment LSTM, we want to incorporate the new evidence at time step t into the cell state. For the cell state to entail the new evidence, we integrate it with backward inference (Equation 4.1). We replace

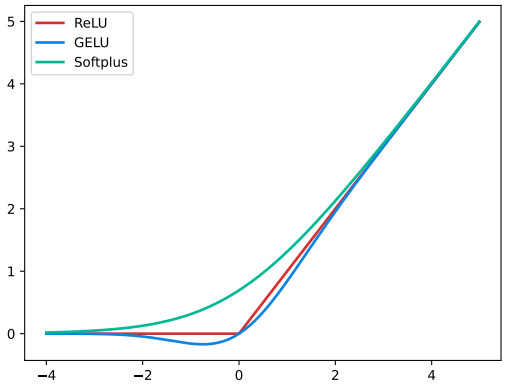


Figure 4.1: Activation functions.

Equation 2.5 with

$$c'_t = f_t \circ c_{t-1} - i_t \circ \log\sigma(-W_c x_t - U_c h_{t-1} - b_c) \quad \text{(backward inference)} \tag{4.3}$$

$$= f_t \circ c_{t-1} + i_t \circ \log(\exp(W_c x_t + U_c h_{t-1} + b_c) + 1) \quad \text{(softplus)} \tag{4.4}$$

The last equation follows from $\log\sigma(-x) = \log(\exp(x)+1)$, which is also known as the *softplus* activation. As can be seen from Figure 4.1, softplus is a smoother version of the ReLU activation function. Considering it an approximation to softplus, we also experiment with using the more common ReLU activation in Equation 4.4.

The backward inference operator's non-negativity will lead to indefinitely accumulating parameter values in c_t. We can apply layer normalization to guarantee a normally distributed input to subsequent computations.

Layer normalization for LSTM. Multiple versions of layer normalization (LN) exist for the LSTM. We use an implementation with normalization on the computation of the weights (inside the sigmoid) and the cell state (on the sum):

$$f_t = \sigma(\text{LN}(W_f x_t + U_f h_{t-1} + b_f)) \tag{4.5}$$

$$i_t = \sigma(\text{LN}(W_i x_t + U_i h_{t-1} + b_i)) \tag{4.6}$$

$$o_t = \sigma(\text{LN}(W_o x_t + U_o h_{t-1} + b_o)) \tag{4.7}$$

$$c_t = \text{LN}(f_t \circ c_{t-1} + i_t \circ \tanh(W_c x_t + U_c h_{t-1} + b_c)) \tag{4.8}$$

$$h_t = o_t \circ \tanh(c_t) \tag{4.9}$$

In the Entailment LSTM, we want to prevent an explosion of the cell state weights due to adding a non-negative transformation of the input and previous hidden state at every time

step. We change Equation 4.4 by adding a normalization step either inside or around the softplus activation:

$$c'_t = \begin{cases} f_t \circ c_{t-1} + i_t \circ \log(\exp(\mathrm{LN}(W_c x_t + U_c h_{t-1} + b_c)) + 1) & \text{(c0)} \\ f_t \circ c_{t-1} + i_t \circ \mathrm{LN}(\log(\exp(W_c x_t + U_c h_{t-1} + b_c) + 1)) & \text{(c1)} \\ f_t \circ c_{t-1} + \mathrm{LN}(i_t \circ \log(\exp(W_c x_t + U_c h_{t-1} + b_c) + 1)) & \text{(c2)} \\ \mathrm{LN}(f_t \circ c_{t-1} + i_t \circ \log(\exp(W_c x_t + U_c h_{t-1} + b_c) + 1)) & \text{(c3)} \end{cases} \tag{4.10}$$

With c0, the evidence vector c'_t will only ever add positive values, and its growth is only limited by the forget and input gates. This induces a higher risk of unstable training through exploding gradients. During training, we therefore clip the gradient norm. c1 negates the non-negative transform of the softplus and centers the update term at 0 with unit variance. c2 includes the weighting by the input gate, and c3 uses layer normalization as the LSTM (see Equation 4.8). Both c2 and c3 performed consistently worse than c0 and c1, so we excluded them from our detailed experiments.

For normalization of the hidden state, we replace Equation 2.6 with one of:

$$h'_t = \begin{cases} o_t \circ \tanh(\mathrm{LN}(c_t)) & \text{(h0)} \\ \mathrm{LN}(o_t \circ \tanh(c_t)) & \text{(h1)} \end{cases} \tag{4.11}$$

The h0 normalization keeps the input to the tanh activation standardized and decouples it from the weight norm of the cell state. The h1 normalization which includes the entire expression performed worse in initial experiments.

Log-odds bias. In the original LSTM, the forget gate removes information from the cell state c_t by setting it to zero. In the entailment interpretation, c'_t holds the log-odds of information being known or unknown. The equivalent operation performed by the forget gate would therefore set the log-odds to unknown, which corresponds to a large negative scalar v_{neg}. We modify Equation 4.4 accordingly:

$$c'_t = f_t \circ c_{t-1} + i_t \circ \log(\exp(W_c x_t + U_c h_{t-1} + b_c) + 1) + (1 - f_t)\, v_{\text{neg}} \tag{4.12}$$

If we choose $v_{\text{neg}} = -6$, we retain a non-zero gradient while still keeping a close-to-zero probability when taking its sigmoid.

Outputting the cell state. For the application of natural language inference, it might make more sense to output the cell state instead of the hidden state, considering that the cell state holds our accumulated evidence for information being known or unknown. Thus, we output the cell state instead of the hidden state, add layer normalization to make the output independent of the weight norm of c'_t, and scale the standard deviation with $|v_{\text{neg}}|$ to arrive at

our log-odds interpretation of $c_{\text{out},t}$:

$$c_{\text{out},t} = \text{LN}(c'_t) * |v_{\text{neg}}| \tag{4.13}$$

The hidden state h'_t is still used in the computation of the update Uh'_t in step $t+1$ as before. In a stacked Entailment LSTM, the output of the first layer $c^{(1)}_{\text{out},t}$ then becomes the input to the second layer $x^{(2)}_t$, instead of the hidden state $h'^{(1)}_t$ as in the previous formulation.

4.3.2 Entailment GRU

In the GRU architecture (see Section 2.1.1), we have no dedicated memory like the cell state c_t. This means that our only candidate for evidence accumulation is the hidden state h_t. We again modify the incorporation of new evidence (see Equation 2.9), and replace the tanh activation with the backward inference operator to get:

$$\tilde{h}'_t = \text{softplus}(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h) \tag{4.14}$$

As in the Entailment LSTM, we experiment with layer normalization on the hidden state update:

$$\tilde{h}'_t = \begin{cases} \text{softplus}(\text{LN}(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h)) & \text{(h0)} \\ \text{LN}(\text{softplus}(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h)) & \text{(h1)} \end{cases} \tag{4.15}$$

or the computation of the new hidden state:

$$h_t = \text{LN}((1 - z_t) \circ h_{t-1} + z_t \circ \tilde{h}_t) \tag{4.16}$$

The Entailment GRU did not achieve similar performance as the Entailment LSTM in our initial experiments. Coupled with the fact that there is no dedicated memory to accumulate evidence, we did not pursue the Entailment GRU further.

4.3.3 Entailment Transformer

We consider solely the Transformer encoder throughout this section. In the Transformer architecture (Section 2.1.4), information gets constantly accumulated over layers. New evidence gets added in the attention sublayer (Equation 2.19). The attention function extracts the relevant information about the other tokens in the sequence, and it is unified with the previously accumulated evidence. To achieve this unification, we apply the backward inference operator to the output of the multi-head attention function:

$$\text{MHA}(x) = f_{\text{merge}}(\text{softplus}(\text{MHA}_i(q_i, k_i, v_i)))W + b \quad \text{, with } 1 \leq i \leq H \tag{4.17}$$

The partitioning of the input dimension onto the attention heads poses a challenge for the

entailment interpretation. As seen in Equation 2.23, the individual heads only read and write their own segment and do not interact except in the weight matrix W in Equation 2.24 and in the feed-forward step (Equation 2.20). In the Transformer paper, the heads are shown to learn different semantic and syntactic functions, e.g. coreference resolution (Vaswani et al., 2017). For independent functions, it makes sense to divide the model dimensions onto the heads. In the entailment interpretation, however, a specific piece of information being known or unknown interacts with the rest of the evidence. We therefore adapt the divide and merge functions. We experiment with f_{divide} either copying its input H times or projecting it into $\mathbb{R}^{dH}$ and reshaping it into $\mathbb{R}^{d\times H}$ to divide it onto the heads. As a result, the input dimensionality for each head MHA_i is d instead of d/H, and the dimension of the input to f_{merge} is $\mathbb{R}^{d\times H}$. The merge function f_{merge} then has to aggregate its input to arrive at an output dimension of d again, so we apply summation or max-pooling:

$$\text{MHA}(x) = \begin{cases} \sum_{i=1}^{H} \omega(\text{MHA}_i(q_i, k_i, v_i))W + b & \text{(sum)} \\ \max\limits_{1\le i\le H} \omega(\text{MHA}_i(q_i, k_i, v_i))W + b & \text{(max)} \end{cases} \tag{4.18}$$

where ω is either an activation, as in Equation 4.17, or the identity, as in the original Transformer formulation (Equation 2.24).

Inverse softplus pooling. We also investigate inverse softplus pooling (ISP) to unify evidence. It is defined as:

$$\text{isp-pool}(x) = \text{softplus}^{-1}\left(\sum_i \text{softplus}(x_i)\right) \tag{4.19}$$

It exhibits the properties of reflexivity:

$$\text{isp-pool}(x_i) = \text{softplus}^{-1}(\text{softplus}(x_i)) = x_i \tag{4.20}$$

associativity:

$$\begin{aligned} \text{isp-pool}(x_i, x_j, x_h) &= \text{softplus}^{-1}(\text{softplus}(x_i) + (\text{softplus}(x_j) + \text{softplus}(x_h))) && (4.21)\\ &= \text{softplus}^{-1}((\text{softplus}(x_i) + \text{softplus}(x_j)) + \text{softplus}(x_h)) && (4.22) \end{aligned}$$

and commutativity:

$$\begin{aligned} \text{isp-pool}(x_i, x_j) &= \text{softplus}^{-1}(\text{softplus}(x_i) + \text{softplus}(x_j)) && (4.23)\\ &= \text{softplus}^{-1}(\text{softplus}(x_j) + \text{softplus}(x_i)) = \text{isp-pool}(x_j, x_i) && (4.24) \end{aligned}$$

The last two properties follow from the associativity and commutativity of the sum. Pooling operations such as the sum or maximum exhibit the same properties. We can therefore replace them with ISP. In combination with projecting the input of the attention heads and summing

their output mentioned above, we can change the aggregation in Equation 4.18 to:

$$\text{MHA}(x) = \text{isp-pool}(\text{MHA}_i(q_i, k_i, v_i))W + b \quad \text{, with } 1 \leq i \leq H \tag{4.25}$$

A further opportunity to replace a pooling operation is in the computation of the fixed-size sentence representation needed for e.g. the NLI task, which we do by max-pooling the Transformer encoder outputs by default. Applying ISP gives us the accumulated evidence from all output positions. Moreover, the ResNet variant discussed below sums the contributions of individual blocks. We can replace this sum with ISP, to add up the evidence from the individual computations. Finally, we use ISP in the attention computation itself by computing:

$$\text{MHA}_i(q_i, k_i, v_i) = \text{softplus}^{-1}\left(\text{softmax}\left(\frac{q_i k_i}{\sqrt{d_k}}\right)\text{softplus}(v_i)\right) \tag{4.26}$$

This behaves like normal attention when the attention weights are skewed or the values are similar, but is bounded by max pooling if the attention weights are uniform and the values are different. As a result, we get a smoother version of attention, where an attention head can either select an individual value vector or unify the evidence from multiple vectors.

Entailment attention. Instead of computing the dot-product attention, we interpret the queries and keys as entailment vectors and compute their alignment either with the backward inference operator (Equation 4.1) or the forward inference operator (Equation 4.2). We replace the dot-product attention in Equation 2.23 as follows:

$$\text{MHA}_i(q_i, k_i, v_i) = \text{softmax}\left(\frac{\sigma(-k_i)\log\sigma(-q_i)}{\sqrt{d_k}}\right)v_i \qquad \text{(backward)} \tag{4.27}$$

$$\text{MHA}_i(q_i, k_i, v_i) = \text{softmax}\left(\frac{\sigma(q_i)\log\sigma(k_i)}{\sqrt{d_k}}\right)v_i \qquad \text{(forward)} \tag{4.28}$$

Modified Base Model

We further try some modifications of the Transformer base model with respect to the softmax, residual connections, and key and value projections.

Non-selective softmax. The softmax is a way to create probabilities from logits, where more probability mass is concentrated on the high logits than for other normalization schemes, such as a simple division by the sum of the logits. In attention, it is used for computing the attention weights α. Since the α_i have to sum to 1, this means that a model has to put its attention somewhere – even if the logits for the current training sample are low everywhere. Since the softmax promotes a concentrated probability mass, it will still force attention to select certain inputs. To make it easier for the softmax to be non-selective, i.e. to output a more uniform attention weight distribution for cases where all logits are low, we add a constant $c = 1$

to the normalization term in the denominator.

$$\text{ns-softmax}(x)_i = \frac{\exp(x_i)}{\left(\sum_j \exp(x_j)\right) + 1} \tag{4.29}$$

With this change, the attention weights no longer sum to 1, and the remainder is interpreted as paying attention to *no specific input* for the current training sample.

ResNet Transformer. The Sparse Transformer (Child et al., 2019) uses an architecture similar to residual networks (ResNet, He et al., 2016), where residual connections bypass the entire attention and feed-forward sublayers. Equations 2.19 and 2.20 change as follows:

$$a_t^l = h_t^{l-1} + \text{MHA}(\text{LN}(h^{l-1})) \tag{4.30}$$

$$h_t^l = a_t^l + \text{FFN}(\text{LN}(a_t^l)) \tag{4.31}$$

We can see that additionally, the layer normalization moves inside the multi-head attention and the feed-forward network.

Tying keys and values. In the base Transformer, keys and values go through different projections W^k and W^v (see Equation 2.21). The reasoning is that the keys are responsible for selecting the right memory positions by aligning with the queries, while the values hold the information. However, if we assume the queries to be meaningful in the output space, it makes sense that the values themselves align with the queries. We therefore experiment with tying the keys and values by projecting with a single matrix W^{kv}:

$$q, k, v = x_t W^q, x_t W^{kv}, x_t W^{kv} \tag{4.32}$$

4.4 Experiments

Naturally, we evaluate our entailment-based architecture modifications on the task of natural language inference. We also compare our changes with the base algorithms on language modeling, to get a sense of the generalization abilities of the resulting representations. We start this section by describing our training protocol.

4.4.1 Datasets

For natural language inference, we use the SNLI corpus, and for language modeling we use WikiText-2.

SNLI. The Stanford Natural Language Inference (SNLI) corpus is a large dataset for the task of natural language inference (Bowman et al., 2015). It consists of 570k premise-hypothesis pairs.

The premises are image captions describing a scene, and the hypotheses were generated by crowd workers to either be entailed by, neutral to, or contradict a given premise. Later studies found annotation artifacts due to the construction process, where certain words are much more common for one of the labels (Gururangan et al., 2018; Poliak et al., 2018). Nevertheless, this corpus is widely used due to its large size. The dataset can be downloaded from the project homepage.[1]

WikiText-2. We use the WikiText-2 corpus (Merity et al., 2017) to test if our entailment representations are generally useful on the task of language modeling. WikiText-2 (and its larger version WikiText-103) are a collection of featured Wikipedia articles, which means that they have been carefully edited and should contain a minimal amount of grammatical mistakes. This quality guarantee makes them a good target for language model training. The WikiText-2 training set consists of 2.1 million tokens from 600 articles. The validation and test set both consist of 60 articles, with a token count of 218k and 246k, respectively. Both the smaller WikiText-2 and the larger WikiText-103 are available from the dataset website.[2]

4.4.2 Training Details

We try to spend the same amount of computation for tuning our baseline models and our proposed variations. Our tuning starts with finding an optimizer that performs well for a constant learning rate. This turns out to be SGD for NLI and Adam for language modeling. We did not get Adam to converge on the NLI task. Once we have decided on an optimizer, we first find the best learning rate and associated schedule. We found striking differences when it comes to learning rate schedules and will elaborate further below. We then fix the learning rate and the schedule and tune regularization, which includes weight decay and dropout. Finally, we run the best setting for multiple runs with different random seeds and report the mean and standard deviation.

Learning Rate Schedule

We experimented with constant learning rates, linearly decreasing learning rates, cosine annealing, and validation-based decaying of the learning rate. In the end, we found that the *1cycle* learning rate schedule (Smith, 2018) gave the best results on both tasks. The 1cycle schedule has three phases: A *warmup* phase where the learning rate is annealed from $\text{lr}_{\text{max}}/10$ to lr_{max}, an *annealing* phase, where it is annealed back down to $\text{lr}_{\text{max}}/10$, and finally a *cooldown* phase, where it is decreased further to $\text{lr}_{\text{max}} * 1\text{e}{-3}$. The annealing can either be linear or follow the cosine curve, and we found linear to perform better. The fraction of training spent in each phase is subject to tuning, and we observed large differences between models and tasks (see below). The 1cycle schedule, where the learning rate depends on the training step,

[1] https://nlp.stanford.edu/projects/snli/

[2] https://www.salesforce.com/products/einstein/ai-research/the-wikitext-dependency-language-modeling-dataset/

substantially outperforms an adaptive schedule that decreases the learning rate when no further improvement in the validation metric is observed. For language modeling with the LSTM, for example, perplexity goes down from 108.9 to 102.1 (see Table 4.3a).

1cycle-4phases. We observed that the LSTM's validation perplexity has not yet fully converged when the (typically short) cooldown phase ends. We therefore introduce a fourth phase we call *burn-in*, where we further decrease the learning rate to a predetermined $\text{lr}_{\text{min}} = \text{lr}_{\text{max}} * 1\text{e}{-6}$, below which we do not expect the gradient updates to have any substantial effect on the model weights anymore. In language modeling, this schedule brings further minor gains and improves the LSTM's perplexity from 102.1 to 101.7. On the NLI task, the fourth phase did not bring further improvements, potentially due to the smaller number of training steps.

Fraction of training spent in each phase. While the original paper suggests the 3 phases to be distributed at about 44% warmup, 44% annealing, and 12% cooldown (Smith, 2018), we found the best values for these hyperparameters to differ strongly for our models and tasks. For language modeling, both LSTM and Transformer spend most of their training in the annealing phase (60–70%), while for NLI, spending 80% of training in the warmup phase achieved the best results for all Transformer models and nearly all LSTM variants. Due to the combinatorial increase in hyperparameter trials with these additional hyperparameters, we independently tune the division into phases from the other hyperparameters.

4.4.3 Natural Language Inference

We train individual sentence representations for the premise and the hypothesis on the SNLI training set for 20 epochs. We train two different classifiers. One is based on entailment scores (Karimi Mahabadi et al., 2019) that force the encoder to learn the entailment relationships without a subsequent non-linear classifier and use interpretable scores for the classification. Our second classifier is a standard multi-layer perceptron (MLP) with a softmax output and heuristic matching features as input, as used by InferSent (Conneau et al., 2017). The matching features are a concatenation of the individual sentence representations u and v, their element-wise product as well as the absolute element-wise difference: $[u; v; u \circ v; |u - v|]$.

We train our own BiLSTM and Transformer encoder baselines and use the results for InferSent reported in their paper (Conneau et al., 2017). BiELSTM uses the softplus activation, c0 and h0 layer normalization, $v_{\text{neg}} = -6$, and outputs the hidden state h. LSTM variants use 1 or 2 layers, the Transformer uses $L = 6$ layers and $H = 8$ attention heads. We learn 512-dimensional encoder representations, which results in 1024-dimensional concatenated embeddings for bidirectional methods. We use the SGD optimizer, the 1cycle schedule, and a maximum learning rate of 0.5 for the Bi(E)LSTM models, and either 0.5 or 1 for the Transformer.

Method	mean	std
BiLSTM	83.12%	0.16%
BiLSTM (2 layers)	83.51%	0.27%
BiELSTM	82.82%	0.22%
BiELSTM (2 layers)	83.26%	0.10%

(a) LSTM variants.

Method	mean	std
Transformer	84.44%	0.24%
Project-sum attention heads	84.49%	0.30%
ISP ResNet	83.99%	0.23%
ISP attention computation	84.19%	0.16%
forward entailment att.	84.49%	0.14%
backward entailment att.	84.59%	0.33%
non-selective softmax	84.44%	0.26%
ResNet	84.27%	0.15%
tied keys and values	84.47%	0.11%

(b) Transformer variants.

Table 4.1: Test accuracy of encoder architectures with entailment scores on natural language inference, showing mean and standard deviation over 5 runs.

4.4.4 Language Modeling

We also evaluate our entailment-based architectures on language modeling to test their use as a general-purpose encoder. We use the data splits provided with the WikiText-2 dataset, and learn task-specific word embeddings from scratch. We train with two different settings. First, we compare our various model architectures by training to convergence with 50 epochs. The embedding size as well as the hidden dimensionality is set to 200. We use the Adam optimizer, a maximum learning rate of 0.01 and regularize with weight decay of 1e−6 and dropout of 0.2 on the input embedding, the LSTM hidden states and the Transformer attention weights and sublayer outputs. We then select the best-performing model to compare with AWD-LSTM (Merity et al., 2018), a heavily regularized vanilla LSTM. This training runs for 750 epochs, with much slower convergence due to stronger regularization, but at the same time much better generalization performance. It uses 400-dimensional embeddings and 1150-dimensional hidden states.

4.5 Results

We now present the results for our best-performing models on NLI and language modeling. In the following Section 4.6, we provide a detailed ablation of our architecture modifications.

4.5.1 Natural Language Inference

We measure mean accuracy on SNLI over 5 runs of the same model with different random seeds and present the results for the entailment scores classifier in Table 4.1, and for the MLP with heuristic matching features in Table 4.2.

For LSTM-based encoders trained with entailment scores, we see from Table 4.1a that adding an additional layer improves accuracy by about 0.4%. The results for our bidirectional En-

Method	mean	std
InferSent	84.82%	–
BiLSTM	85.05%	0.20%
BiLSTM (2 layers)	85.32%	0.23%
Transformer	86.28%	0.17%
BiELSTM	84.53%	0.26%
BiELSTM (2 layers)	85.34%	0.18%

Table 4.2: Test accuracy of encoder architectures with heuristic matching features on natural language inference. Mean and standard deviation over 5 runs.

Method	Valid. Perplexity
GRU (2 layers)	121.5
Transformer (6 layers)	117.8
LSTM (2 layers, adaptive)	108.9
LSTM (2 layers, 1cycle)	102.1
LSTM (2 layers, 1cycle-4phases)	101.7
ELSTM (2 layers)	104.2
ELSTM (3 layers)	99.62

(a) 50 epochs, less regularization.

Method	Test Perplexity
AWD-LSTM (3 layers)	65.37
ELSTM (3 layers)	65.52

(b) 750 epochs, heavy regularization.

Table 4.3: Perplexity (lower is better) on WikiText-2 language modeling.

tailment LSTM (BiELSTM) are slightly behind the LSTM baseline's results. All LSTM models are outperformed by the Transformer variants shown in Table 4.1b, which all achieve very similar accuracy. This is surprising to see, as our initial architecture experiments (detailed in Section 4.6.3) hinted at a larger difference between the models. A possible explanation could be that with the right hyperparameters, the Transformer model is powerful enough to adapt to our changes and converge to the base model's solution. In our experiments, we especially found the initial warm-up of the learning rate to be crucial for good results, as is also reported in the original paper (Vaswani et al., 2017).

The evaluation with heuristic matching features and an MLP classifier mirrors the results for the entailment scores, albeit with an absolute improvement of almost 2 accuracy points (see Table 4.2). The best result is achieved by a base Transformer with 86.28% mean accuracy.

4.5.2 Language Modeling

We show perplexity, i.e. the surprisal of a model when shown a text sequence not seen during training, in Table 4.3. Lower perplexity is better. From Table 4.3a, we see that the LSTM clearly outperforms the Transformer encoder and the GRU. Further improvements can be achieved with the best learning rate schedule. The ELSTM does not reach the perplexity of an LSTM when both models use 2 layers and needs another layer to improve on LSTM's score. The same

Model	Num. parameters
(E)LSTM	1.7M
Bi(E)LSTM	3.3M
Bi(E)LSTM (2 layers)	9.6M
Transformer (6 layers)	19.1M
Transformer (6 layers, project head inputs)	31.7M

Table 4.4: Number of parameters per model (M stands for million).

holds true in the larger scale comparison with the AWD-LSTM (Table 4.3b). However, since this evaluation runs for multiple days, the hyperparameters from Merity et al. (2018) were used for both the AWD-LSTM and the ELSTM, which might put the ELSTM at a disadvantage if they were carefully tuned for the AWD-LSTM.

4.6 Architecture Ablation

The standard number of layers and therefore parameters used in the LSTM and the Transformer differs quite a bit (see Section 4.6.1). The training times for both are comparable, however, as the Transformer can parallelize computation on the entire sequence, while it is sequential in the LSTM. We now compare model parameters and then perform a detailed ablation of architecture choices in both the Entailment LSTM and the Entailment Transformer.

4.6.1 Number of Model Parameters

We list the number of model parameters in Table 4.4, with the unidirectional single-layer LSTM as a reference. Among its parameters, around 0.6M are accounted for by the input to hidden matrix W, while approximately 1M parameters come from the hidden to hidden matrix U (see Section 2.1.1). The BiLSTM doubles this number exactly. When going to two layers, the input-to-hidden weight matrix of the second layer $W^{(2)}$ has 2.1M parameters, as it takes as input the concatenated output of the first layer of both directions. The ELSTM variants have the same number of parameters. We omit to learn adaptive bias and gain parameters for layer normalization as proposed in its original formulation (Ba et al., 2016).

The Transformer with 6 layers has many more parameters. These again come from the weight matrices. The attention sublayer has approximately 1M parameters, dominated by $W^{\{q,k,v\}}$ and W, while the feed-forward sublayer has around 2.1M parameters in W_1 and W_2 (see Section 2.1.4). The Transformer with f_{divide} as projection adds another 2.1M parameters per layer, due to projecting the input to $\mathbb{R}^{dH}$ (Section 4.3.3).

Model	Output	Layers	Activation	Norm	v_{neg}	SNLI	Diff.
BiLSTM	h	1	tanh	none	0	83.12%	
		2				83.71%	
BiELSTM	h	1	softplus	c0	0	82.75%	-0.37%
					-6	83.02%	-0.10%
				c1	0	82.90%	-0.22%
					-6	82.99%	-0.13%
			relu	c0	0	82.23%	-0.89%
					-6	82.77%	-0.35%
				c1	0	81.15%	-1.97%
					-6	81.05%	-2.07%
			tanh	c0	0	81.08%	-2.04%
					-6	82.55%	-0.57%
				c1	0	83.18%	0.06%
					-6	82.69%	-0.43%
		2	softplus	c0	0	83.54%	-0.17%
					-6	83.59%	-0.12%
				c1	0	83.34%	-0.37%
					-6	83.62%	-0.09%
			relu	c0	0	83.26%	-0.45%
					-6	83.41%	-0.30%
				c1	0	83.58%	-0.13%
					-6	83.80%	0.09%
			tanh	c0	0	82.65%	-1.06%
					-6	82.42%	-1.29%
				c1	0	82.91%	-0.80%
					-6	82.98%	-0.73%
	c	1	softplus	c0	0	82.83%	-0.29%
					-6	79.37%	-3.75%
				c1	0	82.95%	-0.17%
					-6	80.16%	-2.96%
		2	softplus	c0	0	82.71%	-1.00%
					-6	79.36%	-4.35%
				c1	0	83.67%	-0.04%
					-6	78.10%	-5.61%

Table 4.5: Comparison of Entailment LSTM architecture choices. Empty fields copy the setting from the line above. Differences in the last column are with respect to the baseline with the same number of layers.

Model	Act. FFN	Act. Attn	SNLI	Diff.
Base	gelu	none	82.62%	
		relu	81.32%	-1.30%
		softplus	81.93%	-0.69%
		tanh	76.91%	-5.71%
	relu	none	82.66%	+0.04%
		relu	81.60%	-1.02%
		softplus	81.75%	-0.87%
		tanh	76.20%	-6.42%
	tanh	none	80.93%	-1.69%
		relu	78.30%	-4.32%
		softplus	78.99%	-3.63%
		tanh	71.66%	-10.96%
ns-softmax	gelu	none	82.26%	-0.36%
	relu		82.98%	+0.36%
	tanh		80.85%	-1.77%
ResNet T	gelu		82.94%	+0.32%
	relu		83.22%	+0.60%
	tanh		82.69%	+0.07%
tied kv	gelu		82.94%	+0.32%
	relu		82.65%	+0.03%
	tanh		80.62%	-2.00%

(a) Comparison of activation functions, f_{divide} = split, f_{merge} = concat.

f_{divide}	f_{merge}	Act. Attn	SNLI	Diff.
split	concat	none	82.62%	
copy	sum	none	41.71%	-40.91%
		relu	41.81%	-40.81%
		softplus	40.29%	-42.33%
		tanh	41.79%	-40.83%
project	sum	none	83.68%	+1.06%
		relu	82.07%	-0.55%
		softplus	40.46%	-42.16%
		tanh	83.68%	+1.06%
	max	none	82.36%	-0.26%
		relu	82.83%	+0.21%
		softplus	79.84%	-2.78%
		tanh	82.02%	-0.60%

(b) Comparison of f_{divide} and f_{merge}, Model = Base, Act. FFN = gelu.

Table 4.6: Architecture comparison for the Entailment Transformer, for activation functions and split and merge functions. Empty fields copy the setting from the line above.

4.6.2 LSTM Architecture

We perform an ablation of the architectural choices in the Entailment LSTM on the NLI task. All models are trained with a constant learning rate of 0.5. We experiment with the modifications presented in Section 4.3.1 and compute the difference to a one- or two-layer BiLSTM. For outputting the hidden state h, we additionally use the ReLU activation function as an approximation to the softplus, and the tanh for controlling the effect of applying layer normalization and log-odds bias to a standard LSTM. The results are shown in Table 4.5.

Most modifications remain very close to BiLSTM's performance, but some runs fall short in a non-conclusive manner (e.g. c1 normalization for a 1-layer ReLU BiELSTM performs badly, but with 2 layers it performs best of all activation-normalization combinations). The softplus activation has the lowest variance of the activation functions. It benefits consistently from the log-odds bias, while the normalizations perform similarly. Unexpectedly, this behavior is reversed when outputting the cell state c instead of the hidden state h. The results for the ReLU activation are close to the softplus's, albeit with higher variance. Neither normalization nor log-odds bias consistently improves the results of the BiLSTM (i.e. the BiELSTM with the tanh activation).

4.6.3 Transformer Architecture

We again use a constant learning rate of 0.5 for the Transformer ablation. We use 4 layers and no dropout or weight decay. When searching for the best hyperparameters for learning rate and regularization, we expect improvements in the range of 1 to 2 accuracy points, as we observe for the base model (see Table 4.1b).

In Table 4.6a, we see that using an activation function for the output of the attention as defined in Equation 4.17 does not improve results. Choosing either a GELU or ReLU activation on the feed-forward network has no consistent effect. The tanh activation, however, substantially decreases performance. The non-selective softmax, the ResNet Transformer, and tying the keys and values improve over the base model in some configurations. However, after tuning the hyperparameters, these advantages disappear and the test accuracy is remarkably close to that of the base model.

In Table 4.6b, we compare different functions for dividing the inputs and merging the outputs of multi-head attention. Using the copy strategy for f_{divide}, we were not able to train models that could consistently decrease the training error. Projecting the inputs up to a higher dimension works much better, and even improves upon the base model by a margin. Unfortunately, the same effect we observed for activation functions repeats here as well; the test accuracy of the tuned model loses the advantage of the initial experiments. Using no attention activation beats the tanh activation. The softplus attention activation seems to be unstable in conjunction with the sum as f_{merge}. Using max-pooling for merging the heads' outputs resolves this, but achieves lower accuracy than projecting and summing.

ISP applied to	SNLI	Diff.
none	82.62%	
attention heads	41.71%	-40.91%
outputs (instead of max-pooling)	80.12%	-2.50%
ResNet blocks	83.32%	+0.70%
attention computation	83.49%	+0.87%

(a) Comparison of variants of inverse softplus pooling, Model = Base, Act. FFN = gelu, f_{divide} = split, f_{merge} = concat.

attention function	SNLI	Diff.
dot-product	82.62%	
backward inference	82.42%	-0.20%
forward inference	82.32%	-0.30%

(b) Comparison of attention functions, Model = Base, Act. FFN = gelu, f_{divide} = split, f_{merge} = concat.

Table 4.7: Architecture comparison for the Entailment Transformer, for pooling and attention functions. Empty fields copy the setting from the line above.

We compare different applications of inverse softplus pooling in Table 4.7a. Using it to merge the output of the attention heads cannot be trained to reach training error much below the random baseline. The other variants do much better. Applying isp-pool to the final outputs to produce a fixed-size sentence representation is 2.5% accuracy points behind max pooling. Applying it to the contributions of individual ResNet blocks performs better than the base model. The best approach in this experiment applies inverse softplus pooling to the attention computation (Equation 4.26).

In Table 4.7b, we compare the dot-product attention with the backward and forward inference operators as a way to compute the alignment of queries and keys. We observe that the performance is very similar, with the dot-product attention achieving the best score on the validation set. Both variants remain competitive after tuning hyperparameters (see Table 4.1b).

4.7 Related Work

To the best of our knowledge, this work is the first to introduce an inductive bias for entailment into the LSTM or Transformer architectures. There is, however, a wealth of research on general modifications of these architectures.

LSTM architecture. The proposal of peephole connections (Gers and Schmidhuber, 2000) lets the gate computation take into account the previous cell state c_{t-1}, additionally to the previous hidden state h_{t-1}. The most widely adapted modification is the GRU (Cho et al., 2014b), as it can achieve similar performance with a simpler architecture and a smaller number

of parameters (see Section 2.1.1). Nested LSTMs (Moniz and Krueger, 2017) introduce an inner LSTM cell that replaces the cell state in order to gain more memory capacity, especially for storing longer-term dependencies. The rotational unit of memory (Dangovski et al., 2019) pursues the same objective by replacing the cell state with rotational associative memory. Ordered neurons LSTM (Shen et al., 2019) use an inductive bias towards tree structures by mapping the entries of a flat memory vector to a tree and updating all nodes in the subtree whenever the parent node gets updated. Finally, adaptive computation time (ACT, Graves, 2016) uses a variable number of steps to process each input, instead of processing a single input token at each time step.

Transformer architecture. The Transformer (Vaswani et al., 2017) was introduced more recently than the LSTM but has nonetheless seen rapid adoption for NLP tasks. The Weighted Transformer (Ahmed et al., 2017) modifies the multi-head attention by feeding the heads separately into the feedforward layer and learning how to combine them, in an attempt to learn decorrelated heads. The Universal Transformer (Dehghani et al., 2019) ties the weights of all Transformer layers and processes each input symbol a varying number of times, determined by the model at processing time. In contrast to ACT, the steps in the Transformer are the number of layers. Related, a Transformer model with a fixed number of layers that do not share weights can decide to stop computation before the final layer has been reached and output the representation from the layer after which it stopped. These so-called *early exit* strategies have been proposed for encoder-only models such as BERT (e.g. Xin et al., 2020, among others), and encoder-decoder models like T5 (Schuster et al., 2022).

Most effort in improving the Transformer architecture has gone into alleviating its main computational bottleneck: the quadratic complexity of self-attention. This especially limits the processing of large input sequences. Transformer-XL (Dai et al., 2019) divides the input into chunks and lets the current chunk look at the cached representations of the previous chunks, thereby enabling learning longer dependencies. The Sparse Transformer (Child et al., 2019) replaces dense attention computation between every input token with two fixed sparse attention patterns. The patterns are allocated to attention heads, and either correspond to attention in the local neighborhood or strided attention. They also introduce the ResNet-like Transformer architecture that we adopt for the ResNet Transformer in Section 4.3.3. The Longformer (Beltagy et al., 2020) extends the Sparse Transformer by introducing dilated attention and giving specific tokens access to full attention, both to increase the receptive field. They increase the dilation in higher layers, to further increase the dependency distances. The Linear Transformer (Katharopoulos et al., 2020) formulates self-attention as a dot-product of linear kernel feature maps and limits it with a causal mask, meaning tokens can only attend to previous tokens in the sequence. The Perceiver (Jaegle et al., 2021) uses a downsampled latent array to represent information about the sequence. New information is integrated into the latent memory with a cross-attention module, then self-attention is performed just in the reduced-size latent space. These and more efficient Transformer variants are surveyed in Tay et al. (2023).

The Transformer has also been a testbed for neural architecture search. For example, the Evolved Transformer (So et al., 2019) is the result of an evolutionary architecture search starting from the original Transformer, which performs minor adaptations to individual sublayers, such as replacing a feed-forward operation by a depth-wise separable convolution, to slightly increase performance on the task of machine translation. For the Primer architecture (So et al., 2021), the neural architecture search space consists of TensorFlow primitives. The search finds an architecture that reaches a vanilla Transformer's performance on autoregressive language modeling 2–4 times faster. The largest improvements are gained from squaring ReLU activations and inserting a 3-by-1 convolution after the projection into queries, keys, and values and before multi-head self-attention. For further information on neural architecture search for Transformers, we recommend a recent survey such as Chitty-Venkata et al. (2022).

4.8 Conclusion

We proposed the introduction of an inductive bias for entailment relationships to current architectures for sequence processing. Specifically, we introduced entailment versions of the LSTM and the Transformer, and compared them with their base algorithms on natural language inference and language modeling. We found that while our variations did show promising results in initial experiments, tuning hyperparameters lead to similar performance to the base models, with no consistent improvement on either task. We assume that these models are such powerful learning methods that the inductive bias we presented is too weak for them to produce qualitatively different outcomes. We observed a similar tendency in related work, where small changes do not lead to consistent improvements across datasets and tasks. Future work should therefore investigate more fundamental changes to the architecture. For contextual entailment vectors, future work should aim to establish that the desired relationships are learned on smaller architectures with reduced capacity and potentially synthetic data.

5 Salient Information Extraction and Representation

Chapter Summary

A document can be understood as a collection of pieces of information, which we call *semantic text units*. For summarization, a subset of all semantic text units is relevant. The goal of a summarization model is to identify the salient parts of the source document.

Current models process the input text as a flat series of tokens. We aim to group tokens into semantic units. The summarization model then decides which units to extract or pay attention to when writing the summary. This hierarchical approach promises better interpretability, as the targets of attention are more meaningful than individual tokens, and also better controllability. To learn the semantic units from the text, we use methods from computer vision that identify objects in a scene and learn representations for them. We evaluate different models and settings, and analyze the attention patterns of the individual model components. Learning representations of semantic text units that are useful for extractive or abstractive summarization proves to be difficult.

Current summarization models are designed to process sequences of symbols. As of now, these are series of subword tokens, typically created by byte-pair encoding (Sennrich et al., 2016). We conjecture, however, that the units of meaning reside at a higher level of abstraction, at the phrase level. In this chapter, we aim to detect semantic units in an unsupervised manner during summary generation. A successful partitioning of the source document into semantic units could enable a different style of summary construction as a composition of units, and evaluation as determining their presence. The result would be easier to understand and control.

5.1 Semantic Text Units

We informally define a semantic text unit (STU) to be a phrase that constitutes a single piece of information. This can be, for example, a concept, entity, object, event, or action. STUs are unique in a document. If an STU appears multiple times in a source document, its representation should be shared. This should make its identification easier and simplify the model's task to relate the pieces of information in the text. In contrast to the definitions mentioned below, semantic text units (a) are extracted by the model instead of human annotators, (b) are taken from the source document instead of the summary, and (c) are not on their own a fact that could be verified. Only in their use together with other semantic text units do they become meaningful statements. We explicitly do not want to make more prescriptions and let the model decide what the best form of semantic text units should be.

Relation to summarization content units. Summarization content units (SCU) are defined in Nenkova and Passonneau (2004). The authors do not explicitly define SCUs, but rather posit that they emerge from annotating different summaries of the same source document and are not bigger than a clause. They are the overlapping parts of the individual summaries and could be described as facts or statements. Thus, they are typically longer than STUs and contain multiple concepts.

Relation to factoids. Factoids are very similar to SCUs. They are defined as atomic information units, and also exist in relation to a set of summaries (van Halteren and Teufel, 2003). The difference lies in their construction. Instead of SCUs, which are found by looking at overlapping information in the summaries, they are created based on a single summary. The atomic information units are listed as factoids. In a second pass over the summaries, factoids are merged if they are never mentioned independently.

Relation to information nuggets. A similar concept exists in question answering. Information nuggets are facts, for which a binary decision whether the answer contains the fact is possible (Voorhees, 2003). Information nuggets again are defined with respect to available facts in the set of answers.

Difference to related concepts. We illustrate the difference between STUs and SCUs, factoids, and information nuggets with an example. Consider the following sentence of the first reference summary of the CNN/DailyMail validation set:

> Flight HX337 was carrying 295 people from Beijing to Hong Kong.

Three possible SCUs, factoids, or information nuggets for this sentence are:

- The flight number was HX337.
- The plane was carrying 295 people.
- The plane was flying from Beijing to Hong Kong.

Compare this to our proposed STUs, appearing verbatim in the source document:

- Flight HX337
- carrying 295 people
- from Beijing to Hong Kong

Our example shows that STUs are in general much shorter than the previously mentioned concepts, and only become statements once combined.

Other definitions. In other work, semantic units are the minimum units of semantic information, for which their truth can be determined (Zhong, 2017; Peyrard, 2019a), atomic facts that no longer need to be split to compare summaries (Liu et al., 2022a), or simply a fixed-size sliding window over the source document (Wu et al., 2022).

5.2 Object Detection and Representation in Computer Vision

Identifying and representing objects in an image is a prominent topic in computer vision research. We present three well-motivated approaches that influenced our work in this chapter.

5.2.1 Capsule Networks

Capsule networks combine ideas of representing objects and object parts with a communication protocol between the different representations (Sabour et al., 2017). The object representations, called *capsules*, can be seen as semantic visual units at different levels. Capsule networks are used to classify single and multiple overlapping MNIST digits (LeCun et al., 1998). The learned representations are also used to reconstruct digits, which is especially challenging for overlapping digits. We now describe the details of capsules and the dynamic routing protocol that forms higher-level representations from lower-level ones.

Algorithm 1 Dynamic routing algorithm from Sabour et al. (2017). Over r iterations, coupling coefficients c_{ij} between lower-level and higher-level capsules are refined. The output vector of the higher level is computed as a weighted combination of lower-level output vectors u_j, then squashed to be in the range $[0,1]$. The logits b_{ij} used to compute the coupling coefficients are updated after each round based on the dot-product of u_j and v_i.

1: **Input**: lower-level inputs u_j, number of iterations r, layer l
2: **Output**: higher-level output v_i
3: for all capsules j in layer l and capsules i in layer $l+1$: $b_{ij} \leftarrow 0$.
4: **for** r iterations **do**
5: for all capsules j in layer l: $c_j \leftarrow \texttt{softmax}(b_j)$ # softmax over outputs
6: for all capsules i in layer $l+1$: $s_i \leftarrow \sum_j c_{ij} W_{ij} u_j$
7: for all capsules i in layer $l+1$: $v_i \leftarrow \texttt{squash}(s_i)$ # squash to interval $[0,1]$
8: for all capsules j in layer l and capsules i in layer $l+1$: $b_{ij} \leftarrow b_{ij} + W_{ij} u_j \cdot v_i$
9: **return** v_i

Capsules. Capsules are vectors that represent an entity, such as an object or part of an object. Each capsule is hard-wired to a single entity, so the definition of the network architecture determines how many entities can exist at each level (corresponding to layers). The individual dimensions of the vector correspond to different properties of the object, such as positions, size, or texture. The vector length encodes the probability of the object's existence in the image and is scaled to a value between 0 and 1.

Dynamic routing. The lower-level capsules j[1] decide which higher-level capsules i to send their information to, by assigning each higher-level capsule a coupling coefficient c_{ij}. The coefficients of a lower-level capsule sum to 1, so they can be interpreted as connection strengths between the lower- and higher-level capsule. They are recomputed for every input image, which makes the routing dynamic. The output vector v_i for the higher-level capsule is computed from the lower-level output vectors u_j, multiplied with a learned weight matrix W and weighted by the coupling coefficients c_{ij}. The coupling coefficients are initialized from learned log prior probabilities. Dynamic routing happens for a predefined number of iterations r, over which the coupling coefficients and the output vector are refined. Coupling coefficients are updated based on the similarity of lower- and higher-level vectors, measured by their dot-product. In the paper, this is also termed *routing-by-agreement*. The dynamic routing algorithm is shown in Algorithm 1.

In essence, dynamic routing is very similar to attention (see Section 2.1.3), with the difference that (a) attention probabilities and outputs are refined over multiple iterations, and (b) the inputs decide which outputs to send their information to in line 5, whereas in attention the outputs decide how much of each input they want to use. Formally, dynamic routing computes

[1] We swap the use of i and j in the paper to match our formulation in the rest of the thesis. j is the index for inputs (lower level) and i for outputs (higher level).

coupling coefficients by taking the softmax over outputs, whereas attention normalizes over inputs:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_k \exp(e_{ik})} \quad \text{(attention)} \tag{5.1}$$

$$c_{ij} = \frac{\exp(e_{ij})}{\sum_k \exp(e_{kj})} \quad \text{(dynamic routing)} \tag{5.2}$$

We call the traditional approach *top-down attention* since the higher level (outputs) selects information from the lower level (inputs). The variant from dynamic routing where inputs send their information to the higher level is called *bottom-up attention*.

5.2.2 GLOM

GLOM is a proposal for a neural network architecture that can parse an image into a part-whole hierarchy (Hinton, 2021). In contrast to capsule networks, the parts of the network are not statically assigned to a single entity, and depend on the input image. Representations at each level communicate with representations at the level below and above, and pay attention to nearby representations at the same level. Representations are updated over multiple iterations. For higher levels of abstraction, the number of objects in the image that should be represented decreases. Instead of limiting capacity, GLOM encourages representations for the same object to converge. These can then be clustered into so-called *islands*. A grid is put on top of an image, dividing it into multiple cells. Each grid cell is called a *column* that consists of stacked autoencoders that learn local representations for their image region. The weights between all columns are shared. During the reconstruction of the input, the decoder can look at the column responsible for the area of the image it wants to generate, and pay attention to the representation at the level that is most useful. As an example, if it reconstructs the image of a cat and needs to draw an eye, it would pay most attention to the corresponding mid-level representation.

Forming islands. To encourage islands of similar representations to form, GLOM proposes a different version of self-attention for same-level representations. By avoiding a projection into query, key, and value spaces in Equation 2.21, GLOM performs self-attention on the input representations. As a result, contributions from similar vectors get higher weight in the dot-product in Equation 2.23, so similar vectors become more similar over multiple attention iterations. On the task of inpainting, i.e. filling in masked parts of an image, GLOM expects this formation of islands to create useful representations for the missing parts of the image.

An initial implementation of the GLOM architecture found good reconstruction accuracy on a synthetic dataset, both at the level of individual parts as well as entire objects (Culp et al., 2022).

5.2.3 Slot Attention

Slot attention (Locatello et al., 2020) is a method from computer vision to compute representations for objects in an image. It works as an autoencoder that encodes the image into a fixed number of representations (called *slots*) and then tries to reconstruct the original image from these representations. The slots compete to explain parts of the input over multiple iterations of attention. The paper demonstrates that slots bind to the individual objects in the image. Superfluous slots bind to the image background. The experiments are conducted on synthetic datasets for unsupervised object discovery and supervised set prediction. We next describe the slot attention algorithm.

Slot attention algorithm. The algorithm for slot attention from Locatello et al. (2020) is shown in Algorithm 2. In lines 8 and 9, a function similar to the (query-key-value) dot-product attention of the Transformer is computed. In contrast to the Transformer, which computes the softmax over keys, slot attention computes it over slots, i.e. the queries (see line 8). This means that instead of each query deciding which keys to look at, in slot attention the inputs decide which slots they should send their information to. Moreover, in line 9, slot attention normalizes a second time along the axis of inputs, which means that the attention weights are no longer normalized over slots. According to the authors, this was done to "improve stability". Algorithmically, it seems that this step negates the competition among the slots to explain parts of the input that was achieved with the first normalization. In line 10, each slot's updates are integrated with the representation of the previous iteration through a GRU cell, followed by passing through an MLP in line 11 with a residual connection. For comparison, in a Transformer layer, the outputs of cross-attention are fed to two MLPs with a nonlinearity in between (see Equation 2.25). The slot attention algorithm stops after a set number of iterations T.

5.3 Slot Attention for Text

We want to use slot attention for text representation. While it is applicable to language understanding more generally, we apply it in the context of summarization. We hope to discover the semantic text units with the object discovery inherent to slot attention. Our expectation is that the semantic units will turn out to be the salient phrases of the source document.

As a base model, we use BERTSUMABS (Liu and Lapata, 2019b), described in Section 3.2. It consists of a pretrained BERT encoder and a randomly initialized Transformer decoder and is trained on a summarization dataset. We employ the slot attention module on top of the encoder outputs z, and pass the slot outputs z' as inputs to the Transformer decoder that

Algorithm 2 Slot attention algorithm from Locatello et al. (2020). The input is a set of N vectors of dimension $D_{\texttt{inputs}}$ which is mapped to a set of K slots of dimension $D_{\texttt{slots}}$. They initialize the slots by sampling their initial values as independent samples from a Gaussian distribution with shared, learnable parameters $\mu \in \mathbb{R}^{D_{\texttt{slots}}}$ and $\sigma \in \mathbb{R}^{D_{\texttt{slots}}}$. They set the number of iterations to $T = 3$.

Input: $\texttt{inputs} \in \mathbb{R}^{N \times D_{\texttt{inputs}}}$, $\texttt{slots} \sim \mathcal{N}(\mu, \text{diag}(\sigma)) \in \mathbb{R}^{K \times D_{\texttt{slots}}}$
Parameters: k, q, v: linear projections for attention; `GRU`; `MLP`; `LayerNorm` (x3)
Output: $\texttt{slots} \in \mathbb{R}^{K \times D_{\texttt{slots}}}$
  `inputs = LayerNorm(inputs)`
  for $t = 1 \ldots T$
    `slots_prev = slots`
    `slots = LayerNorm(slots)`
    $\texttt{attn} = \texttt{Softmax}(\frac{1}{\sqrt{D_{\texttt{slots}}}} k(\texttt{inputs}) \cdot q(\texttt{slots})^\top, \texttt{axis='slots'})$ # norm. over slots
    $\texttt{updates} = (\texttt{attn/Sum(attn, axis='inputs'))}^\top v(\texttt{inputs})$ # norm. over inputs
    `slots = GRU(state=slots_prev, inputs=updates)` # GRU update (per slot)
    `slots += MLP(LayerNorm(slots))` # residual MLP (per slot)
  return `slots`

integrates its information with cross-attention to generate a summary $\hat{y}$.

$$z = \text{BERT}(x) \tag{5.3}$$

$$z' = \text{Slot Attention}(z) \tag{5.4}$$

$$\hat{y} = \text{Decoder}(z') \tag{5.5}$$

Aside from tuning the slot attention algorithm's hyperparameters (number of slots K, number of iterations T, dimensionality D of inputs and slots), we also experimented with several architecture components. Our adaptations can be grouped into four topics: modifying the slot initialization (Section 5.3.1), changing the normalization (Section 5.3.2), utilizing the slots as an information bottleneck (Section 5.3.3), and guiding the attention from slots to encoder outputs (Section 5.3.4).

5.3.1 Slot Initialization

As presented in Algorithm 2, slots are initialized by independently sampling from a normal distribution with learned mean μ and standard deviation σ. This initialization is independent of the input and, by design, agnostic to order among the slots. This makes sense methodologically, but the slots become a bit harder to interpret. We investigate several changes to initialization below.

Per-slot μ and σ. Initial experiments showed that the learned μ and σ did not deviate much from their initialization as 0 and 1, respectively. This could mean that to be useful to all slots, the initialization has to stay generic. Following Behjati and Henderson (2021), we adapt the initialization of slots to learn a separate μ per slot. We additionally experiment with a per-slot σ. We conjecture that the specialization of individual slots could prove beneficial, or even necessary. In the Transformer architecture, it has been found that the heads of multi-head attention specialize to different syntactic and semantic functions in specific layers (Voita et al., 2019).

Initialization from a mixture model. Initializing slots from a unimodal normal distribution may be too restrictive for our task. We also experiment with initializing each slot from a mixture of Gaussians, each mixture component with their learned μ and σ. The mixture weights of each slot must also be learned.

Initialization with positional prior. We investigate whether we can instill an inductive bias for the position in the source document into the slots. To that effect, instead of initializing from a normal distribution, we initialize each slot with an absolute position embedding taken from the BERT encoder. If we have fewer slots than position embeddings (512 in the BERT model), we average neighboring position embeddings or linearly project them into the right dimensions.

Initialize from encoder outputs. In case the initialization of slots from a normal distribution or positions is too far from a useful representation for the decoder, we also initialize the slots with the encoder outputs, which are known to be useful to the baseline decoder.

If the slot attention module does not find a way to provide additional value on top of the encoder outputs, it can decide to forward them to the decoder. Since there is no residual connection around layer normalization, attention computation, and the GRU update, the encoder outputs will still change inside each iteration of slot attention. As a sanity check, we set the updates in line 10 to be the value-projected encoder outputs.

Initializing projection matrices. We update slot representations in multiple iterations. In each iteration, we apply projection into query, key, and value space. If these projections are not (close to) the identity projection, all subsequent iterations after the first might wrongly project their inputs away from query, key, and value space. We therefore initialize the projections q, k, and v to identity projections.

5.3.2 Softmax Normalization

As we already mentioned in Section 5.2.3, the slot attention algorithm includes a double normalization on the attention weights, first over slots, and then over inputs (in lines 8 and 9 of Algorithm 2). According to the authors, the second normalization is for stability reasons. We attempt to remove the weighted mean in line 9 and directly multiply the attention weights with the inputs projected into value space:

$$\text{updates} = \text{softmax}\left(\frac{k(\text{inputs}) \cdot q(\text{slots})^T}{\sqrt{D_{\text{slots}}}}\right) v(\text{inputs}) \tag{5.6}$$

5.3.3 Slots as an Information Bottleneck

With an information bottleneck (Tishby et al., 1999), we reduce the capacity of the neural network to force the selection of important information. If we set the number of slots to the number of input tokens, slots can learn to represent positions in the input. For true learning of salient phrases or entities, they must be forced to aggregate information from multiple input tokens. We try to achieve this by limiting the capacity of the slot attention module. In abstractive summarization, limiting the capacity has the added benefit that only the relevant information for generating the summary can be kept.

Reducing the number of slots. The simplest way to reduce the capacity is to limit the number of slots. We can vary the number of slots from a very small number to an even larger number than the number of input tokens. In the best case, this could give us disentangled features of the input, similar to overcomplete dictionaries in the sparse coding literature.

Dropout. Another straightforward way to reduce capacity is to apply dropout (Srivastava et al., 2014) to slot representations. Dropout is controlled by a probability p of dropping an individual dimension from an output vector of slot attention.

Dimensionality reduction. A further common method to limit the capacity of a neural network is to reduce the size of its representations. Since we train the slot attention module from scratch, we can initialize it with a smaller slot dimension D_{slots}.

Activation threshold pruning. We prune slot outputs by the magnitude of their activations, only keeping those that exceed a threshold that is set as a hyperparameter.

L_0-drop layer. Preferably, our model would learn which slots to keep and which to ignore once slot attention has finished. Especially for summarization, where not the entire input is relevant, this is a desirable property. Luckily, the L_0-drop layer promises to do just that (Zhang

et al., 2021). It associates a gate g_i with each slot output z_i' and encourages closing the gates of unneeded slots (sets them to 0). Gates are sampled from a hard concrete distribution (Louizos et al., 2018), a differentiable approximation to a binary distribution that has most of its probability mass on its endpoints at either 0 or 1. The sparsity-inducing loss is the expected number of open gates

$$\mathscr{L}_{L_0} = \sum_i 1 - p(g_i = 0|\theta_i) \tag{5.7}$$

where θ_i are the slot-specific parameters of the hard concrete distribution. The L_0-drop loss is added to the cross-entropy loss with a scaling factor λ. We apply L_0-drop either to entire slots like in Behjati and Henderson (2021) or to the individual dimensions of the slot representations as in dropout.

5.3.4 Guiding Attention Patterns

To increase the interpretability of slots and provide an inductive bias on which part of the input they should pay attention to, we experiment with different loss terms on the attention patterns from slots to encoder outputs. The respective losses are scaled with a hyperparameter λ and added to the cross-entropy loss.

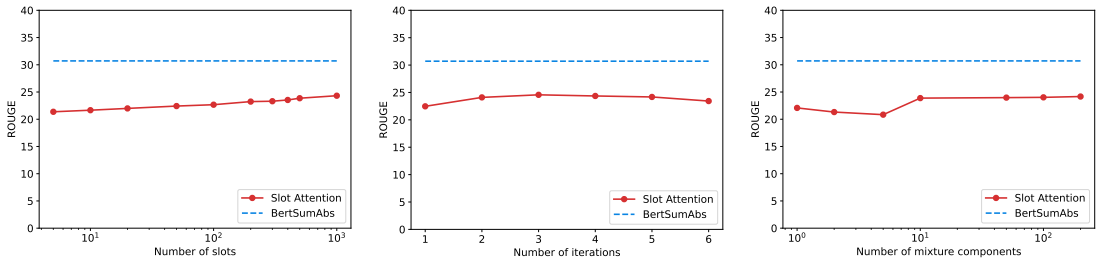
Lexical guidance. A simple attention loss based on the lexical overlap of the source document and the reference summary penalizes attention to input tokens that do not appear in the reference. Lexical overlap is computed from n-gram overlap, with $n \in 1,2$.

Semantic guidance. To avoid penalizing synonyms and related words, instead of lexical overlap we look at semantic guidance as well. We first compute BERT embeddings for tokens of the source document and the reference summary. For each source token, we use the maximum cosine similarity to any reference token as a target for how much attention should be paid to that source token.

5.3.5 Experiments

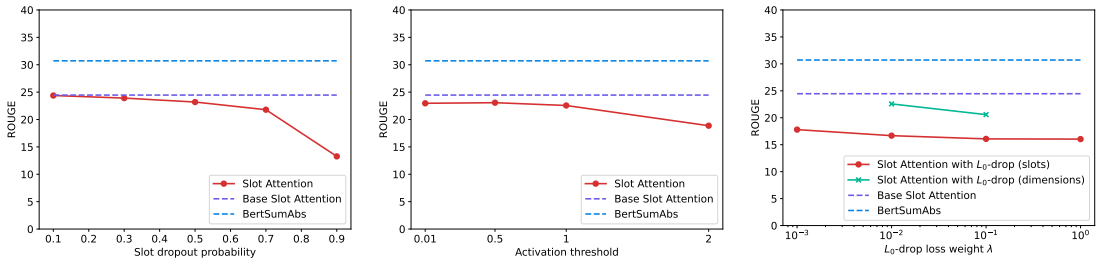
We compare our slot attention variations with the BERTSUMABS base model on the Curation Corpus (described in Section 2.5.1).

Training details. We keep the training setup fixed for all our models. We train for a maximum of 10 epochs or 30k training steps with a batch size of 5. We use early stopping based on validation performance. Our BERT encoder is a BERT-base model, and the decoder is a randomly initialized Transformer decoder with 6 layers, 768 hidden size, 8 decoder heads, 2048 FFN inner dimension, and a dropout probability of 0.1 on both the attentions and hidden



(a) Number of slots (b) Attention iterations (c) Mixture components

Figure 5.1: Slot attention algorithm experiments with 25k training steps.



(a) Dropout (b) Threshold pruning (c) L_0-drop

Figure 5.2: Slot attention information bottleneck experiments with 25k training steps.

states. We train the BERT encoder with a maximum learning rate lr_{max} of 1e−5, the randomly initialized parameters of slot attention and the Transformer decoder with a maximum of 1e−4. $\text{lr}_{\text{min}} = \text{lr}_{\text{max}}/100$ for both parameter groups. We use a warmup of 10% of training steps, where we linearly increase the learning rate from lr_{min} to lr_{max}, then decay it linearly down to lr_{min} for the rest of the training steps.

ROUGE performance. In this chapter, we define ROUGE to be the mean of ROUGE-1/2/L, to be able to compare our many variants on a single number. In all of our experiments, we observed that ROUGE-1/2/L were highly correlated, and better mean ROUGE scores corresponded to better or equal scores in each of the individual scores. We therefore do not expect to introduce a bias with this choice of reporting. The results exhibit the performance on the validation set.

In Figures 5.1 and 5.2, we show initial experiments with fewer training steps. In all of those experiments, introducing the slot attention module decreases ROUGE performance compared to the BERTSUMABS model. As we increase the number of slots, ROUGE performance steadily increases (Figure 5.1a). After $T = 3$ iterations, ROUGE performance decreases with more iterations (Figure 5.1b). Increasing the number of mixture components very slightly increases ROUGE, but only after 10 components (Figure 5.1c). For further experiments, we use a base setting for slot attention with 100 slots and 3 iterations.

Model	ROUGE	
	100 slots	512 slots
BERTSUMABS	31.93	
Base slot attention	26.62	30.21
Initialization		
Per-slot μ	26.54	-
Per-slot μ and σ	26.37	-
Positional initialization (mean)	27.21	-
Positional initialization (proj.)	24.88	-
Positional initialization (per slot)	-	28.84
Initialization from encoder (mean)	24.87	-
Initialization from encoder (proj.)	25.41	-
Initialization from encoder (per slot)	-	31.82
GRU updates from encoder (per slot)	-	31.91
Identity projections	27.00	29.90
Normalization		
Different normalization	23.56	-
Attention guidance ($\lambda = 0.01$)		
Lexical guidance ($n = 1$)	21.22	-
Lexical guidance ($n = 2$)	20.73	-
Semantic guidance	26.72	-

Table 5.1: ROUGE performance of slot attention variants with 50k training steps.

In Figure 5.2, we present the effect of different information bottleneck techniques on the ROUGE score. Figure 5.2a shows dropping out slots, Figure 5.2b depicts activation threshold pruning, and Figure 5.2c displays adding an L_0-drop layer. We see that especially adding the L_0-drop layer decreases ROUGE by a substantial margin.

In Table 5.1, we compare different initialization, normalization, and attention guidance settings with BERTSUMABS and the base setting of slot attention. We see that learning a μ and σ per slot has no substantial effect, as does initializing from mixture components (Figure 5.1c, where 1 mixture component corresponds to the base setting). A positional prior helps for slot attention with 100 slots, but not with 512. In contrast, initializing with the encoder outputs helps for 512 slots and hurts for 100 slots. As a sanity check, if we set the GRU updates to the encoder outputs, we get the same performance as BERTSUMABS, so the GRU does integrate updates properly. Initializing the projection matrices with identity matrices only slightly helps the 100-slot variant. Removing the weighted mean as a second normalization degrades the ROUGE score. Similarly, lexical guidance from unigrams and bigrams decreases ROUGE by a lot, whereas guiding the attention patterns with a semantic loss from BERT embeddings performs on par with the base model.

us president donald trump's administration has asked the us supreme court to delay a lawsuit accusing the us of failing to address climate change. the lawsuit alleges that the trump administration violated the environmental protection of fossil fuels. in 2015, president trump accused the us government of violating their rights to greenpeace. however, the court of appeals has now ruled that the lawsuit is unlawful.

Figure 5.3: Generated summary from slot attention with 100 slots.

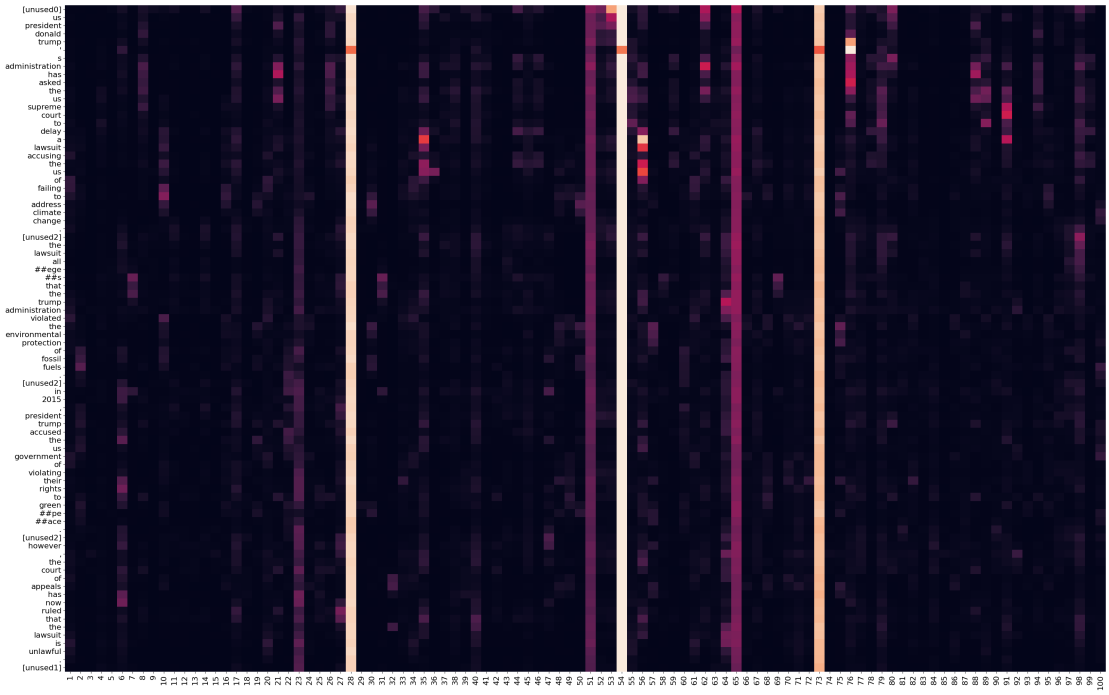


Figure 5.4: Heatmap of cross-attention from the decoder to slot representations. Attentions are normalized by each generated token (rows).

How does the decoder use the slots? We analyze the attentions from the decoder to the slot representations for our base slot attention model with 100 slots. Aside from the exceptions we mention below, the other models exhibited the same behavior. The generated summary for an example in the validation set is shown in Figure 5.3. It is qualitatively inferior to BERTSUMABS's summary, and in particular incoherent ("environmental protection of fossil fuels", "rights to greenpeace") and incorrect (the administration asked to dismiss the lawsuit, the court has ruled that the case can proceed to trial). The cross-attentions are shown in Figure 5.4. For all but one summary token, the decoder pays the most attention to the same three slots (28, 54, and 73), and a smaller amount to two further slots (51 and 65). The exception is for the generation of the apostrophe after the word "trump", where the decoder attends strongly to slot 76. The decoder attends to some slots for the generation of a specific phrase, such as slot 76 for the phrase "trump's administration has asked the", and slots 35 and 56 for the phrases "delay a lawsuit" and "the us of", but not the word "accusing" between these two phrases.

As qualitatively seen in this example, and supported by our manual inspection of many other examples, the decoder focuses on the same few slots for the whole generation of the summary, without a meaningful identification of phrases by other slots.

Initialization: The same holds true for models with different initializations, except for the ones that are initialized from encoder outputs. In these cases, the attention patterns become diagonal but are less pronounced than the ones we see when using the base model without slot attention. Since ROUGE is also lower, there is no benefit to adding the slot attention module in this case.

Normalization: If we skip the second normalization term in line 9 of Algorithm 2, the decoder pays attention to more slots (up to 50%), but again allocates the same attention weight to each slot throughout the entire generation (see Figure 5.6a).

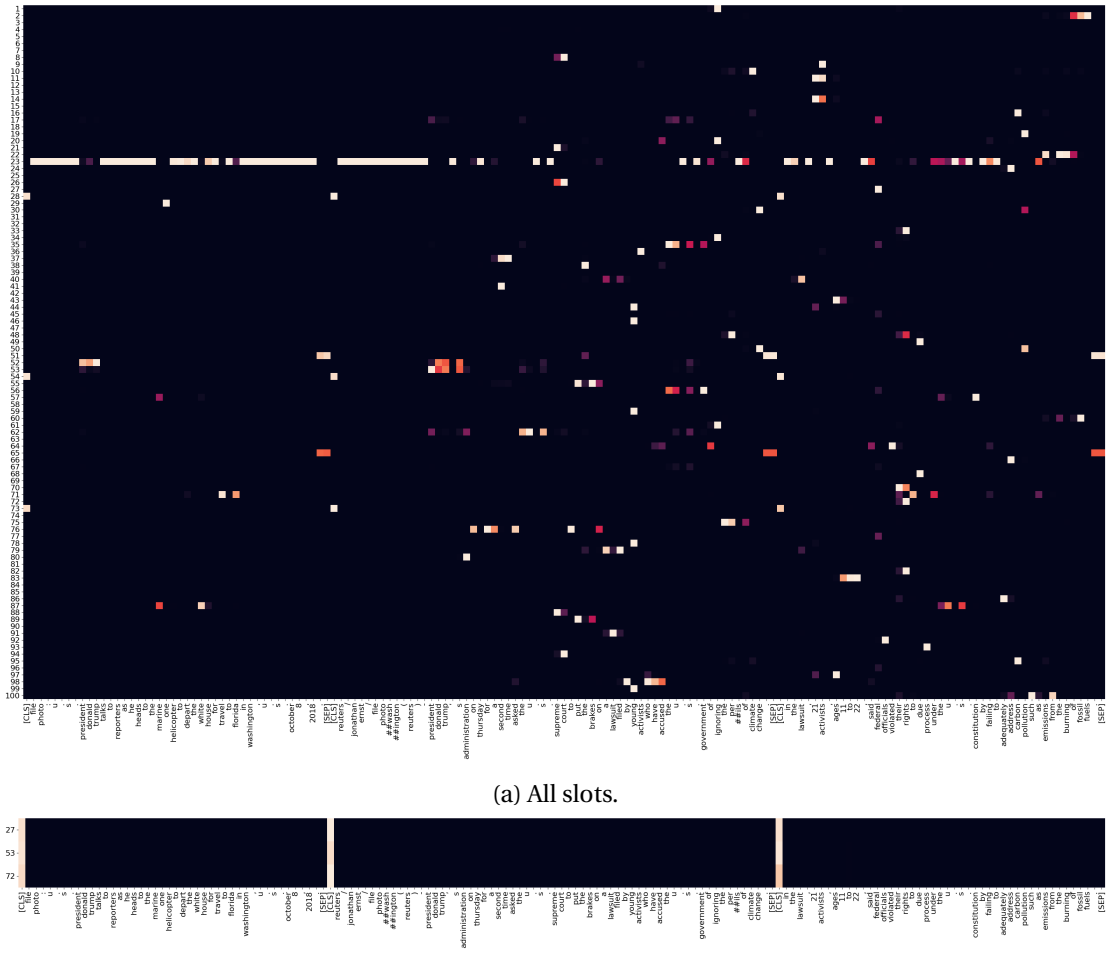
Information bottleneck: When introducing an information bottleneck through dropout, activation threshold pruning, or an L_0-drop layer, the decoder pays attention to all slots equally during generation. This holds true already from very small dropout probabilities ($p = 0.1$) or L_0-drop loss weights ($\lambda = 0.001$).

Guided attention: Semantic guidance exhibits the same attention patterns as our base slot attention model. With lexical unigram or bigram guidance, however, the maximum attention weight from the decoder is put on only one slot for generating all summary tokens (see Figure 5.7a). Our next analysis therefore focuses on what the selected slots represent.

What do the slots represent? To answer this question, we analyze the attentions from the last iteration of slot attention to the encoder outputs.[2] Figure 5.5a shows the attentions for the same model and example as before. With the exception of slot 23, which pays uniform attention to large parts of the input, the remaining slots focus on single source tokens. For example, slots 44, 46, 59, and 99 focus on "young". In particular, the slots do not seem to specialize or compete for explaining parts of the input, as desired by the method. This also holds true for other initializations, again except for initialization from encoder outputs, where the attention is diagonal.

Remember that in Figure 5.4, we saw that slots 28, 54, and 73 received the most attention from the decoder. In Figure 5.5b, we present only the attentions of these three slots. We see that they pay attention to every CLS token in the document, ignoring the rest of the tokens. We conjecture that the CLS tokens serve as a sentence representation, and the slot attention module forwards this higher-level information to the decoder. The reduced capacity (3 slot representations encode the entire source document) leads to the incoherent and factually inaccurate generated summary in Figure 5.3.

[2] According to Brunner et al. (2020), the contribution of the source token to the encoder output at the same position decreases with each Transformer layer. However, 70% of tokens are still the top contributor to the output representation at the same position (Figure 3b in their paper). We thus label the encoder outputs with the source token at that position.



(a) All slots.

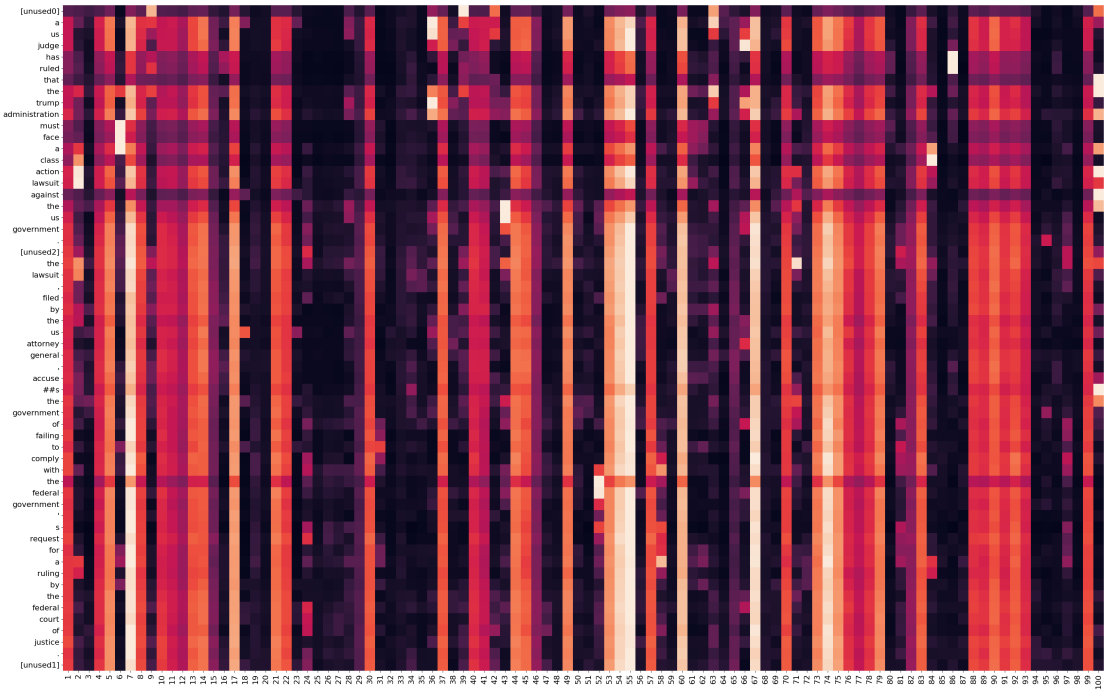
(b) Selected slots with high attention in decoder only attend to CLS tokens.

Figure 5.5: Attention from slots to encoder outputs in the base model, normalized by slot (rows). Only the first three sentences of the source document are shown.

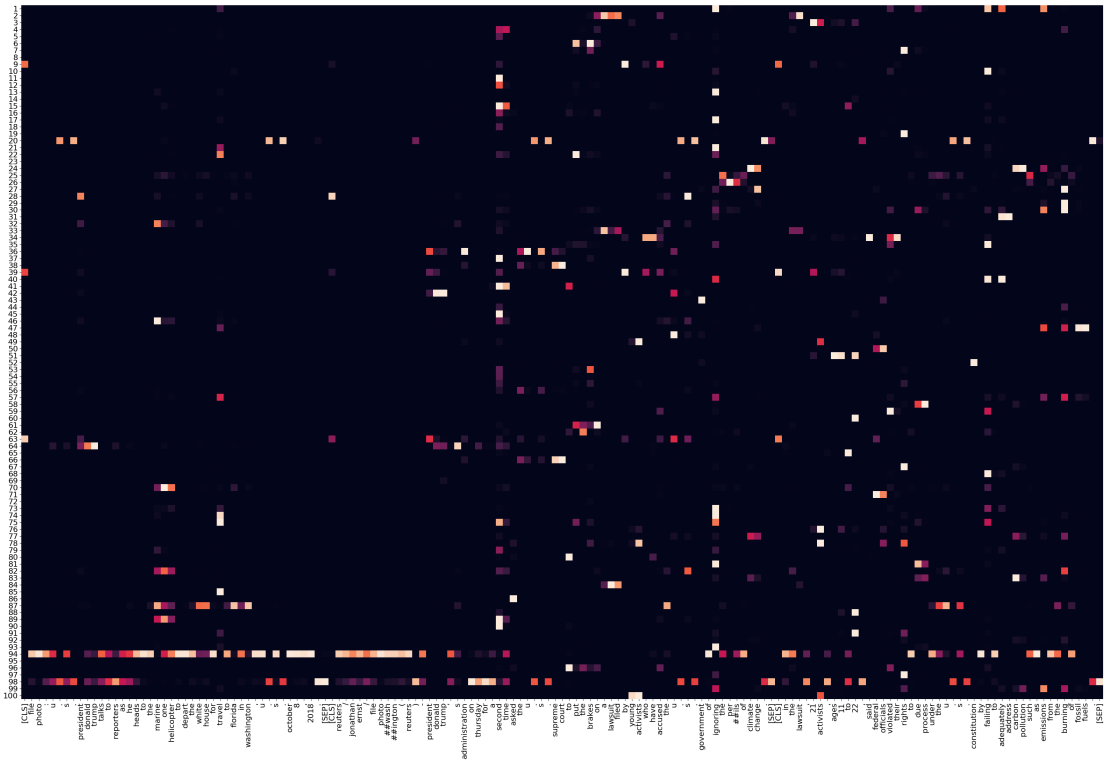
Normalization: When we remove the second normalization term, the decoder pays attention to approximately 50% of the slot representations, and that attention stays constant throughout summary generation (see Figure 5.6a). In Figure 5.6b, we see that with the exception of slots 94 and 98, slots encode individual inputs. Again, multiple slots specialize on the same source token. The attention patterns of slots 94 and 98 suggest that they could be encoding the background, i.e. unwanted information. On closer inspection, unfortunately, they also include important parts of the source document.

Information bottleneck: With an information bottleneck, all slots divide their attention equally among the entire source document, and no meaningful pattern emerges.

Guided attention: Semantic guidance exposes the same attention pattern as the base setting. For lexical guidance, every slot pays attention to the same few source tokens, which are salient tokens. For different weights λ of the focused attention loss, this corresponds to a

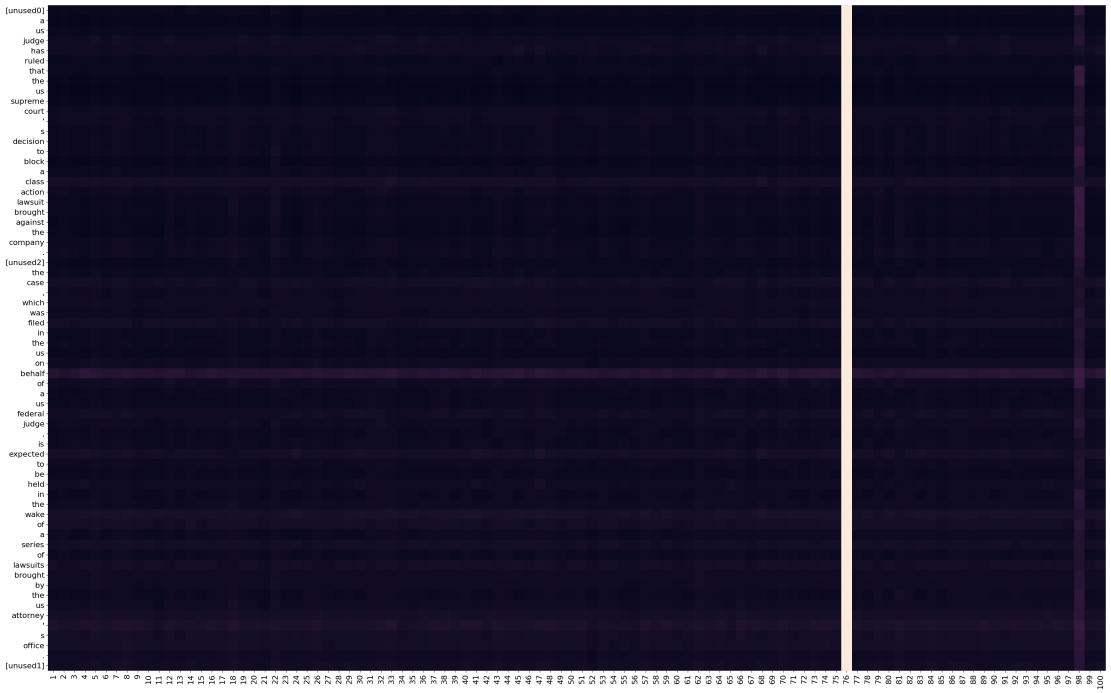


(a) Attention from decoder to slots.

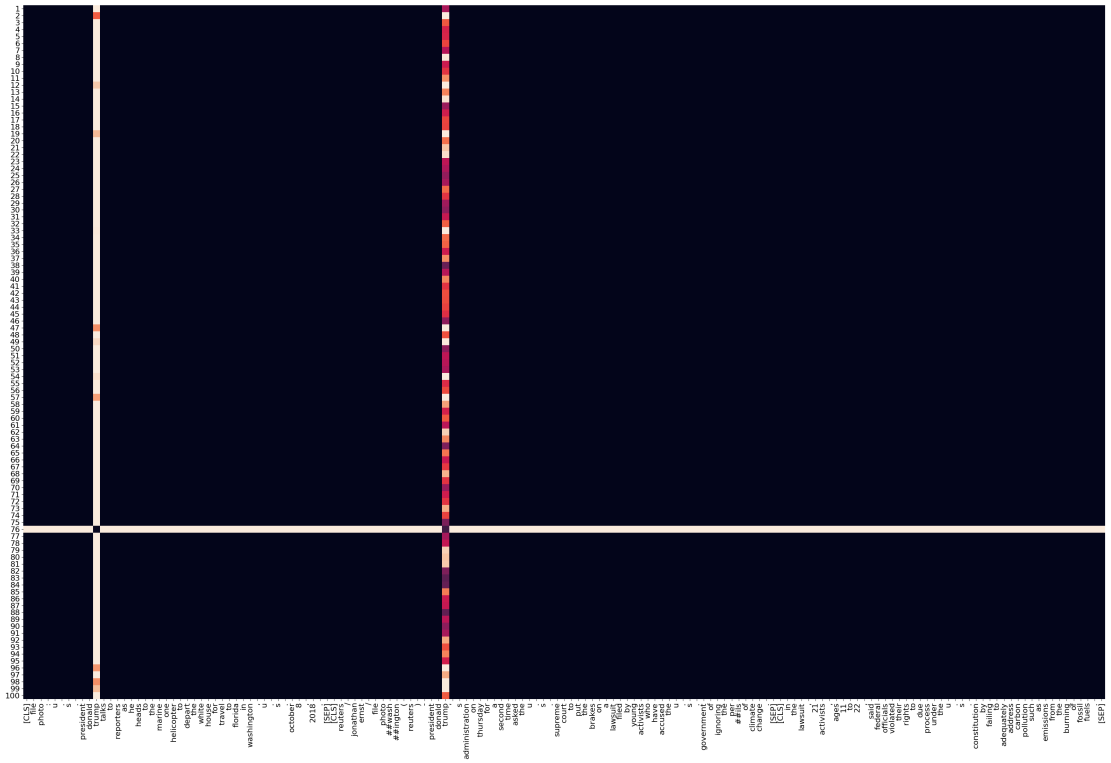


(b) Attention from slots to encoder outputs.

Figure 5.6: Attention in the slot attention model without second normalization term.



(a) Attention from decoder to slots.



(b) Attention from slots to encoder outputs.

Figure 5.7: Attention in the slot attention model with lexical bigram guidance.

different number of source tokens. For λ = 1e−2, all slots look at a single source type ("trump", appearing twice in Figure 5.7a). For λ = 1e−3, it is 3 source tokens, and for λ = 1e−4, the attention pattern is back in the base setting. However, the notable exception to this rule is the one slot that receives all attention from the decoder, which pays equal attention to all source tokens, except for the selected source type (see Figure 5.7b). It looks like the model has found a way to bypass our attention guidance: By putting the attention of all slots except one on a single salient source token, it minimizes our guidance loss. It then uses the designated slot to generate the summary. The downside of its solution is the drastically reduced capacity (only a single slot representation instead of 100). The resulting summary is quantitatively (ROUGE score) and qualitatively inferior.

5.4 Pretraining Slot Attention on Extractive Summarization

Just applying slot attention in a sequence-to-sequence model on the task of summarization does not seem to find semantic units, as seen in Section 5.3. However, guiding attention patterns (Section 5.3.4) provides a promising direction for further investigation. In this section, we test if we can leverage extractive ground truth to guide the slots to specialize to the salient phrases of the source document. If we succeed, we can use extractive summarization as a pretraining objective, as in Liu and Lapata (2019b). The idea is to train our encoder on phrase-level extractive summarization, to then use it to initialize the encoder of an abstractive summarization model.

5.4.1 Modeling Adaptations

The extractive slot attention model consists of a BERT encoder followed by a slot attention module and a linear classifier. We classify whether to include a token in the extractive summary from the slot outputs. Our solution to classifying input tokens from slot representations either computes input representations from slot representations or reformulates the loss over inputs to a loss over slots (more details below). In the slot attention module, we also experimented with the adaptations presented in Section 5.3, but focus on introducing an information bottleneck in the experiments presented later on.

Phrase-level ground truth. Extractive summarization ground truth is typically not available. Prior work at the sentence level selects reference extracted sentences from the highest ROUGE-2 score with the reference summary (Liu and Lapata, 2019b). Since we aim to get phrase-level ground truth, we use two different approaches. First, a simple ground truth baseline is computed from the bigram overlap between the source document and the target summary. This approach can become noisy, since some common bigrams appear repeatedly across the source document, sometimes also in irrelevant sentences. Therefore, our second approach finds the largest lexical overlap between the source document and reference summary that does not cross sentence boundaries. It then removes the found tokens from the set

[CLS] a **hong kong** airlines flight **was forced to make an emergency landing** after **a bomb threat** sparked a scare for the carrier and passengers today. [SEP] [CLS] the airbus a330-200 was flying **from beijing to hong kong** when the airline **received a** report that there could be **a bomb** on board. [SEP] [CLS] the security scare occurred as the national people's congress, a national legislature comprised of nearly 3,000 lawmakers, met in beijing for china's most important political gathering **of the** year. [SEP] [CLS] a **hong kong** airlines flight **was forced to** land **in wuhan** after someone claimed **a bomb** was on board [SEP] [CLS] the airbus a330-200 was met by police officers and firefighters when it landed **at an airport in wuhan** [SEP] [CLS] **flight hx337 was** halfway into its three-hour journey when **it was forced to make an emergency landing** [SEP]

(a) Bigram overlap.

[CLS] a hong kong airlines flight was forced to make an emergency landing after **a bomb threat** sparked a scare for the carrier and passengers today. [SEP] [CLS] the airbus a330-200 was flying **from beijing to hong kong** when the airline **received a** report that there could be a bomb on board. [SEP] [CLS] the security scare occurred as the national people's congress, a national legislature comprised of nearly 3,000 lawmakers, met in beijing for china's most important political gathering of the year. [SEP] [CLS] a hong kong airlines flight was forced to land in wuhan after someone claimed a bomb was on board [SEP] [CLS] the airbus a330-200 was met by police officers and firefighters when it landed **at an airport in wuhan** [SEP] [CLS] **flight hx337 was** halfway into its three-hour journey when **it was forced to make an emergency landing** [SEP]

(b) Largest overlap.

Figure 5.8: Extractive phrase-level ground truth (in bold) for the beginning of the first example of the CNN/DailyMail validation set.

of overlapping tokens. These two steps are repeated until there are no tokens left in the set. The smallest acceptable overlap is of size 2. This is similar to extractive fragments (Grusky et al., 2018), but extractive fragments are selected by greedily moving over the target summary, which can result in fragmented overlap segments. Our approach guarantees that the largest overlap is always found. Additionally, extractive fragments include single token matches. Figure 5.8 shows the resulting versions of extractive ground truth on the first example from the validation set.

Computing input representations from slots. In general, the number of slots differs from the number of input tokens, and there is no ordering among slots or assignment of slots to inputs. To classify input tokens for the extractive summarization task, we compute input representations from slot representations, weighted by normalized attention scores α_{ij}. The attention scores α_{ij} are between encoder output representations z_i and slot j. The input representations for position i are computed as:

$$h_{\text{input}}^{(i)} = \frac{\sum_j \alpha_{ij} h_{\text{slot}}^{(j)}}{\sum_j \alpha_{ij}} \tag{5.8}$$

Reformulating the loss. Alternatively, we can reformulate the loss over input tokens to a loss over slots. In that case, we classify the slots, and re-weight our target by its contributions from

the inputs:

$$y_{\text{slot}}^{(j)} = \sum_i y_{\text{input}}^{(i)} \alpha_{ij} \tag{5.9}$$

The model outputs $\hat{y}_{\text{slot}}^{(j)}$ will try to predict the soft targets $y_{\text{slot}}^{(j)}$. We can now predict whether an input token is extracted by computing:

$$\hat{y}_{\text{input}}^{(i)} = \sum_j \alpha_{ij} \hat{y}_{\text{slot}}^{(j)} \tag{5.10}$$

We do not normalize the α_{ij} here since we do not want to know whether the mean of the slots extracts input i, but whether *any* of the slots extracts it.

Focused attention. We encourage each slot to focus on a specific part of the input, preferably consecutive tokens. We implement this by adding a penalty that computes the central position each slot pays attention to, and penalizes attention to different positions by their distance to the central position. The central position $p_{\text{central}}^{(j)}$ for slot j is computed as:

$$p_{\text{central}}^{(j)} = \sum_i \alpha_{ij} i \tag{5.11}$$

and the loss for a given slot j is:

$$\mathscr{L}_{\text{focus}}^{(j)} = \frac{1}{n} \sum_i \alpha_{ij} \, |i - p_{\text{central}}^{(j)}| \tag{5.12}$$

with n the number of input positions. $\mathscr{L}_{\text{focus}}$ is the mean over the per-slot losses and gets added to the cross-entropy loss, scaled by a hyperparameter λ.

5.4.2 Experiments

We compare our slot attention variations with the BERTSUMEXT base model on CNN/DailyMail (described in Section 2.5.1).

Training details. The training setup is the same for all our models. We train for a maximum of 20 epochs or 30k training steps with a batch size of 10, randomly sampling 10% of training examples. We use early stopping based on the validation F1 score. We initialize the encoder-only model from a BERT-base model. We train the BERT encoder and the classifier with a maximum learning rate lr_{max} of 1e−5, and the randomly initialized parameters of slot attention with a maximum of 1e−4. The learning rate schedule is the same as in Section 5.3.5.

F1 score. We compute the F1 score between the model's extracted tokens and the ground truth extracted tokens. The results for bigram extractive ground truth are shown in Table 5.2.

Model	F1	
	10 slots	100 slots
BERTSUMEXT	50.99	
Loss computation		
Attention-weighted slots	49.76	48.92
Reformulated loss	0.26	33.51
Information bottleneck		
L_0-drop	50.06	49.26
L_0-drop with identity projections	49.92	49.98
L_0-drop with different normalization	-	46.29
L_0-drop with identity projections and normalization	-	50.86
Focused attention		
Focused attention	45.99	47.30
Focused attention with L_0-drop	-	46.68
Focused attention with L_0-drop and identity projections	-	46.00

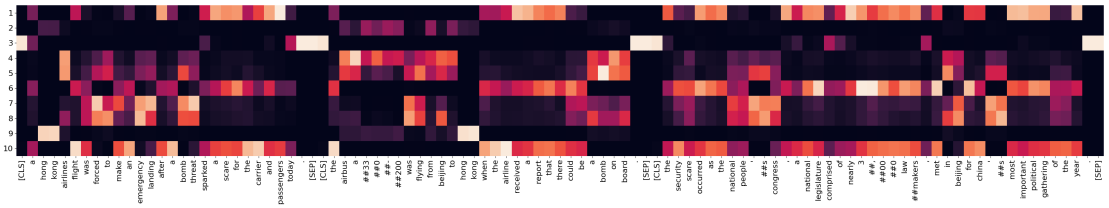
Table 5.2: F1 score for experiments with bigram overlap as ground truth extracted tokens.

Model	F1	
BERTSUMEXT	19.02	
Slot attention with L_0-drop (loss weight λ)	$\lambda = 0.01$	$\lambda = 0.001$
L_0-drop	0.00	14.38
L_0-drop with per-slot μ	20.49	17.58
L_0-drop with identity projections	16.48	17.83
L_0-drop with per-slot μ and identity projections	0.00	17.56

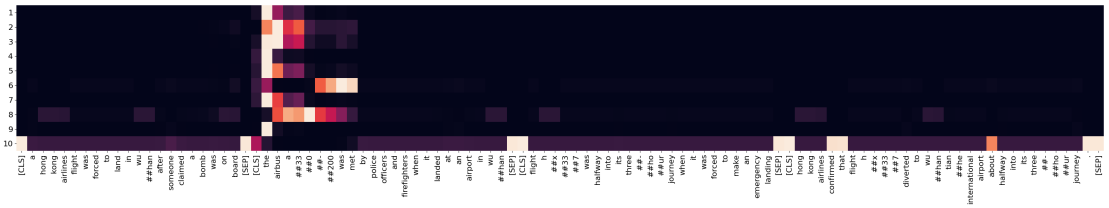
Table 5.3: F1 score for experiments with largest overlaps as ground truth extracted tokens.

BERTSUMEXT reaches an F1 score of 50.99. First, we compare our two methods of computing the loss from slot attention with 10 and 100 slots, respectively. We observe that reformulating the loss to be over slots does work well, also for other settings of slot attention. We therefore use attention-weighted slots for the remaining experiments. Most of our information bottleneck configurations work similarly well, and the setting with 100 slots, an L_0-drop layer, initializing projection matrices with the identity, and skipping the second normalization term almost reaches BERTSUMEXT's performance. Focused attention produces worse F1 scores, but we will check for interesting attention patterns further below.

For extractive ground truth from the largest n-gram overlap, the results are shown in Table 5.3. Our base model BERTSUMEXT gets an F1 score of 19.02. Compared to the bigram overlap setting, the task of extracting the largest phrases is much harder. We compare different variants of adding the L_0-drop layer to the slot attention module. Most variants perform below the baseline, except for one result with per-slot μ and an L_0-drop loss weight of 0.01.



(a) Attention-weighted slots.



(b) Focused attention.

Figure 5.9: Attention from slots to source tokens, trained on bigram overlap.

What do the slots pay attention to? We look at the attentions from slots to source tokens for three models, which each exhibit a different and interesting model behavior.

Attention-weighted slots: Our first model is the base slot attention model with attention-weighted slots, trained on bigram overlap as ground truth. We present the attention visualization of the model with 10 slots in Figure 5.9a, for the first 3 source sentences. Two slots stand out. Slot 3 pays attention to the CLS and SEP tokens, and periods. Slot 9 attends to "hong kong". In Figure 5.10, we show the extracted summary by this model. The extracted phrases are: "hong kong airlines", "airbus a330-200", "from beijing to hong kong", "a bomb on board", and "in". Comparing the extracted phrases to the attention weights in Figure 5.9a, it seems that slot 9 ("hong kong") is always extracted. Slots 4 and 5 attend to most other extracted tokens, but they also attend to tokens that are not extracted. This is also true to a lesser extent for slots 7 and 8. Slots 1, 6, and 10 attend exclusively to unextracted tokens. In particular, they seem to pay attention to the complement of the rest of the slots.

Focused attention: The second model we look at is the focused attention model. For this model, we see a very different attention pattern in Figure 5.9b. Here we show sentences $4--7$ of the source document on the x-axis. Slots $1--9$ only pay attention to a small part of the input, the token "the" at the start of the fifth source sentence, and a few subsequent tokens. Slot 10 pays attention to CLS and SEP tokens, periods, and occasional individual tokens ("confirmed that", "about"). We see the same pattern for the other examples, but slots $1--9$ attend to different words and positions in the document. The selected words do not carry significant meaning themselves or appear in especially salient contexts. Our focused attention loss defines a central position independently per slot, so it is surprising to see that apparently, all slots converge on the same central position in the document.

L_0*-drop:* Our third analyzed model is slot attention with L_0-drop, loss weight 0.01 and per-slot

Figure 5.10: Extracted summary by the attention-weighted slots base model. Green highlighted tokens are extracted by the model and reference, blue only by the reference, and red only by the model.

μ, trained on largest n-gram overlaps. In Figure 5.11, we show its attention pattern. On the left, the values of the gates in the L_0-drop layer are shown. They present a very peculiar pattern. The gates are fully open for 20 slots, 90% and 10% open for one slot each, and closed for the remaining 78 slots. The slots with open gates pay attention to large parts of the input, without focusing on anything in particular. The remaining slots have their gates closed, but pay attention to very specific parts of the source document (hong kong airlines, airbus a330-200, ","). Even though these slots are dropped, the eventually extracted summary consists of these tokens. We can only speculate how this happens. One possible explanation is that the classifier learns to extract the tokens ignored by the active slots, which would be a rather cumbersome strategy.

In summary, our extractive pretraining approaches have not led to interpretable attention patterns for our slot attention module.

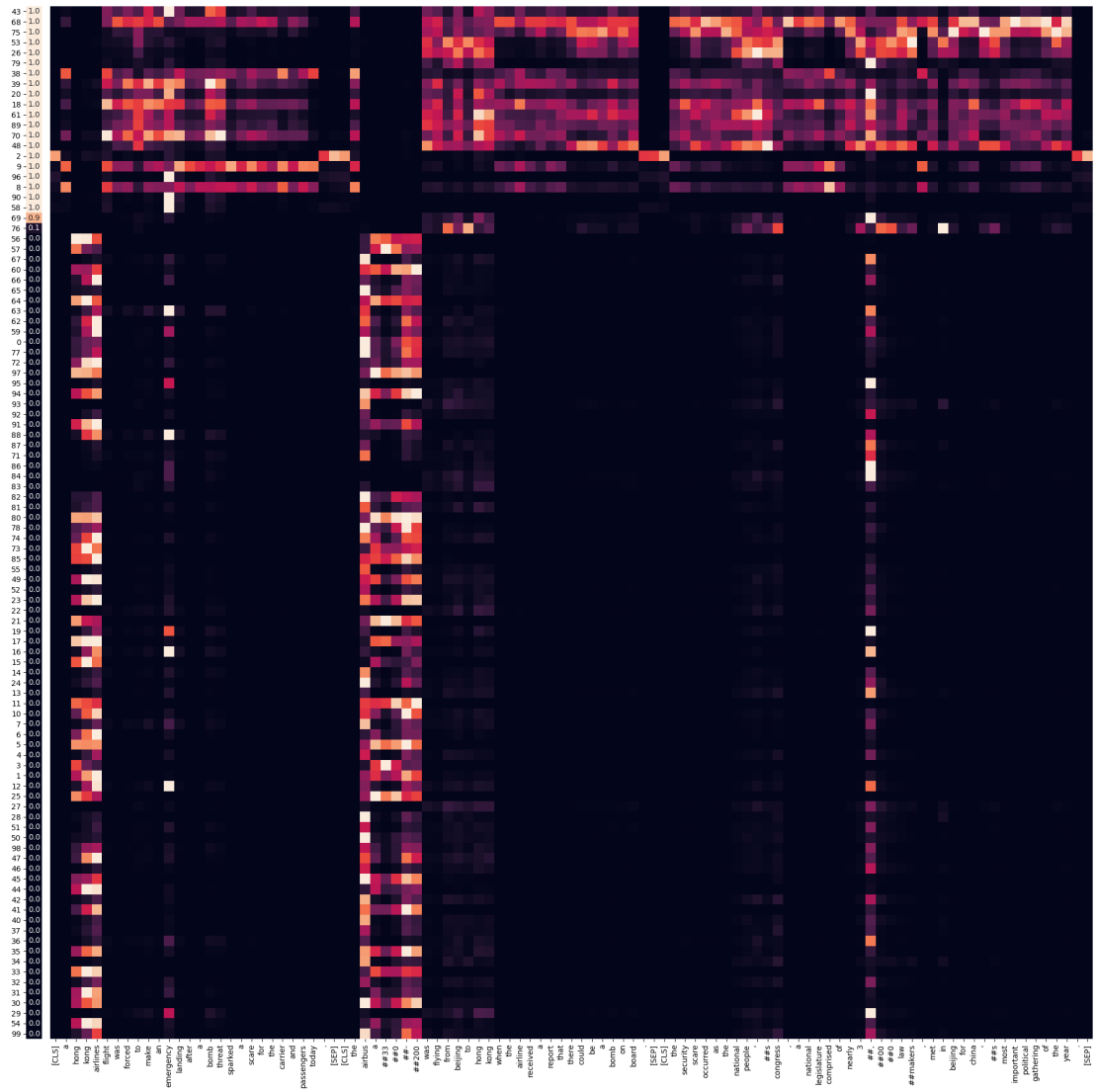


Figure 5.11: Per-slot gate values and attention weights from slots to source tokens, normalized per slot. Slots are sorted by descending gate values. The model uses L_0-drop and was trained on the largest n-gram overlaps.

Model	ROUGE	ROUGE-1	ROUGE-2	ROUGE-L
BERTSUMABS	32.43	40.82	18.44	38.02
BERT with oracle mask	54.76	61.47	43.46	59.34

Table 5.4: Abstractive ROUGE performance on CNN/DailyMail of BERTSUMABS vs. BERT with oracle mask (with a batch size of 3 and 200k training steps).

the hack was discovered by london-based security experts at mdsec. they built a device capable of simulating a person's pin entry over usb. this gadget then tries every combination until the correct one is found. hack is even able to bypass apple's built-in fail safe that locks the device after 10 incorrect attempts are made.

(a) Ground truth summary.

the gadget uses a device to bypass pin entry over usb. it 's unclear if the correct one is the device. the hack comes after 10 incorrect attempts. london-based mdsec said it is a device that can be used in a device.

(b) Generated summary.

Figure 5.12: The generated summary by BERT with oracle mask is incoherent and incorrect but achieves high ROUGE scores. ROUGE-1/2/L: 56.86/28.00/49.02.

Upper bound with oracle mask. To test the limits of our idea, we also consider an upper bound for using extractive slot attention as an initialization for an abstractive summarization model. We put an oracle mask on bigrams of the source document that also appear in the summary. As a result, ROUGE scores increase massively (by 69%, see Table 5.4) over BERTSUMABS, but summaries become less coherent and sometimes plain incorrect (see Figure 5.12). This shows that blindly improving ROUGE scores does not improve summary quality. Together with the missing interpretability link between slots' attentions and extracted tokens, we decided to not pursue extractive slot attention further.

5.5 Object Representations from Bottom-Up Attention

We would like to bring the ideas of slot attention to an architecture that is more similar to the proven Transformer architecture, which we also use for the other parts of our summarization model. One of the main ideas of slot attention is the normalization over slots in line 8 of Algorithm 2, in which the inputs decide which slots to send their information to. This idea also appears as dynamic routing in capsule networks, where the lower-level units decide to which higher-level units to send their information (Sabour et al., 2017). We call this idea *bottom-up attention*. In this section, we integrate bottom-up attention into the Transformer architecture. We aim to generate object representations at multiple levels with hierarchical bottom-up attention. We call our model the *objecter*.

5.5.1 Generating Multi-Level Object Representations

We aim to discover objects of different levels of abstraction, similar to WordNet hierarchies (Miller, 1995) or the part-whole hierarchies in capsule networks and GLOM (see Section 5.2). We mirror the different levels of abstraction in the *object generator* module of the objecter model. At each level of object representations, we reduce the number of available representations, such that each higher level encodes more general concepts than the one below.

Notation. For the task of abstractive summarization, we use x to describe the source document, with x_i being the i-th token after BPE tokenization. Similarly, y is the reference summary, y_i are the reference summary tokens, and $\hat{y}$ is the generated summary. The latent object representations z have two indices, the first one indicating their level, and the second one their position inside the level.

Object discovery model. In our object discovery model, we use a frozen BERT encoder (initialized from a BERT model finetuned on summarization) to get token representations z_{0j}. We call these *level-0 representations*. The object generator then performs bottom-up attention (see Figure 5.13) to generate higher-level representations. At each higher level, the number of representations is reduced. The number of representations per level is predefined as a hyperparameter. We then convert the representations at each level into a representation per input position, such that we can aggregate them across levels. The representations from different levels are either summed or down-projected and concatenated, to create the final object representations (with the same dimensionality as the base model). A randomly initialized Transformer decoder generates the summary autoregressively with cross-attention to these object representations.

$$z_0 = \text{BERT}(x) \tag{5.13}$$

$$(z_1, \ldots, z_l) = \text{Generator}(z_0) \tag{5.14}$$

$$\hat{z} = \text{combine}(z_0, \ldots, z_l) \tag{5.15}$$

$$\hat{y}_i = \text{Decoder}(\hat{z}, y_1, \ldots, y_{i-1}) \tag{5.16}$$

Extractive model version. We can also use the object discovery model for extractive summarization. To that extent, we replace the decoder in Equation 5.16 with a classifier that predicts the extractive label for input position j from the aggregated object representations over all levels:

$$\hat{z}_{(:,j)} = \text{combine}(z_{0j}, \ldots, z_{lj}) \tag{5.17}$$

Algorithm 3 Object generator algorithm. It creates L levels of object representations. At each level, T iterations of bottom-up attention are performed, followed by a feed-forward network. The number of outputs O_l per level is a hyperparameter and generally differs between levels (higher levels have fewer outputs).

1: **Input**: $\texttt{inputs} \in \mathbb{R}^{N \times D_{\texttt{inputs}}}$
2: **Parameters**: linear projections for bottom-up attention, feed-forward network
3: **Output**: $\texttt{all\_outputs} \in \mathbb{R}^{L \times O_l \times D_{\texttt{outputs}}}$
4: `all_outputs = [ ]`
5: **for** $l = 1 \ldots L$
6: `  outputs = Init(inputs)`
7: `  ` **for** $t = 1 \ldots T$
8: `    outputs = BottomUpAttention(inputs, outputs)`
9: `    outputs = FeedForwardNetwork(outputs)`
10: `  all_outputs += outputs`
11: `  inputs = outputs`
12: **return** `all_outputs`

Object generator module. The algorithm for the object generator is described in Algorithm 3. It creates object representations at different levels of abstraction. For each level, it initializes the outputs either from a Gaussian or from the inputs. It then performs one or more iterations of bottom-up attention, followed by a feed-forward network. The outputs of the lower level serve as the input for the next higher level. Finally, all outputs are collected and returned. A detailed description of initializing the output representations and bottom-up attention follows next.

Initialization of outputs. When creating object representations, we need to run bottom-up attention between each pair of levels. In the view of query-key-value attention, the lower-level representations serve as keys and values, while the higher-level representations are the queries. To use outputs as queries, we need to initialize these higher-level representations first (line 6 in Algorithm 3). As in slot attention, we can initialize the representations from a Gaussian distribution (for more details on initialization, see Section 5.3). Alternatively, we can also initialize them from the inputs, i.e. the lower-level representations. If there are fewer higher-level than lower-level representations, we average lower-level representations in a window to initialize higher-level ones. The window size is the number of lower-level representations divided by the number of higher-level representations. If level 0 is composed of k times as many representations as level 1, the initialization of its representations z_{1j} is

$$z_{1j} = \frac{1}{k} \sum_{m=2j}^{2j+k-1} z_{0m} \tag{5.18}$$

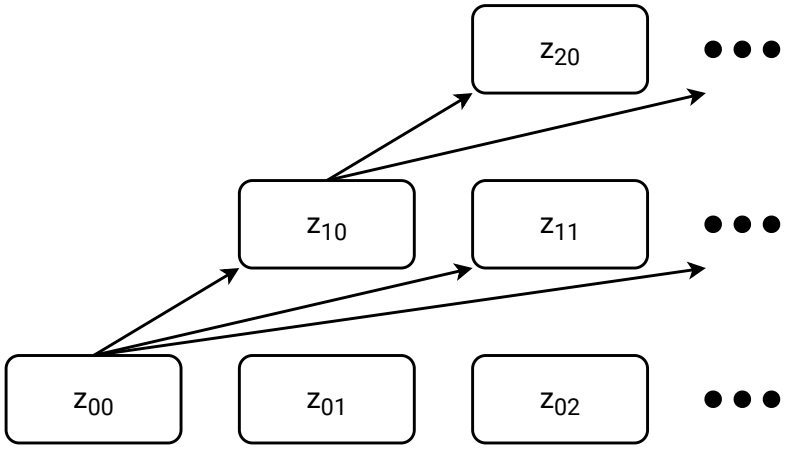


Figure 5.13: Bottom-up attention in the generator. The lower-level representations decide how much of their information to send to which higher-level representation.

Bottom-up attention. We adapt the definition of the attention sublayer of the Transformer architecture (see Section 2.1.4) for our bottom-up attention. We change Equation 2.23 and take the softmax over queries instead of over keys:

$$\mathrm{MHA}_i(q_i, k_i, v_i) = \mathrm{softmax}\left(\frac{k_i q_i}{\sqrt{d_q}}\right) v_i \tag{5.19}$$

This is the same idea as in dynamic routing in Equation 5.2. Bottom-up attention is visualized in Figure 5.13.

Combining representations. We compute input representations from higher-level object representations by multiplying each level-i representation with the cumulative attention weight from the given level-0 representation to the level-i representation and then summing those weighted representations. For level 1, this is

$$\hat{z}_{1j} = \sum_{k=0}^{n_1} z_{1k} \alpha_{jk}^{(1)} \tag{5.20}$$

with n_1 the number of level-1 representations, and $\alpha_{jk}^{(1)}$ the attention weight from the level-0 representation at position j to the level-1 representation at position k. For level 2, it is

$$\hat{z}_{2j} = \sum_{k=0}^{n_2} z_{2k} \sum_{m=0}^{n_1} \alpha_{jm}^{(1)} \alpha_{mk}^{(2)} \tag{5.21}$$

with n_2 and $\alpha_{mk}^{(2)}$ equivalently being the number of representations and attention weights for level 2.

Once we have the same number of representations for the input at every level, we either sum

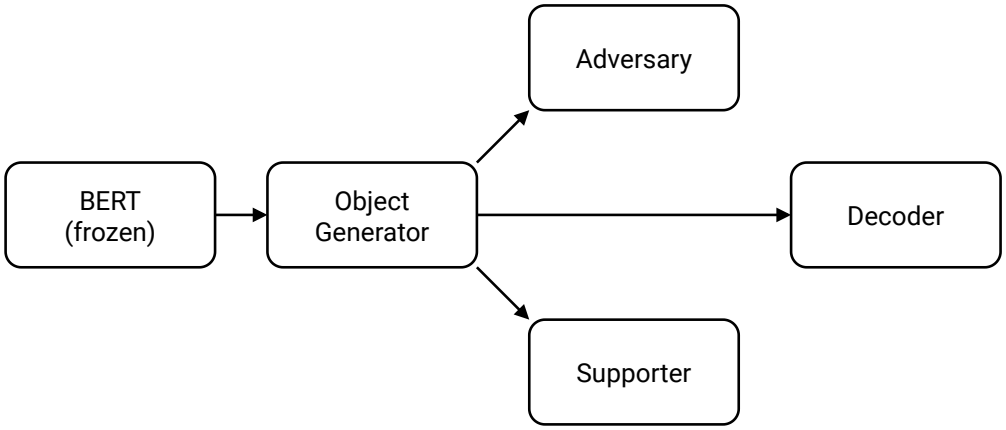


Figure 5.14: Object discovery architecture. The source document is encoded by a frozen BERT encoder. The object generator generates object representations. An adversary tries to predict masked representations from same-level representations, and a supporter tries to infer lower-level representations from higher-level ones. Finally, a decoder generates the summary using the BERT and object representations.

the representations, or down-project and concatenate them. For the latter, the idea is to assign each level to one or more attention heads in the decoder. Each attention head should only process information from a single level. The target size of down-projection is therefore a multiple of the decoder attention head size.

5.5.2 Minimizing Mutual Information with Adversarial Training

In our current architecture, multiple slots can represent the same concepts in the input. To force them to specialize to separate objects, we minimize mutual information between object representations. To achieve this, we use adversarial training (Goodfellow et al., 2014). We introduce an *adversary* and a *supporter* component that reduce same-level dependencies, but at the same time encourage between-level dependencies. Our model architecture is depicted in Figure 5.14. Both adversary and supporter are Transformer encoders that predict the embedding of a masked input as in BERT.

Adversary. The adversary tries to predict a masked object representation from all other object representations at that level. Since the BERT encoder is frozen, we do not apply the adversary to level-0 representations.

$$z_{ij} = \text{Adversary}(z_{i0}, \ldots, z_{ij-1}, z_{ij+1}, \ldots, z_{in}) \tag{5.22}$$

$$= \text{Adversary}(z_i \setminus z_{ij}) \tag{5.23}$$

The adversary tries to exploit dependencies between the representations to predict the masked one. It fails when mutual information between same-level representations is minimized. The

representations should therefore specialize to separate concepts in the text, giving us distinct object representations.

Supporter. While we minimize mutual information between representations at the same level, a masked representation should still be predictable from the combination of same-level and higher-level representations. Thus, a supporter predicts the masked lower-level representation from the remaining lower-level representations and all the higher-level representations.

$$z_{ij} = \text{Supporter}(z_i \setminus z_{ij}, z_{i+1}) \tag{5.24}$$

Loss. The object discovery model's loss is the combination of the decoder loss, the adversary loss, and the supporter loss. If the adversary performs badly in predicting the masked representation, that means that we have successfully removed mutual information between the representations, so we subtract that term from the loss.

$$\mathscr{L} = \mathscr{L}_{\text{Decoder}} - \mathscr{L}_{\text{Adversary}} + \mathscr{L}_{\text{Supporter}} \tag{5.25}$$

The adversary's and supporter's losses are the mean squared error when predicting the masked representation.

Training. The object discovery model and the adversary get updated alternatingly since their objectives are opposed. This idea comes from training the discriminator and generator in generative adversarial networks (Goodfellow et al., 2014).

Efficiency. For computational efficiency reasons, the adversary and supporter losses are only computed on a subset of the encoder representations z_i, with $i > 0$. Level-0 representations are excluded since they are the outputs of the frozen BERT model and therefore do not receive gradient updates during training.

Decoder set attention. An alternative to the combine function in Equation 5.15 is to keep each level's representations as they are and pass them to the decoder as a set. The decoder then performs cross-attention to a set of encoder representations. To achieve this, we adapt the standard Transformer decoder's cross-attention (Equation 2.28). We change it to introduce a cross-attention to every level's latent representations z_i separately and sum them:

$$b_t^l = \text{LN}(a_t^l + \sum_i \text{CrossAttention}(a_t^l, z_i)) \tag{5.26}$$

The output projection in Equation 2.24 of multi-head attention can be either applied to all cross-attention outputs jointly or learned for each level's output.

Model	F1
BERTSUMEXT	19.02
Objecter	
1 level, 10 outputs	14.16
1 level, 100 outputs	19.21
2 levels, 100-10 outputs	16.23

Table 5.5: F1 score for experiments with largest overlaps as ground truth extracted tokens.

5.5.3 Experiments

We compare our objecter variants with the BERTSUMABS base model on CNN/DailyMail (described in Section 2.5.1).

Training details. We keep the training setup fixed for all our models. We train for a maximum of 10 epochs or 200k training steps with a batch size of 3, sampling 10% of training examples per epoch. We use early stopping based on validation performance. Our BERT encoder is a BERT-base model, and the decoder is a randomly initialized Transformer decoder with 6 layers, 768 hidden size, 8 decoder heads, 3072 FFN inner dimension, and a dropout probability of 0.1 on both the attentions and hidden states. The adversary and supporter are Transformer encoders with the same hyperparameters as the decoder, except with 4 layers instead of 6. We freeze the BERT encoder and train the remaining randomly initialized parameters of the generator, adversary, supporter, and decoder with a maximum of 1e−4. We use the same learning rate schedule as in Section 5.3.5.

Performance on extractive summarization. As a start, we test the base objecter model without adversary and supporter on extractive summarization and use the largest n-gram overlap as ground truth. We combine the objecter generator's representations and BERT's outputs as described in Section 5.5.1, using the sum of level representations. In Table 5.5, we see that generating one level of 100 object representations in the generator performs on par with the BERTSUMEXT baseline. Adding another level with 10 outputs decreases the F1 score.

ROUGE performance of objecter model. Next, we turn to abstractive summarization and report the average of ROUGE-1/2/L for different variations of the objecter base model (again without adversarial training) in Table 5.6. Combining the representations by summing only provides good results with one level of object representations and 100 outputs. For down-projection and concatenation[3], adding a second level in the object generator does not decrease performance noticeably. In the second part of Table 5.6, we ablated several architecture

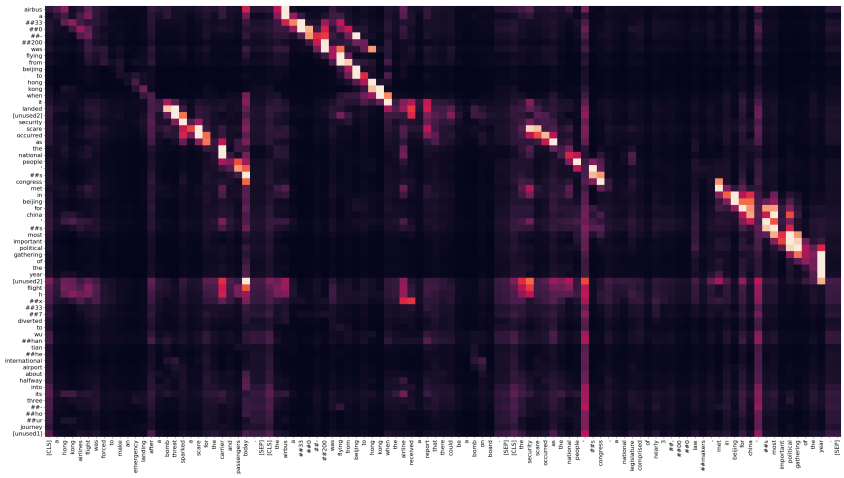
[3]We project the representations to use 4 attention heads for BERT embeddings (level 0), 2 heads for level-1 and 2 heads for level-2 representations.

Model	ROUGE
BERTSUMABS	32.59
Objecter, choice of levels and outputs	
1 level, 10 outputs (sum)	12.99
1 level, 100 outputs (sum)	32.22
2 levels, 100-10 outputs (sum)	12.87
1 level, 100 outputs (concat)	32.51
2 levels, 100-10 outputs (concat)	32.40
Architecture choices (1 level, 100 outputs)	
Iterations $T = 1$	32.22
Iterations $T = 2$	32.24
Iterations $T = 3$	14.35
Use Q, K, V projections	32.48
Use FFN layer	32.45
Share parameters between layers	32.22
Softmax temperature $\tau = 1.0$	32.22
Softmax temperature $\tau = 0.1$	32.31
Softmax temperature $\tau = 0.01$	32.13
Initialization (1 level, 100 outputs)	
Per-slot μ	32.33
Per-slot μ and σ	32.56
Initialize from inputs	32.37

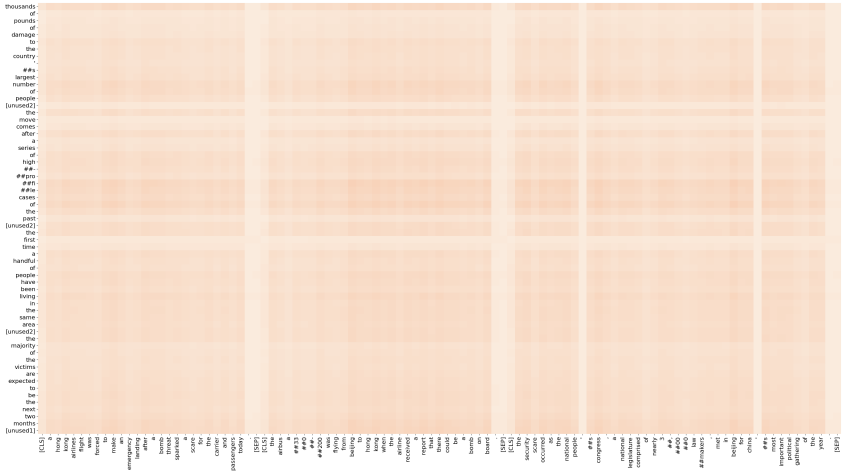
Table 5.6: Abstractive ROUGE performance on CNN/DailyMail of BERTSUMABS vs. the objecter model.

components, none of which had a significant impact on performance, except for increasing the number of bottom-up attention iterations to 3, which degrades the performance. In the last part, we show that different initializations for the object representations do not substantially change ROUGE scores.

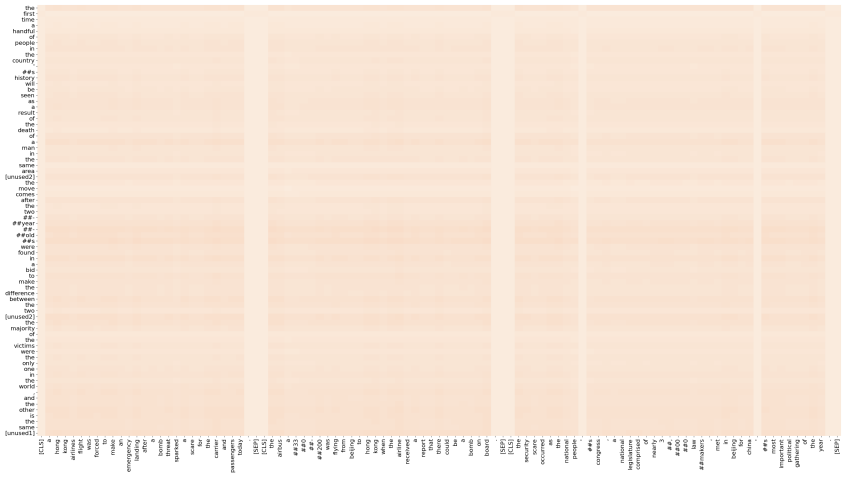
Does the decoder use the object representations? To answer this question, we analyze the attentions to the object representations. For each level, we mask all but the current level and thus force the decoder to generate a summary from the remaining object representations. In Figure 5.15, we see that only the BERT embeddings at level 0 show reasonable attention patterns. The higher levels of object representations have a near-uniform distribution of attention from the decoder to input positions, which suggests that they are ignored by the decoder.



(a) Level 0.



(b) Level 1.



(c) Level 2.

Figure 5.15: Attention from the decoder to object representations when all levels except the specified one are masked. The first three sentences of the source document are shown.

ROUGE performance with adversarial training. In Table 5.7, we show the mean score of ROUGE-1/2/L. In the first section, we note that the objecter with 2 levels scores around 2 ROUGE points lower than BERTSUMABS. It is surprisingly insensitive to changes in the loss weight λ. For efficiency reasons, we sample input positions to compute the adversary and supporter loss. Usually, they are sampled separately, and in the second section, we see that using the same positions does not change the ROUGE performance. To evaluate the importance of the BERT embeddings at level 0 in writing the summary, we gradually reduce their capacity in the third section. When allocating only a single attention head (instead of 4) to the level-0 representations, we see a drastic drop in ROUGE. Similarly, when increasing dropout on the BERT embeddings, we see that performance degrades increasingly quickly. Completely ignoring BERT representations (corresponding to a dropout probability $p = 1$) achieves a very low ROUGE score of 17.42. In the final section of Table 5.7, we present the results of decoder set attention, and compare it to the base setting with a score of 31.56. As we again decrease the influence of BERT embeddings by increasing dropout, decoder set attention with one level degrades slower than attention to a sum of the representations. With two levels, it does not achieve good ROUGE scores for any value of dropout. In the next paragraph, we take a closer look at the attentions in decoder set attention with one level.

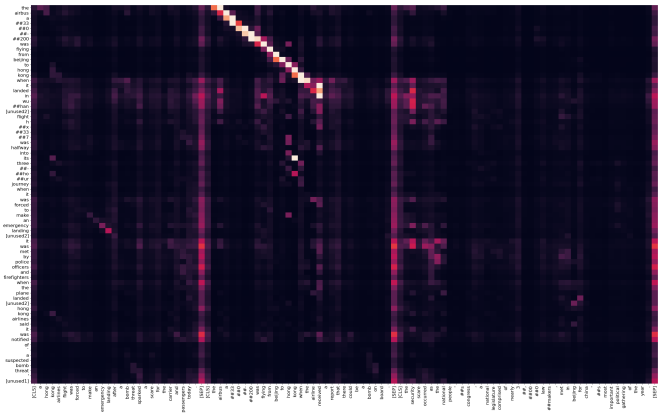
Attention to objects in decoder set attention. In decoder set attention, the object representations of each level can be accessed directly by the decoder with cross-attention. We thus do not need to mask the other levels as before. We show the attentions of a model with one level of 100 generated object representations in Figure 5.16. In Figure 5.15a, we again see the diagonal patterns in the cross-attentions from the decoder to level-0 object representations. These appear when the summary and source tokens match, a sign of copying (more on this in Chapter 6). The cross-attentions to level-1 representations in Figure 5.16b show that the decoder puts the maximum attention weight on the same slot (37) during summary generation. Slot 49 is also active throughout summary generation, with no clear pattern. With the exception of slot 16, which receives high attention for a phrase ("was flying from beijing to hong kong when it landed"), slots do not specialize to specific parts of the input. This is confirmed when looking at the bottom-up attention inside the object generator, shown in Figure 5.16c. The attention from level 0 to level 1 does not show positional or otherwise interpretable patterns. Individual source positions send their information to multiple slots equally. When we look at slot 37 in particular, it receives the most attention from the lower-level representations for "a", "to", "the", and "makers". We conclude that neither bottom-up attention nor adversarial training has resulted in interpretable object representations that specialize to inputs and seem useful for abstractive summarization.

5.6 Related Work

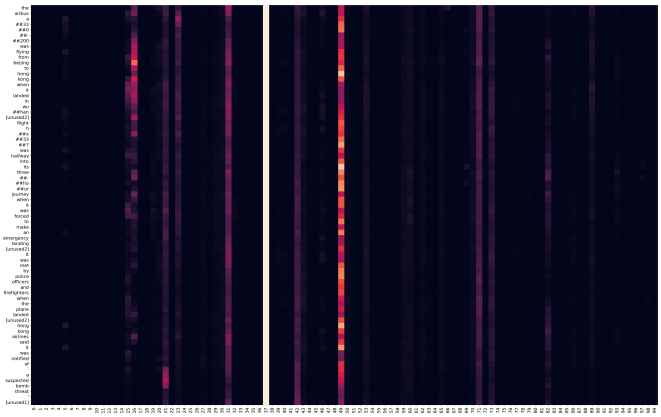
We survey methods to learn object representations in computer vision, applications to NLP, and applications of slot attention in computer vision.

Model	ROUGE
BERTSUMABS	32.59
Objecter (2 levels, 100 and 10 outputs, concatenate)	
Adversary/supporter loss weight $\lambda = 0.01$	30.80
$\lambda = 0.1$	30.84
$\lambda = 1$	30.77
$\lambda = 10$	30.87
$\lambda = 100$	29.38
$\lambda = 1000$	19.26
Share sampled positions for adversary/supporter	
$\lambda = 0.01$	30.74
$\lambda = 1$	30.94
$\lambda = 10$	30.54
Limit use of BERT representations	
Reduce capacity to 1/4	22.37
Dropout BERT with $p = 0.1$	30.91
Dropout BERT with $p = 0.5$	27.85
Dropout BERT with $p = 0.9$	21.86
Ignore BERT representations ($p = 1$)	17.42
Decoder set attention, 1 level, 100 outputs	
Combine as sum	31.56
Combine as set ($p = 0$)	31.40
Set, dropout BERT with $p = 0.1$	31.65
Set, dropout BERT with $p = 0.5$	30.54
Set, dropout BERT with $p = 0.9$	23.16
2 levels, 100 and 10 outputs	
Set ($p = 0$)	13.44
Set, dropout BERT with $p = 0.1$	13.11
Set, dropout BERT with $p = 0.5$	13.86
Set, dropout BERT with $p = 0.9$	13.85
Learned MHA output projections, 1 level, 100 outputs	
Set ($p = 0$)	32.16
Set, dropout BERT with $p = 0.1$	32.05
Set, dropout BERT with $p = 0.5$	20.09
Set, dropout BERT with $p = 0.9$	25.30
2 levels, 100 and 10 outputs	
Set ($p = 0$)	13.61
Set, dropout BERT with $p = 0.1$	13.22
Set, dropout BERT with $p = 0.5$	13.74
Set, dropout BERT with $p = 0.9$	13.44

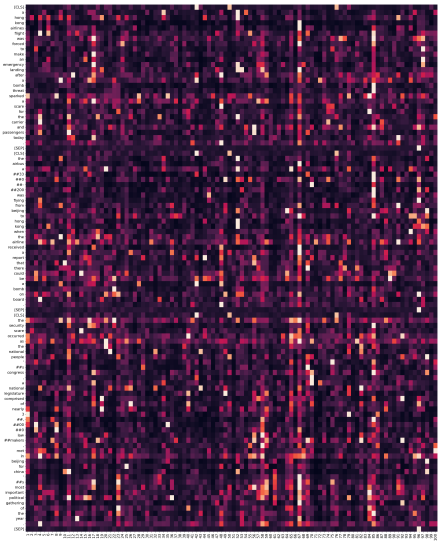
Table 5.7: Abstractive ROUGE performance on CNN/DailyMail of BERTSUMABS vs. the objecter model with adversarial training.



(a) Decoder attention to level 0.



(b) Decoder attention to level 1.



(c) Attention from level 0 to 1.

Figure 5.16: (a) and (b) Attention from the decoder to object representations at level 0 and 1. (c) Bottom-up attention from level-0 to level-1 representations in the object generator, normalized by row.

Learning object representations in computer vision. Greff et al. (2016) train Tagger, a denoising autoencoder to reconstruct the original image from corrupted versions. They use a fixed number of networks with shared weights that each represent an object, by predicting an appearance vector and group assignment probabilities for each pixel in the input. They use amortized inference over up to five iterations to update the predictions. In contrast to Tagger, IODINE is an object detection method with explicit object representations from a fixed number of latent vectors in a variational autoencoder. The decoder separately predicts the positions in the image and the appearance, refining its predictions in multiple iterations. The final image is composed of the spatial mixture of the objects (Greff et al., 2019). AIR is proposed in Eslami et al. (2016), a recurrent neural network that can handle a variable number of objects in an image by outputting the representation of one object at each time step. The network determines the number of objects by predicting the value of a latent variable to be either 1 (proceed) or 0 (stop). Crawford and Pineau (2019) introduce SPAIR, a VAE with a convolutional neural network as the encoder. They divide an image into grid cells, then predict for each cell the presence of an object, alongside its appearance, depth, position, and size. While Tagger and IODINE are *scene-mixture* models, where an image is explained by a mixture of full-scale components, AIR and SPAIR are *spatial-attention* models that obtain explicit and disentangled geometric representations of the objects. The former models are better at handling occlusions, while the latter succeed at representing the locality and properties of a single object. SPACE (Lin et al., 2020) combines these two approaches by processing foreground objects like a spatial-attention model, and decomposing overlapping objects and background segments with component mixtures, like a scene-mixture model. Burgess et al. (2019) train MONet, a scene-mixture VAE with a recurrent attention network to reconstruct image regions based on predicted attention masks. At each time step, the attention network decides which image regions (that are still unexplained) it wants to process. It determines the mask, and a VAE computes the component's reconstruction. The next time step will no longer have to explain the image region covered by the previous masks. As with Tagger, this allows for a variable number of objects per image, each with their own representation from the component VAE. Li et al. (2020a) extend MONet to use multiple views of a scene to iteratively update object representations, then predict the scene from a novel viewpoint. Engelcke et al. (2020) models interactions between object locations to decompose and generate images. They first encode the image with a CNN and then predict a fixed number of component masks with an RNN, depending on previously predicted component locations. From the mask and the image, the appearance representation for a component is independently computed and the image is reconstructed with a Gaussian mixture model. In an extension (Engelcke et al., 2021), they no longer split masks and appearance but combine location coordinates into two dedicated dimensions of image pixel embeddings. The pixel embeddings are then clustered into attention masks, from which object representations are inferred. While training is done with a fixed number of clusters, inference can be run with a flexible number. However, this produces fewer objects at the cost of worse segmentation. Xu et al. (2022) add special object tokens to the input, similar to the [CLS] token in BERT, to learn object representations with a Vision Transformer. On higher levels, the number of object tokens is reduced. After each

level (that possibly consist of multiple layers), the output representations are clustered, by assignment to the most similar object token, which serves as the cluster centroid. Related to the ideas of capsule networks and GLOM, the Complex AutoEncoder generates representations whose magnitude encodes the presence or absence of a feature, and whose clustering is determined by the relative phase difference (Löwe et al., 2022). An extension to slot attention alternates binding of slots to the objects (called spatial binding) with binding of factors of variation, e.g. shape or color, to a set of dimensions of each object's representation (Singh et al., 2023). This procedure improves disentanglement within an object's representation, and for example, allows exchanging the color of two objects by simply swapping the activations in the respective dimensions.

Applications of object representations in NLP. Slot attention has been used in NLP to induce semantic units at the character level in multiple languages (Behjati and Henderson, 2021). As in our work, different initialization techniques as well as dropping unnecessary slots with the L_0-drop layer are employed. In contrast to our work, the capacity of the decoder is limited for reconstructing the input to ensure that the learned slot representations carry most of the meaning. For generating a summary, we expect that a powerful decoder is necessary to generate a fluent and coherent output text. Lin et al. (2018) compute representations for semantic units with a convolutional neural network with dilation. The number of CNN layers as well as the dilation ratio control the granularity of semantic units (phrase, sentence, or larger). In their two-step hybrid attention, the decoder first attends to the semantic unit representations, and then to the word-level output of an LSTM. They perform no further analysis of the induced semantic units. Gyllensten et al. (2019) introduce *recursion-grams* (r-grams) that are generated by successively merging the most frequent adjacent character(s) (including white-space), as in byte-pair encoding (Sennrich et al., 2016). They use r-grams for language-agnostic text segmentation, as they can naturally extend to multi-word expressions, unlike white-space delimited segmentation. The authors train r-gram embeddings with the word2vec skip-gram algorithm (Mikolov et al., 2013a) and find that these are competitive with embeddings from white-space delimited units for various similarity and analogy tasks. When they inspect the nearest neighbors in multiple languages, they find that acronyms are close to their multi-word expressions. This study suggests that semantic units could be identified from the frequency statistics of a language alone. In previous work, multi-word expressions were also identified from linguistic knowledge about their behavior, such as orthography, inflection, or the diversity of their contexts and their part of speech (Tsvetkov and Wintner, 2011).

Multi-modal object representations. There exist several studies that learn multi-modal representations for vision and language from self-supervised pretraining through masking text, image regions, or objects. ViLBERT (Lu et al., 2019) and LXMERT (Tan and Bansal, 2019) use cross-attention between a vision and a language model to combine the two modalities. TIMAM (Sarafianos et al., 2019) obtains textual representations from BERT and visual features from a ResNet, then trains the model to make the representations from the two modalities

indistinguishable for a discriminator. UNITER (Chen et al., 2020) uses a single model, and only masks the input of one modality while using the other to predict the missing information. While they use word-image region alignment based on optimal transport, OSCAR (Li et al., 2020c) uses an object detector to label objects. They then mask either object labels or text input tokens. KD-VLP (Liu et al., 2022c) extends the word-region alignment pretraining task from UNITER to a phrase-region task. Many more vision-language pretraining approaches exist and are presented, for example, in a recent survey (Khan et al., 2022).

In image captioning, object representations play an especially important role (Mitchell et al., 2010), and can be improved and made more interpretable through matching text with the detected objects in the image (Wang et al., 2018b). For videos, ActBERT (Zhu and Yang, 2020) uses paired video sequences and descriptions to relate actions, objects, and descriptions. This helps to find videos from a text query, caption videos, and answer questions from videos. Similarly, visual reasoning with compositional natural language instructions benefits from matching text and object representations with bidirectional attention (Tan and Bansal, 2018). But text representations can also benefit from visual grounding for general language understanding tasks. Vokenization (Tan and Bansal, 2020) extends the masked language modeling pretraining objective by jointly predicting visual tokens ("vokens") from BERT encoder outputs to visually ground the representations. It shows improved performance on language understanding and question answering benchmarks.

Applications of slot attention in computer vision. The ideas of slot attention and MONet are used in Wang et al. (2021) on images to learn object representations and associated masks, then reconstruct the image. An image caption or question about the image is given to a semantic parser that identifies the objects in the sentence. A neuro-symbolic program executor then answers the question or classifies the correctness of the caption. They find that the multi-modal input improves the segmentation of objects. Gopalakrishnan et al. (2021) use slot attention to learn sub-routines (a clustering of action sequences) for an agent trained with reinforcement learning. They adapt the initialization of slots by learning slot-specific μ and σ parameters. Seitzer et al. (2023) use slot attention as an information bottleneck in an autoencoder. Instead of the inputs, the decoder reconstructs the encoder outputs. They freeze the encoder to avoid its outputs to collapse and make reconstruction trivial. For evaluation, they analyze attention patterns between decoder and slots, and slots and encoder outputs, as we presented earlier. Xie et al. (2022) note that slot attention performs worse than a trivial pixel-level representation on a compositionality task. They attribute this to slot attention assigning multiple slots to the same object, something that is not supposed to happen by design (competition among slots to explain parts of the input). Another issue they identify is that slot attention does not have blank slots that do not represent objects. This matches our findings and was the reason we tried to introduce an information bottleneck on slots, and drop them entirely with the L_0-drop layer. Kipf et al. (2022) weakly supervise the initialization of slots by assigning them to objects in the first frame of the video. Objects are identified with image segmentation masks or bounding boxes. Any superfluous slots are assigned to a null

value. The slots then manage to temporally track the objects in the video. Sajjadi et al. (2022) use slot attention for novel view synthesis, by training a mixing module that decides how to compose the objects for each novel view direction, characterized by the camera position and the direction ray pointing from the camera to the pixel in the image. Baldassarre and Azizpour (2022) train slot attention with a contrastive loss on the representations, where the model has to identify the same object in different crops of an image and distinguish them from other objects. They show that representations from regular attention, without the competition for explaining parts of the input, fail to select a single object.

5.7 Conclusion

In this chapter, we aimed to discover semantic units and unsupervisedly learn their representations as a by-product of learning to generate summaries. We found that neither slot attention nor bottom-up attention, two object discovery methods proposed for object detection in computer vision, could be successfully integrated into a summarization model. The various models we trained learned to ignore our object representations and fall back to the BERT encoder outputs, or they used the representations to encode the entire input. Further inductive biases did not help to specialize individual representations to specific parts of the input. When we forced the model to make use of the object representations by limiting access to BERT encoder outputs, the summary quality degraded drastically. To date, our initial motivation seems intuitive to us, and we are quite surprised by the difficulties we had training these models.

5.7.1 Representation Learning for Encoder-Decoder Models

With the rise of pretrained encoder-decoder models, learning representations that facilitate communication between the encoder and the decoder has become less appealing. While randomly initialized decoders benefited from structured input representations, pretrained decoders can only make use of them if they are part of the model during pretraining. An important part of pretraining an encoder-decoder model on large amounts of data is to negotiate and stabilize the communication between the encoder and the decoder. This advantage is lost when the representations are changed after pretraining and the decoder has to learn the new encoder output format. Regardless of the results in this chapter, the introduction of pretraining complicates making good use of the suggested inductive bias, further supporting the Bitter Lesson (Sutton, 2019).

A major goal of devising structured representations is interpretability, and this goal persists even after the introduction of pretraining. It is an open question, however, whether the best way to achieve interpretability is through structured encoder output representations, or if a different approach is more promising.

Evaluation Part II

6 Hallucination Detection

Chapter Summary

In abstractive summarization, hallucinations are model generations that are not supported by the source document. Current methods for detecting hallucinations operate mostly on noun phrases and named entities, and restrict themselves to the XSum dataset, which is known to have hallucinations in 3 out of 4 training examples (Maynez et al., 2020). We instead consider the CNN/DailyMail dataset where the summarization model has not seen abnormally many hallucinations during training. We automatically detect candidate hallucinations at the token level, irrespective of their part of speech. Our detection comes essentially *for free*, as we only use information the model already produces during the generation of the summary. This enables practitioners to jointly generate a summary and identify possible hallucinations, with minimal overhead. We repurpose an existing factuality dataset and create our own token-level annotations. The evaluation on these two datasets shows that our model achieves better precision-recall tradeoffs than the baselines, which additionally require a model forward pass.

Publication in this Chapter

This chapter builds on the material in our publication:
Marfurt, A., and Henderson, J. (2022). Unsupervised Token-level Hallucination Detection from Summary Generation By-products. In *Proceedings of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)* (pp. 248-261).
PDF: https://aclanthology.org/2022.gem-1.21.pdf
Code and data: https://github.com/idiap/hallucination-detection

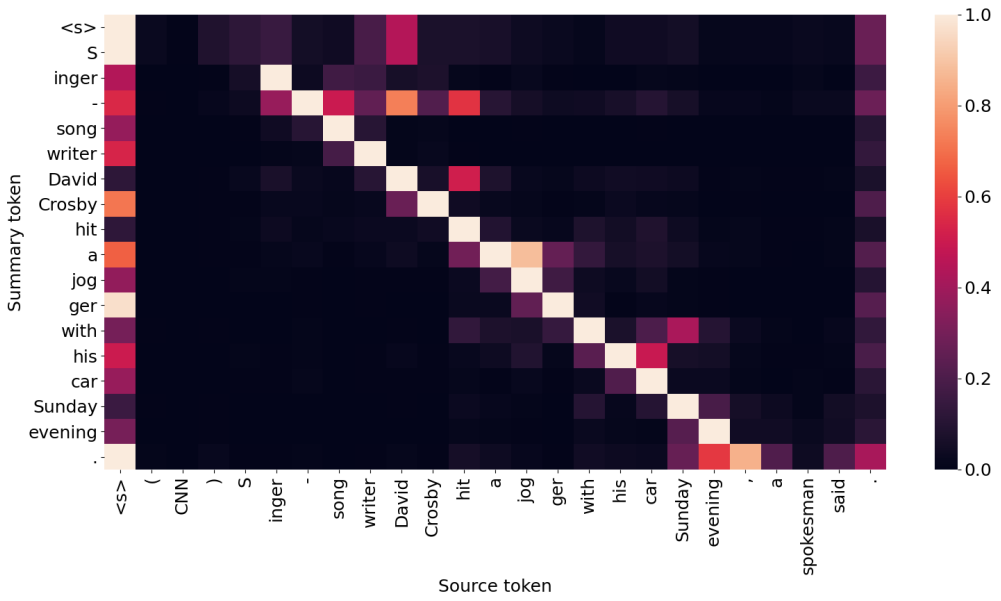


Figure 6.1: BART cross-attentions align copied segments of the summary with the respective segments in the source. Attention weights are normalized by row. Only the first summary and source sentences are shown.

After having investigated interpretable representations in Part I, we now turn to evaluate summarization models in Part II. We start by taking a closer look at hallucinations, i.e. information in the generated summary which is not faithful to the source document. Hallucinations are a prominent remaining failure mode of state-of-the-art summarization models.

We propose to use the diagonal cross-attention patterns present in Transformer-based abstractive summarization models (see Figure 6.1) to align the summary with the source document. We then detect hallucinations in an unsupervised fashion for segments of aligned and unaligned tokens by computing statistics from the encoder's self-attentions and the decoder's next-word probabilities. These by-products arise when generating a summary with any Transformer model. In this chapter, we use BART (Lewis et al., 2020). Our method demonstrates good results compared to its competitors, while at the same time requiring negligible additional computation. Token-level hallucination detection proves to be a difficult task, in particular on intrinsic hallucinations (defined in the following), where all models struggle to detect any hallucinations.

6.1 What are Hallucinations?

We adopt the definition from Maynez et al. (2020) and define *intrinsic hallucinations* as combinations of information from the source document that cannot be inferred from it, and *extrinsic hallucinations* as information that is not present in the source document. Paraphrases

and information that can be directly inferred from the source document, however, do not constitute hallucinations. Furthermore, whether some information is a hallucination is an orthogonal problem to whether that information is factually correct, a question we do not consider in this thesis.

6.2 Hallucination Detection on XSum

A lot of recent work has addressed the problem of hallucinations, predominantly on the XSum dataset (Narayan et al., 2018a). XSum is an outlier, however, in that over 75% of its reference summaries contain hallucinations (Maynez et al., 2020). Models trained (or finetuned) on this dataset are consequently prone to hallucinate themselves when summarizing an article. Additionally, current work focuses on detecting hallucinations for noun phrases and named entities (Wang et al., 2020; Durmus et al., 2020; Scialom et al., 2021), sometimes with the addition of dates and numbers (Narayan et al., 2021). Recent work has shown, however, that summarization models also make mistakes in other parts of speech, such as predicates (Pagnoni et al., 2021).

We therefore extend current hallucination detection research to CNN/DailyMail and the token level. By repurposing the existing factuality dataset FRANK (Pagnoni et al., 2021) and annotating our own TLHD-CNNDM, we contribute two evaluation datasets. They are further described in Section 6.5.1.

6.3 Hallucination Detection Methods

6.3.1 QG-QA Models

Multiple studies use automatic question generation and answering models to ask questions about entities in the generated summary, and try to answer them from the source document (Wang et al., 2020; Durmus et al., 2020; Scialom et al., 2021). If the question cannot be answered from the source document, the entity is considered a hallucination.

FEQA. FEQA (Durmus et al., 2020) generates questions about the summary's entities, then tries to answer them from the source document. It then computes the token-level F1 score between the summary's text and the predicted text span from the source. Unmatched answers indicate hallucinations. We compute word-level probabilities by averaging the F1 scores of all spans the word is part of.

6.3.2 Dependency-arc Entailment

DAE. Dependency arc entailment (DAE) (Goyal and Durrett, 2020, 2021) decides from its dependency arcs whether the generated summary sentence is entailed by the source document.

While DAE is technically a factuality detection method, we conjecture that hallucinations in the summary should not be entailed by the source document either. We thus use their method to get entailment probabilities for each dependency arc.

In footnote 6 of Goyal and Durrett (2021), the authors propose that a word is non-factual if any of its arcs are non-factual. We therefore compute word hallucination probabilities as the maximum probability of non-factuality of its dependency arcs. We use their model variant trained with entity-based synthetic data on CNN/DailyMail.

6.3.3 Token-level Prediction with an External Model

Pretrained language models can also be finetuned to directly predict a hallucination label for each input token. Zhou et al. (2021) do this with the help of synthetic training data, where factual tokens are automatically replaced with hallucinations. We call their method *Fairseq* in the following, based on its GitHub repository name.[1]

Fairseq. We use the model finetuned on XSum and evaluate how it transfers to the CNN/DailyMail dataset. Since we compare to our unsupervised method, we leave retraining the model on CNN/DailyMail to future work. We evaluate both model settings, with and without access to the reference summary.

6.4 Unsupervised Hallucination Detection

In the process of generating a summary, a Transformer-based abstractive summarization model creates a number of by-products, such as decoder next-token generation probabilities, encoder and decoder self-attentions, and decoder to encoder cross-attentions, for each layer and attention head of the model. These can be easily accessed from e.g. the Hugging Face transformers library (Wolf et al., 2020).

6.4.1 Motivation

It is debated whether model attentions can be used to explain model decisions (Jain and Wallace, 2019; Wiegreffe and Pinter, 2019) and how much a Transformer encoder's output representation still represents the token at its position in the input (Brunner et al., 2020). Nevertheless, we posit that the diagonal attention patterns observed in Figure 6.1, together with the fact that the source and target tokens match for the entire segment, is a strong enough signal to claim that a summarization model copied this segment from the source.

Additionally, we conjecture that the faithfulness of a summary to the source document is not inherently a question that spans multiple sentences, in contrast to a summary's factuality

[1] https://github.com/violet-zct/fairseq-detect-hallucination

(Pagnoni et al., 2021). As a consequence, we detect hallucinations at the token level by processing summary sentences in isolation.

6.4.2 Initial Alignment

From the observations above, we start by aligning summary and source positions based on cross-attentions. In BART cross-attentions, the maximum cross-attention weight is often put on the beginning-of-sequence token in the source. If the token is a preposition, a high attention weight is also put on its preceding and succeeding tokens. We therefore accept a target-source alignment of target token t_i iff it matches a source token in its top-4 cross-attention weights. This constitutes our initial alignment.

6.4.3 Context Voting

In the second step, we expand the initial alignment with a position-based voting algorithm. For each target token t_i, its context tokens $t_{i-l},\ldots,t_{i-1},t_{i+1},\ldots,t_{i+l}$ in a window of size l[2] around t_i vote on the expected source position of t_i given their own alignment and an assumed diagonal attention pattern. If a token is not aligned with the source, it does not vote. We accept a vote when at least half the neighboring tokens agree. We perform voting for a maximum of 10 rounds, and we stop early when it has converged, which often happens after 2 rounds.

After these two alignment stages, we have a set of aligned segments, with a token-level correspondence between summary and source, and a set of unaligned tokens. We now look to detect intrinsic hallucinations in the former set, and extrinsic ones in the latter.

6.4.4 Classifying Aligned Tokens

Aligned tokens appear in the source document, and consequently do not constitute extrinsic hallucinations. To assign a probability of them being intrinsic hallucinations, we compare the characteristics of their aligned source segments. Maynez et al. (2020) speculate that intrinsic hallucinations are potentially a failure of document modeling. We add that the encoder may also have performed well at document modeling, but the communication to the decoder through the representational bottleneck may have failed. In the latter case, we should be able to read the association of two source segments from the strength of the encoder's self-attentions between the two segments. We determine the association strength α of two aligned segments seg_1 and seg_2 by the area-normalized sum of encoder self-attention weights (enc_{ij} and enc_{ji}) between the two segments:

$$\alpha(\text{seg}_1,\text{seg}_2)=\frac{\sum_{i\in\text{seg}_1,j\in\text{seg}_2}\text{enc}_{ij}+\text{enc}_{ji}}{2*|\text{seg}_1|*|\text{seg}_2|} \tag{6.1}$$

[2]We choose $l=3$ as our window size.

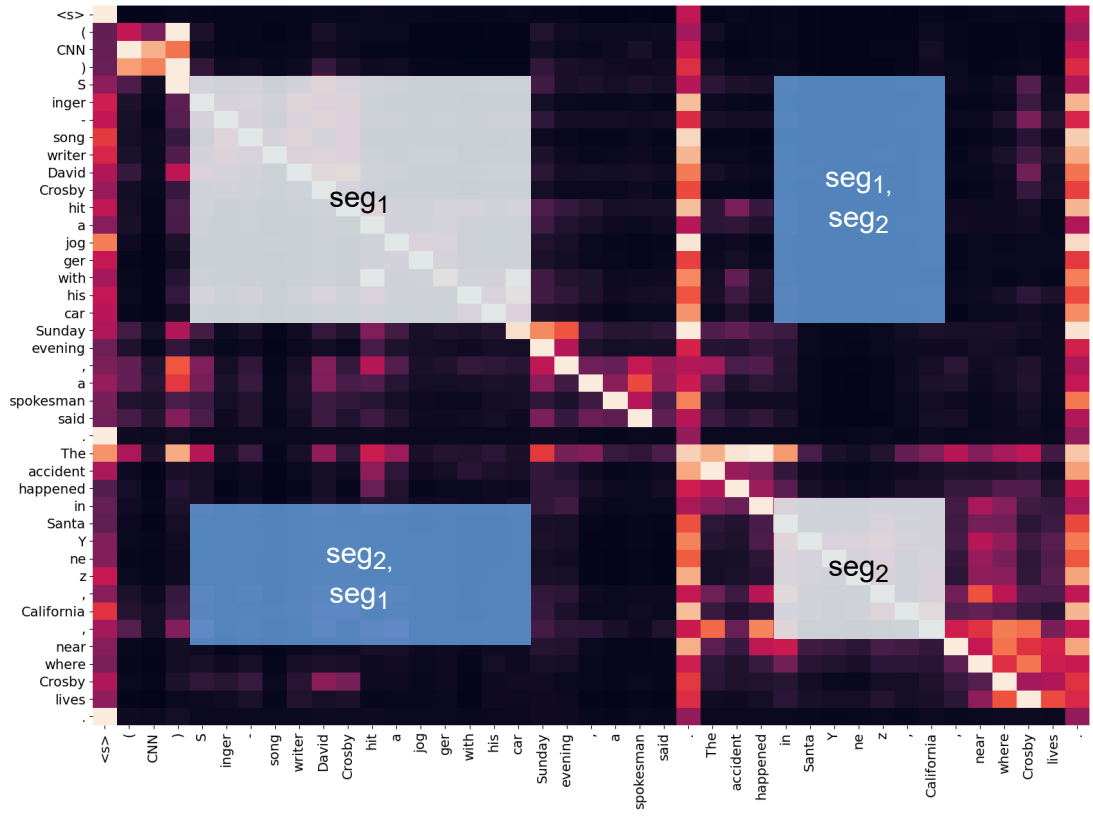


Figure 6.2: BART encoder self-attentions relate the aligned segments seg_1 and seg_2 of the source document (grey boxes) by their interactions (blue boxes). Only the first two source sentences are shown.

where i and j are the source indices of segments seg_1 and seg_2, and $|.|$ is the cardinality. Figure 6.2 visualizes the areas whose attention weights are summed with blue boxes. The score for a segment is the mean α to all other segments in its summary sentence. The higher the score, the higher our confidence in the two segments being semantically close, and therefore not intrinsic hallucinations. As an example, Figure 6.3 shows that the fourth segment has the smallest association strength to the other segments. Indeed, this is an intrinsic hallucination. It talks about the present state of the mansion, while the predicate concerns the past.

6.4.5 Classifying Unaligned Tokens

While unaligned tokens can still appear in the source document and result in an intrinsic hallucination, the prevalent errors for this set of tokens are extrinsic hallucinations. We found that generated summaries sometimes contain sentences entirely unrelated to the article, most likely an artifact of data collection. Our first score β_{align} is the fraction of the summary sentence tokens that are aligned.

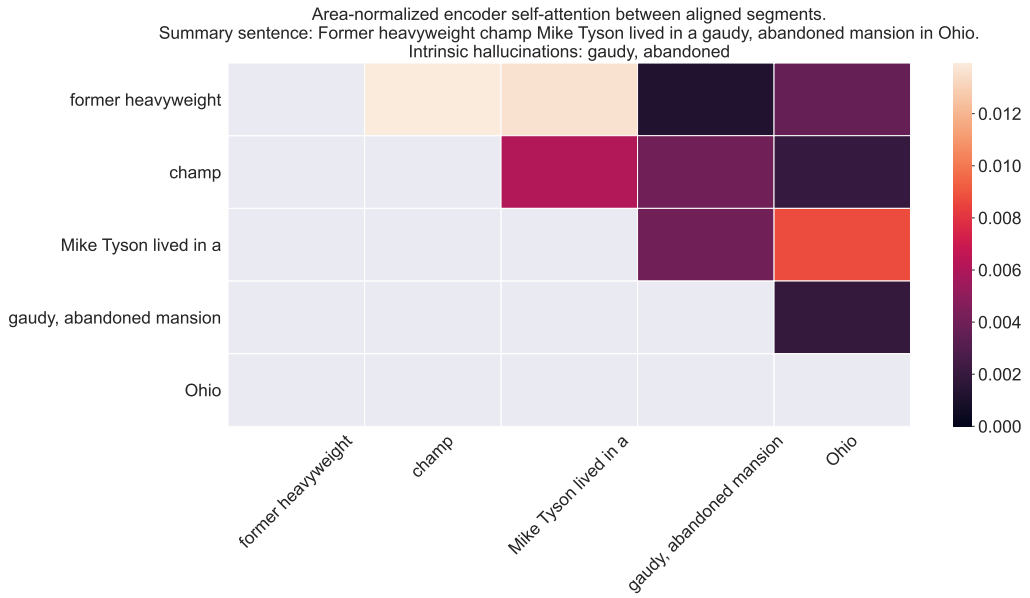


Figure 6.3: Association strength α between aligned segments. The intrinsic hallucinations in the fourth segment show the least interaction with other segments. Full example in Section 6.6.8.

For unaligned tokens in mostly-aligned sentences, we conjecture that generations by a strong language model fit in well (both syntactically and semantically) with the source document and the summary written so far, and thus should be expected by the model. In the opposite case, unexpected generations lead to a higher amount of surprisal. The expected surprisal of a language model can be quantified with the entropy of its next-word decoding probabilities (Meister et al., 2020). Figure 6.4 shows the decoding entropy of an example summary. We thus propose a second score β_{entropy} as the inverse smoothed decoding entropy:

$$\beta_{\text{entropy}}(t_i) = \frac{1}{H(t_i)+1} \tag{6.2}$$

with $H(t_i)$ the entropy of the next-word probability distribution of target token t_i.

Only the generation of the first token of an unexpected segment is surprising (as seen in Figure 6.4), and subsequent completions of the segment have high probability and low entropy. We therefore split a span of unaligned tokens into segments based on the decoding entropy. The construction is as follows: As long as the decoding entropy of the next token t_i decreases the mean decoding entropy of the current segment seg, it is added. Otherwise, a new segment is started.

$$\text{seg}' := \begin{cases} \text{seg} \cup t_i & \text{if } H(t_i) < \frac{\sum_{t_j \in \text{seg}} H(t_j)}{|\text{seg}|}, \\ t_i & \text{otherwise.} \end{cases} \tag{6.3}$$

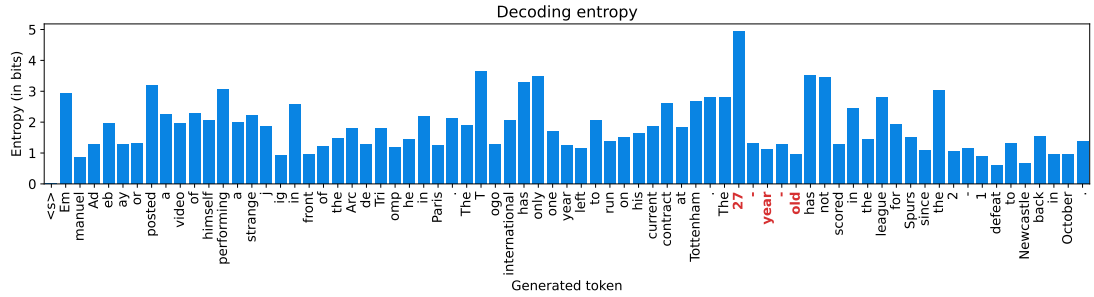


Figure 6.4: Summary containing an extrinsic hallucination (tokens in bold red). The decoding entropy of the first hallucinated token is high, and those of the subsequent tokens are low. We determine the hallucination score (Equation 6.2) of the entire segment (Definition in Equation 6.3) from its first token.

6.4.6 Converting Scores to Probabilities

All our faithfulness scores are nonnegative, and upper bounded by 1. A higher score means less chance of hallucination. We thus convert each faithfulness score s to a hallucination probability p by scaling and inverting it.

$$p = 1 - \frac{s - s_{\min}}{s_{\max} - s_{\min}} \tag{6.4}$$

where $s_{\min}$ and $s_{\max}$ are the minimum and maximum scores across the entire dataset. In an offline evaluation setting, one can compute all scores on a dataset first, and then get $s_{\min}$ and $s_{\max}$. For the online setting, these values have to be set. On our two datasets, we observe that the minimum and maximum values do not change much, so we expect the current values to transfer to new datasets. They are $[0, 0.02]$ for α, $[0.08, 0.71]$ for β_{entropy}, and β_{align} is already in the correct range.

BART-GBP. As we will see in the ablation study in the results in Section 6.6, the association strength α decreases the performance of our detection method. Our final model, *BART-GBP* (BART generation by-products), therefore only uses the β_{align} and β_{entropy} scores.

6.5 Experiments

We study CNN/DailyMail (Hermann et al., 2015), a summarization dataset known to be highly extractive (Grusky et al., 2018) and therefore less likely to contain a lot of hallucinations.

6.5.1 Datasets

Finding an existing dataset to evaluate our method is difficult since we need access to the model's attentions and decoding probabilities alongside the outputs.

FRANK. We repurpose FRANK, a factuality metric evaluation dataset (Pagnoni et al., 2021). It consists of 250 summaries from the CNN/DailyMail test set, obtained from SummEval (Fabbri et al., 2021). FRANK introduces a typology of factual errors, which we convert to hallucination annotations by using examples of predicate, entity, and circumstance errors as candidates for intrinsic hallucinations, and out-of-article errors as candidates for extrinsic hallucinations. Our publically available model version produces slightly different outputs from theirs, so we manually correct labels where the outputs differ. Our adapted dataset contains 57 hallucinated words (31 intrinsic, 26 extrinsic) which correspond to 0.4% of the 15,700 total words. At the sentence level, 3.5% contain at least one hallucinated word (31/897), while at the summary level, it is 9.2% (23/250).

TLHD-CNNDM. Since the number of hallucinations in FRANK is low, we additionally collect human annotations ourselves. We produce BART model outputs for the CNN/DailyMail test set (excluding the FRANK examples) by using the standard Hugging Face implementation with the default parameters. To arrive at an interesting dataset, we first rank summary sentences by two criteria: 1) the number of non-contiguous alignments to the source document found by lexical overlap, and 2) the number of words that do not appear in the source document. Both criteria are length-normalized. We pick the top 75 examples from both lists, arriving at 150 summary sentences. We then perform a human annotation as detailed below. Our dataset contains 299 hallucinated words out of a total of 2,100 (14.2%). Of those hallucinations, 51 are intrinsic, and 248 are extrinsic. Of the 150 sentences, 78 contain at least one hallucination (52%). The annotator agreement with the majority class (following Durmus et al., 2020) is 94.6%, and 73.9% and 86.3% for intrinsic and extrinsic hallucinations, respectively. We name our dataset *TLHD-CNNDM* (token-level hallucination detection for CNN/DailyMail).

TLHD-CNNDM human annotation details. Our human annotation was performed with 3 sets of 3 annotators, each annotating 50 examples. The full instructions include the definitions, an example annotation, and clarifying notes. They are as follows:

Hallucination detection
This study evaluates hallucinations in automatic summarization models. A hallucination is information that is not directly supported by the article that the model has to summarize.
Main question: Can the summary sentence in question be inferred directly from the article?
There are two types of hallucinations: intrinsic and extrinsic hallucinations. They are defined as follows (from Maynez et al., 2020):
Intrinsic hallucination: Combination of information from the article that does not follow from it
Extrinsic hallucination: Information not present in the article
Not a hallucination: Paraphrases, or information directly inferred from the article
Importantly, this is not a question of whether the summary is true or false, just whether it faithfully represents the information in the article.

The goal in this study is to annotate a summary sentence with intrinsic and extrinsic hallucinations, by copying the words that cannot be inferred from reading the article. Here's an example (the part in red is the annotation that you will do [your annotations can stay black]):

Example annotation
Article: Manchester City was defeated by Crystal Palace 2-1 at the Etihad Stadium on Sunday. Glenn Murray and Jason Puncheon scored the goals for Palace, while Yaya Toure was the only scorer for City. City's best striker Sergio Aguero was left on the bench for yet another game. The result is especially shocking when comparing the squad's total transfer fees: £40m pounds for Crystal Palace vs. £500m for Manchester City.
Full summary: Crystal Palace beat Manchester City 2-1 on Saturday. Yaya Toure was left on the bench, and Crystal Palace have spent £40m on transfer fees so far this season.
Sentence in question: Yaya Toure was left on the bench, and Crystal Palace have spent £40m on transfer fees so far this season.
Intrinsic hallucinations: Yaya Toure
Extrinsic hallucinations: so far this season
Explanation: It was Sergio Aguero that was left on the bench, not Yaya Toure (since he scored a goal, we know that he was playing). We're looking for a hallucination that is as small as possible, that's why we didn't mark "Yaya Toure was left on the bench", or "was left on the bench". For the extrinsic hallucination, there is no mentioning that the spending was for this season only. There is also a mistake in the first sentence of the summary (Saturday vs. Sunday in the article), but this is not the sentence in question, so we ignore it.

Notes

- If there are no hallucinations, leave the line blank.
- If there are multiple hallucinations in the sentence, separate them with a comma.
- Sometimes a sentence is not complete, or there are multiple sentences in one, but a period is missing to separate them. Just treat the "sentence in question" as if it were a single sentence. (These are artifacts of sentence splitting/the training data, which we do not evaluate here.)
- The examples below have a visual help: Text overlaps of more than two words between the sentence and the article are written in **bold** and numbered at the end, like this: [1]. This is just a help for you to find information faster, and does not mean the model copied the parts from there. Example: **Article: This year's harvest was**[1] especially rich **on apples.**[2] **Sentence: This year's harvest was**[1] high **on apples.**[2]
- Hint: Read the sentence in question first, and then look for the relevant information in the article.

6.5.2 Model Details

For generating our summaries, attentions, and decoding probabilities, we use the BART-large model finetuned on CNN/DailyMail from Facebook on Hugging Face[3] with its default hyperparameters. The model comprises 12 layers in the encoder as well as the decoder, 16 attention heads, a hidden dimension of 1024, and a maximum input length of 1024. For generation, BART uses beam search with a beam size of 4, a minimum summary length of 56 and a maximum of 142 tokens, a length penalty of 2.0, encouraging longer outputs, and trigram blocking (Paulus et al., 2018).

In generation with beam search, multiple beams are active at each generation step, but only one beam is eventually selected. We extract the attention and decoding probabilities of the correct beam by backtracking through the generation steps.

When inspecting cross-attentions, we found layers 10 and 11 (out of 12) to show the cleanest diagonal patterns (as presented in Figure 6.1). Other layers either have less focused attention, or they look at the previous token (mostly lower layers), the beginning-of-sequence token, or periods. We average the attentions from layers 10 and 11 of cross-attention to compute our initial alignment. We select the same layers for the encoder self-attentions.

As previously stated in Section 6.4, we use the top-4 cross-attentions to compute the initial alignment. For context voting, we employ a window size of 3, a minimum agreement of 50%, and a maximum of 10 voting rounds. For rescaling our scores, we use $[s_{\min}, s_{\max}]$ of $[0, 0.02]$ for α, and $[0.08, 0.71]$ for β_{entropy}. β_{align} does not need to be rescaled.

Since BART uses BPE tokenization (Sennrich et al., 2016), we aggregate the hallucination probabilities of subwords by taking the maximum. To compute the results in Section 6.6, we then vary the probability threshold for classifying a hallucination to get precision-recall and ROC curves. For this, we use scikit-learn[4] and make sure the convex hull is computed correctly.

6.5.3 Baselines

As baselines, we use four classes of models: lexical overlap, an entity-focused question-generation-answering model, a dependency entailment-based model, and a token-level classification model trained on synthetic data.

Lexical-n. This baseline lexically aligns the summary and the source document. It greedily adds the longest matching span, down to a span length of n. This baseline classifies all unaligned tokens as (presumably extrinsic) hallucinations. For aligned tokens, our most successful heuristic determines the hallucination probability for each aligned span as the fraction of aligned tokens that have an alignment in the same source sentence as the current

[3] https://huggingface.co/facebook/bart-large-cnn
[4] https://scikit-learn.org, version 1.1.1

span:

$$1 - \frac{|\text{tokens aligned to same source sentence}|}{|\text{all aligned tokens}|}. \tag{6.5}$$

FEQA. FEQA (Durmus et al., 2020) generates questions about the summary's entities, then tries to answer them from the source document. It then computes the token-level F1 score between the summary's text and the predicted text span from the source. Unmatched answers indicate hallucinations. We compute word-level probabilities by averaging the F1 scores of all spans the word is part of.

DAE. Dependency arc entailment (DAE) (Goyal and Durrett, 2020, 2021) decides from its dependency arcs whether the generated summary sentence is entailed by the source document. While DAE is technically a factuality detection method, we conjecture that hallucinations in the summary should not be entailed by the source document either. In footnote 6 of Goyal and Durrett (2021), the authors propose that a word is non-factual if any of its arcs are non-factual. We therefore compute word hallucination probabilities as the maximum probability of non-factuality of its dependency arcs. We use their model variant trained with entity-based synthetic data on CNN/DailyMail.

Fairseq. With the help of synthetic training data, where factual tokens have been automatically replaced with hallucinations, pretrained language models can be finetuned to directly predict a hallucination label for each input token (Zhou et al., 2021). We use the model finetuned on XSum and evaluate how it transfers to the CNN/DailyMail dataset. Since we compare to our unsupervised method, we leave retraining the model on CNN/DailyMail to future work. We evaluate both model settings, with and without access to the reference summary. We call this method *Fairseq* based on its GitHub repository name.

Method	Best F1	PR AUC	ROC AUC
FRANK			
FEQA*	0.0245	0.0062	0.3327
DAE*	0.0419	0.0157	0.7164
Fairseq w/o ref*	0.1651	0.0723	0.8129
Fairseq w/ ref*	0.0682	0.0232	0.7017
Lexical-1	0.1913	0.0677	0.8788
Lexical-2	0.0854	0.0335	0.8058
Lexical-3	0.0610	0.0268	0.7672
BART-GBP	**0.2778**	**0.1777**	**0.8934**
TLHD-CNNDM			
FEQA*	0.3156	0.2031	0.3899
DAE*	0.3167	0.1988	0.5803
Fairseq w/o ref*	**0.3957**	0.3255	**0.7375**
Fairseq w/ ref*	0.2672	0.1714	0.5521
Lexical-1	0.3937	0.2819	0.6846
Lexical-2	0.3535	0.2166	0.4802
Lexical-3	0.3025	0.1785	0.2599
BART-GBP	0.3806	**0.3502**	0.7332

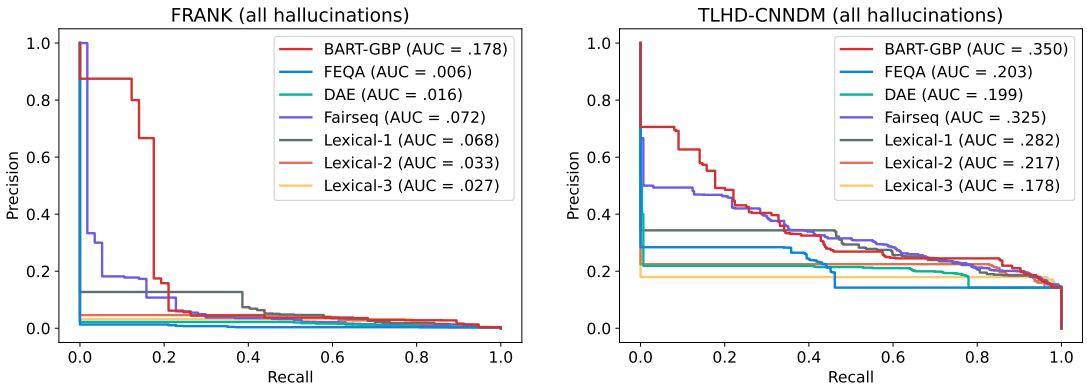
Table 6.1: Best F1 score on the precision-recall curve, the area under the precision-recall curve, and the area under the ROC curve. Methods marked with * require an additional model forward pass, which increases runtime and resource use.

6.6 Results

We use precision-recall curves to evaluate the hallucination detection methods. Precision-recall is the preferred metric when finding the instances of the positive class (hallucinations) has exceptionally high value compared to the instances of the negative class. Section 6.6.7 also shows ROC curves.

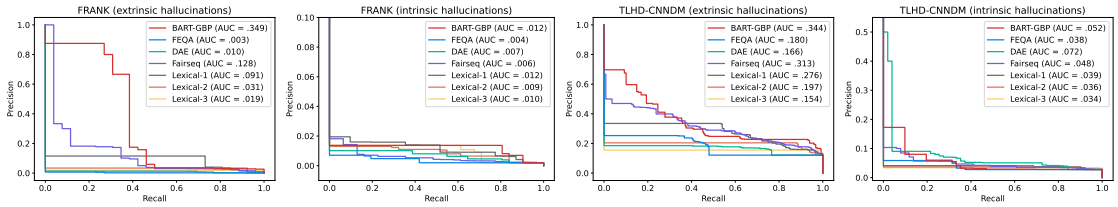
6.6.1 Precision-Recall Results

Our main result is shown in Table 6.1, which considers performance when classifying hallucinations of both intrinsic and extrinsic types. We present the best F1 score on the precision-recall curve, the area under the precision-recall curve, and the area under the ROC curve. Additionally, we show whether the method requires an additional model forward pass, which incurs a longer runtime and higher resource costs, by marking the respective methods (with *). BART-GBP performs best on the FRANK dataset and has the largest AUC for precision-recall on the TLHD-CNNDM dataset. For the other metrics, it is close behind the highest score, all while being completely unsupervised. Fairseq without access to the reference summary performs well on TLHD-CNNDM, but worse on FRANK. The setting without access to the reference



(a) FRANK (b) TLHD-CNNDM

Figure 6.5: Precision-recall curves for all hallucinations in the FRANK and TLHD-CNNDM datasets.



(a) Extrinsic (b) Intrinsic (c) Extrinsic (d) Intrinsic

Figure 6.6: Precision-recall curves for the label subsets of extrinsic and intrinsic hallucinations in the FRANK (6.6a, 6.6b) and TLHD-CNNDM (6.6c, 6.6d) datasets.

summary does better across all datasets and metrics, and is therefore reported from now on.

The precision-recall plots in Figure 6.5 give further details on the main result. BART-GBP manages to get high precision for the data points where it is most certain, something other methods struggle with. At higher levels of recall, the difficulty of the task leads to lower precision scores across all methods. The FRANK dataset, where only 0.4% of tokens are hallucinations, is very challenging (see Figure 6.5a). With 14.2% of positive labels, TLHD-CNNDM is less extreme but still proves to be difficult for all methods, as seen in Figure 6.5b.

6.6.2 Extrinsic Hallucinations

Figures 6.6a and 6.6c show the models' performance on the label subset of extrinsic hallucinations. To evaluate this subset, we remove data points that are gold intrinsic hallucinations in order to not unfairly penalize models for detecting those and vice versa for the evaluation of intrinsic hallucinations. Apart from BART-GBP and Fairseq, the Lexical-1 baseline manages to find some hallucinations. However, it does not provide a fine-grained trade-off between precision and recall, in contrast to BART-GBP.

Scores	FRANK	TLHD-CNNDM
α	0.0051	0.1440
β_{align}	**0.1993**	0.3260
β_{entropy}	0.0685	0.3198
β_{align}, β_{entropy}	0.1777	**0.3502**
α, β_{align}, β_{entropy}	0.0390	0.1687

Table 6.2: Ablation study for different combinations of scores. The metric is the area under the precision-recall curve. BART-GBP is the combination of β_{align} and β_{entropy}.

6.6.3 Intrinsic Hallucinations

As we can see from Figures 6.6b and 6.6d, finding intrinsic hallucinations proves to be very difficult for all methods. We therefore zoom in both graphs on the y-axis. BART-GBP performs well relative to the baselines. Notably, for the TLHD-CNNDM dataset, DAE manages to find some hallucinations at some of its highest probability selections but quickly diminishes at higher recall.

In summary, BART-GBP gets consistent and very competitive results in both datasets and on all label subsets, even while being an unsupervised method. The ROC curves in Figures 6.7 and 6.8 in Section 6.6.7 further confirm this finding.

6.6.4 Ablation Study

We are interested to see how each of our designed scores contributes to finding hallucinations. In Table 6.2, we show an ablation study with the area under the precision-recall curve as the performance metric. We see that of all individual scores, β_{align} performs best. Combining it with β_{entropy} (by taking the maximum of both probabilities for each token) further improves results on the TLHD-CNNDM dataset, but not on FRANK. α performs barely above a baseline that would classify all data points as hallucinations. This came as a surprise to us, as we expected α to perform better from the motivation in Section 6.4. Adding α to the β scores decreases performance drastically. This comes from the fact that our scores are not calibrated, so the distribution of each score will be different. As a result, when taking the max of multiple scores, one of them may dominate. When we plot a histogram of our scores' values, we see that this is the case for α, leading to such a performance deterioration in the case of combining all three scores. Since α on its own does not score well, we do not further calibrate our scores.

6.6.5 Maximum Possible Hallucination Recall

We motivated our approach by arguing that token-level methods are superior to entity-based question-generation-answering systems (like FEQA) or dependency arc entailment-based DAE. These methods may miss some hallucinated tokens as they only compute hallucination

Method	FRANK	TLHD-CNNDM
FEQA	38.60%	46.15%
DAE	80.70%	77.93%
Fairseq	100.00%	100.00%
Lexical-n	100.00%	100.00%
BART-GBP	100.00%	100.00%

Table 6.3: Maximum possible recall of FEQA (entity-based), DAE (dependency arc entailment), and the token-level methods Fairseq, Lexical-n and BART-GBP.

Score	FRANK			TLHD-CNNDM		
	All	Extrinsic	Intrinsic	All	Extrinsic	Intrinsic
Aligned (α)	50.88%	11.54%	83.87%	19.06%	11.29%	56.86%
Unaligned (β_{entropy})	52.63%	96.15%	16.13%	81.27%	88.71%	45.10%
Both (β_{align})	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%

Table 6.4: Maximum possible recall of aligned and unaligned token scores wrt. all, extrinsic, or intrinsic hallucinations.

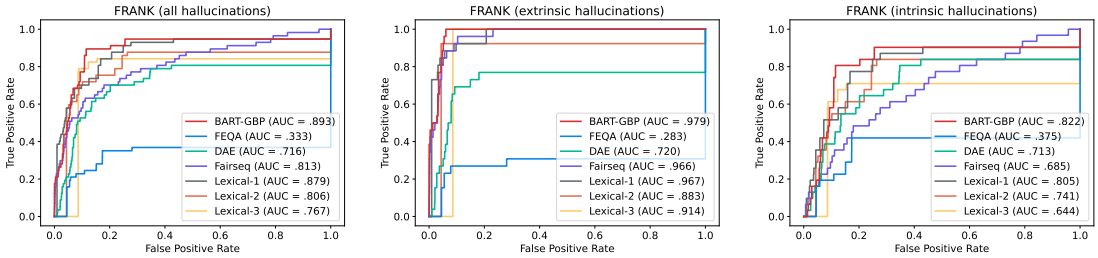
probabilities for a subset of all tokens. To verify how many there are, we analyze the recall each method achieves when it classifies all tokens that it considers as positives.

The results are shown in Table 6.3. The disadvantage for FEQA and DAE is substantial. FEQA classifies less than half of the tokens labeled as hallucinations in the FRANK and TLHD-CNNDM datasets. DAE is limited to a recall of around 80%, as it cannot detect tokens that are not part of one of the dependency arcs considered for entailment.

6.6.6 Maximum Recall of (Un)aligned Tokens

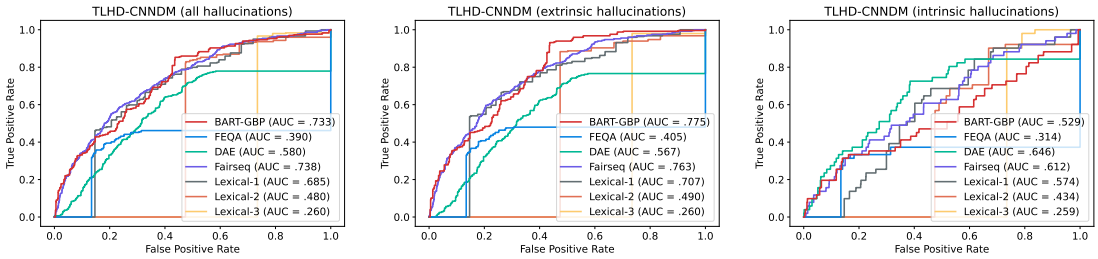
Aligning the summary with the source document forms the basis of our method. How many hallucinations are part of aligned spans, and how many are unaligned? We perform this analysis in Table 6.4. We can see that extrinsic hallucinations are mostly part of unaligned spans, which are scored by β_{entropy}. Intrinsic hallucinations in the FRANK dataset are mostly part of aligned spans, scored by α. In the TLHD-CNNDM dataset, however, intrinsic hallucinations are only part of aligned spans around half of the time.

Note that aligned and unaligned scores can add up to slightly more than 100%. This occurs when some BPE tokens of the same word are aligned and others are not (e.g. when a name appears together with a possessive *'s*).



(a) All hallucinations (b) Extrinsic hallucinations (c) Intrinsic hallucinations

Figure 6.7: ROC curves for hallucinations in the FRANK dataset.



(a) All hallucinations (b) Extrinsic hallucinations (c) Intrinsic hallucinations

Figure 6.8: ROC curves for hallucinations in the TLHD-CNNDM dataset.

6.6.7 ROC Results

Figures 6.7 and 6.8 show the ROC curves on the FRANK and TLHD-CNNDM datasets. There is a large label imbalance in both datasets, with the positive class only making up 0.4% of FRANK's labels and 14.2% of those in the TLHD-CNNDM dataset. This has to be considered when looking at these figures.

BART-GBP performs best on both datasets and label subsets, except for intrinsic hallucinations in the TLHD-CNNDM dataset in Figure 6.8c, where DAE and Lexical-1 perform better.

One thing that is easily visible from the ROC curves is the fraction of positive labels that can be discovered by a detection method. When a curve flattens out, it is no longer able to find more hallucinations without labeling all tokens as positive. This further highlights the strengths of the token-level methods BART-GBP and Lexical-n.

6.6.8 Hallucination Examples

We present two examples of hallucinations, one of intrinsic hallucination from the FRANK dataset, and one of extrinsic hallucination from the TLHD-CNNDM dataset. In the former example, Mike Tyson's mansion is now in a gaudy, abandoned state, but was not while he still lived in it. In the latter example, the name of the stadium (Old Trafford) is never mentioned in the article, so it is an extrinsic hallucination. As an aside, factuality cannot be determined,

since the article only talks about a "meeting" of the two teams and does not mention the home team.

Intrinsic hallucination from FRANK.
Article: (CNN)A trip to a former heavyweight champ's gaudy, abandoned mansion. The tallest and fastest "giga-coaster" in the world. A dramatic interview with a famed spiritual leader – and the tearful reaction by one of his former students. These are some of the best videos of the week: In the 1980s and '90s – before he moved to Vegas and started keeping tigers as pets – former heavyweight boxer Mike Tyson lived in a Southington, Ohio, mansion. The home featured an indoor swimming pool, a marble-and-gold Jacuzzi (with mirrored ceiling, naturally) and an entertainment room large enough for small concerts. Tyson sold the house in 1999; it's due to become, of all things, a church. The video can be seen at the top of this story. Not a fan of roller coasters? You may want to skip the next video – but for the rest of us, the thrill of watching is the next best thing to being there. The Fury 325 can be found at Carowinds amusement part in Charlotte, North Carolina. Watch the video: In a CNN exclusive, Alisyn Camerota looked into allegations that Bikram yoga creator Bikram Choudhury sexually assaulted six former students. "He's a person who's based a lot of truths on a lot of lies," said Sarah Baughn, who alleges that Choudhury sexually assaulted her. Watch the video: CNN's Karl Penhaul spoke to a shepherd who witnessed the final seconds of Germanwings Flight 9525, which crashed in the French Alps last week. "I saw the plane heading down along the valley and I said, 'My God, it's going to hit the mountain,' " Jean Varrieras told Penhaul. "I ducked my head. ... Then after that, I saw the smoke." Watch the video: Magician and comedian Penn Jillette was part of a panel speaking to CNN's Don Lemon about the controversial Indiana religious freedom law. Jillette, an avowed atheist and libertarian, noted "we are not talking about forcing people to engage in gay sex, or even endorse gay sex." His provocative opening led to an energetic back-and-forth with the Alliance Defending Freedom's Kristen Waggoner and the ACLU's Rita Sklar. Watch the video: A professor of physics at a British university asked 100 people to create a composite with facial features they thought were beautiful – and then asked another 100 to rate their attractiveness. You'll never guess what celebrities best fit the model. Watch the video:
BART summary: Former heavyweight champ Mike Tyson lived in a gaudy, abandoned mansion in Ohio. CNN's Karl Penhaul spoke to a shepherd who witnessed the final seconds of Germanwings Flight 9525. Penn Jillette was part of a panel speaking to CNN's Don Lemon about the controversial Indiana religious freedom law.
Intrinsic hallucinations: gaudy, abandoned

Extrinsic hallucination from TLHD-CNNDM.
Article: Gareth Barry has advised his Everton team-mate Ross Barkley against

moving to Manchester City at this young stage of his career. Barry speaks from experience having spent four seasons at the Etihad before arriving on Merseyside and the veteran midfielder believes it is still too early for the 21-year-old to decide on his future. Ahead of the Toffees meeting with Manchester United on Sunday, Barry told the Mirror: 'Personally, I think he's still too young to make that move. Ross Barkley's rise to stardom has seen him repeatedly linked with Premier League champions Man City . Everton team-mate Gareth Barry has advised the youngster not to leave Goodison too soon . 'He's still learning the game. He's got the right manager here to push him to the next level. 'As soon as he reaches that next level, then there's another decision to be made. At the moment, I think it's too early.' And asked if considered the Premier League champions to be a graveyard for young talent, Barry added: 'I think so, yeah.' Barkley has overcome his early season struggles to play an influential role in Everton's recent revival and Barry believes the youngster he mentors daily can achieve anything he wants in the game. The 21-year-old signs autographs for fans after coming through a difficult start to the season . Veteran midfielder Barry spent four seasons at City before being found surplus to requirements . 'I sit next to him in the changing room at the training ground. I speak to Ross quite often,' said Barry. 'You feel sorry for him sometimes because the expectation is getting thrown on to his shoulders – people are expecting of him, week in, week out, goals and assists. 'That hasn't happened, but at the same time he's still improving as a player and growing in maturity. 'His ability and his strengths are there for everyone to see, he can go on and be a top top player.'
BART summary: Ross Barkley has been linked with a move to Manchester City. Everton team-mate Gareth Barry believes it is too early for the 21-year-old to leave Goodison Park. Barry spent four seasons at the Etihad before arriving on Merseyside. Everton face Manchester United at Old Trafford on Sunday.
Extrinsic hallucinations (last sentence): at Old Trafford

6.7 Related Work

Several different methods have been proposed to detect hallucinations. Specialized decoding strategies are used to nudge the model to stay closer to the source vocabulary (Aralikatte et al., 2021) or its entities (Narayan et al., 2021). Filippova (2020) determine the degree of hallucination from the differences in probabilities assigned by a conditional and an unconditional language model.

In the related area of factuality detection, Cao et al. (2022) use the same idea to identify hallucinated but factual summaries. Entailment-based classifiers are used to evaluate a summary's factuality at the level of text or dependency arcs (Falke et al., 2019; Goyal and Durrett, 2020). It is also common to create synthetic data for a classifier by corrupting the input, for hallucinations (Zhou et al., 2021) as well as factuality (Cao et al., 2020; Kryscinski

et al., 2020). However, the error distributions obtained synthetically can differ from those of models (Goyal and Durrett, 2021). More types of factuality errors are identified in Pagnoni et al. (2021) with detailed human annotation, finding discourse and semantic frame errors. These detection methods can be used to identify mistakes or rerank multiple outputs (e.g. Ladhak et al., 2022).

6.8 Conclusion

We have presented BART-GBP, a method to detect hallucinations from the by-products of summary generation of a BART abstractive summarization model, trained and evaluated on CNN/DailyMail. We first aligned the segments of the summary and source document using cross-attentions, and then used encoder self-attentions and decoding probabilities to detect intrinsic and extrinsic hallucinations, respectively. This happens with minimal computational overhead, compared to prior work that uses external models that require an additional model forward pass. Our evaluations show that this is a difficult task, and especially intrinsic hallucination detection needs to be addressed in future work. We hope to contribute to this endeavor with our method and our token-level annotated dataset, TLHD-CNNDM.

6.8.1 Hallucination Definition

The results in this chapter are limited by several factors. Firstly, the definition of what constitutes a hallucination is neither set in stone, nor a mathematical construct, and therefore open to interpretation. We experienced this first-hand from the feedback of our annotators. This makes the task of teaching a model to identify hallucinations all the more difficult, and the gap to optimal performance in the results (for all methods) makes this visible.

Another limitation is given by the model under study. We already mentioned in Section 6.4 that the interpretability of attention patterns is a debated topic in the research community. A model trained to faithfully *explain* its decisions would be even better suited to perform this kind of analysis.

6.8.2 Transfer to Other Models

While we do not assume that our method transfers easily to some attention-based RNN architectures, we saw indications that it could transfer to other Transformer-based summarization models. In initial experiments, we used BERTSUMABS (Liu and Lapata, 2019b), which shows very similar cross-attention patterns (see Figure 6.9). There are some small differences, however. BERTSUMABS puts its maximum attention weight to the copied word more often, but still shows a lot of attention to CLS/SEP tokens in the source and BOS/EOS tokens in the summary. Additionally, the tokenization is different which can have an impact on the alignment stage. In BART, for example, the same word can be tokenized in different ways when it is preceded

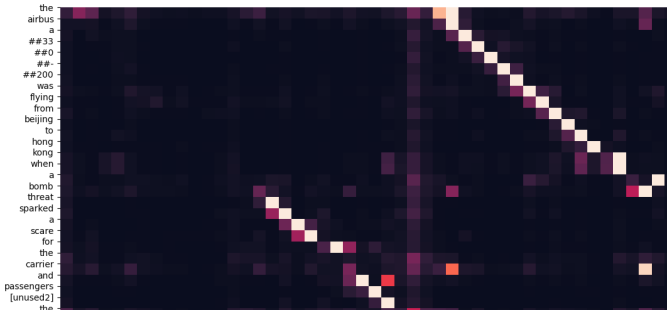


Figure 6.9: BERTSUMABS cross-attention patterns are very similar to those of BART, both Transformer-based summarization models.

by the BOS token, whitespace, or punctuation. This sometimes prevented our method from aligning the same word due to unmatched tokens.

6.8.3 Transfer to other datasets.

We do not expect these results to transfer to datasets that have a large percentage of hallucinations, i.e. XSum. We are not aware of other datasets with those same hallucination characteristics. However, we expect that other summarization datasets could benefit from our method, especially those that are similarly extractive as CNN/DailyMail. The scoring range to convert scores into probabilities may have to be recomputed.

6.8.4 Prevalence of sports topics in hallucinations.

The prevalence of sports topics in CNN/DailyMail hallucinations hints at divergence issues between the source and reference (Wiseman et al., 2017; Dhingra et al., 2019; Kryscinski et al., 2019) for these topics: True additional information (such as standings) is added by the author/editor. It is interesting to note that while models trained on XSum learn to hallucinate consistently, CNN/DM models learn to hallucinate on sports topics. While removing hallucinations from the training data could address hallucinations, this seems infeasible, and detecting hallucinated model outputs is a more practical approach.

7 Semi-Structured Annotations for Interpretation

Chapter Summary

A wide variety of tasks have been framed as text-to-text tasks to allow processing by sequence-to-sequence models. We propose a new task of generating a semi-structured interpretation of a source document. The interpretation is semi-structured in that it contains mandatory and optional fields with free-text information. This structure is surfaced by human annotations, which we standardize and convert to text format. We then propose an evaluation technique that is generally applicable to any such semi-structured annotation, called *equivalence classes evaluation*. The evaluation technique is efficient and scalable; it creates a large number of evaluation instances from a comparably cheap clustering of the free-text information by domain experts. For our task, we release a dataset about the monetary policy of the Federal Reserve. On this corpus, our evaluation shows larger differences between pretrained models than standard text generation metrics.

Publication in this Chapter

This chapter builds on the material in our publication:
Marfurt, A., Thornton, A., Sylvan, D., van der Plas, L., and Henderson, J. (2022). A Corpus and Evaluation for Predicting Semi-Structured Human Annotations. In *Proceedings of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)* (pp. 262-275).
Best resource paper award.
PDF: https://aclanthology.org/2022.gem-1.22.pdf
Code, data, and models: https://github.com/idiap/semi-structured-annotations

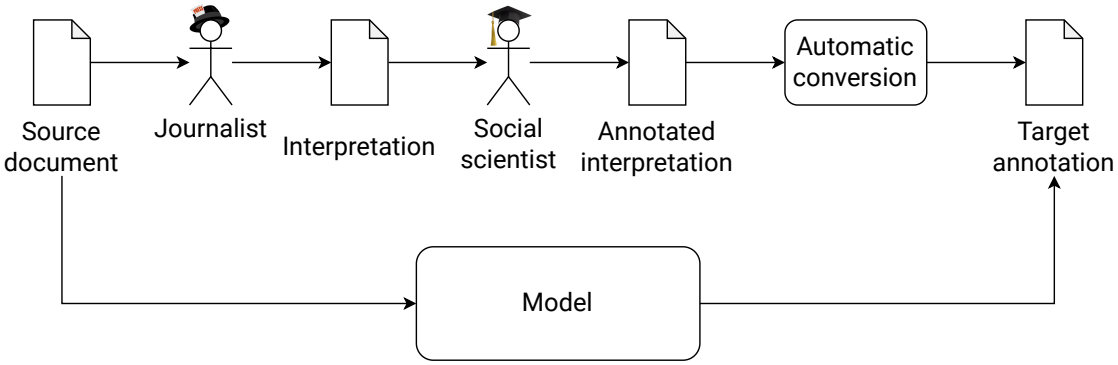


Figure 7.1: Our proposed interpretation task. A journalist creates an interpretation of a source document. A social scientist extracts and categorizes the relevant parts of the interpretation by means of annotation, which is converted into a text-only format. The model learns the interpretation process by directly predicting the target annotation from the source document.

While the previous chapters focused on improving individual subtasks of summarization, the current chapter addresses interpretation. On top of the ability to extract salient information and abstract, interpretation requires speculating on the reasons and motives of statements. In this chapter, we train models to interpret with the help of semi-structured annotations of the aspects of an interpretation. We also devise an efficient specialized evaluation from annotated examples. Even though the available dataset is small, we show that generative models are able to pick up on the details of the task.

7.1 Learning to Interpret

General-purpose sequence-to-sequence models have achieved impressive results on conditional text generation (Radford et al., 2019; Brown et al., 2020), machine translation (Liu et al., 2020; Xue et al., 2021), and text summarization (Lewis et al., 2020; Zhang et al., 2020a). This has led to their application to ever more tasks; as long as the task can be formalized in a text-to-text format, it can be processed by these models (Raffel et al., 2020).

We apply sequence-to-sequence models in a different setting: documents interpreting other documents. This phenomenon is pervasive in our daily lives, be it a critic reviewing a play or book, a website presenting highlights of a travel guide, or, as in this chapter, a journalist writing an article about an organization's press release.

For social scientists, these reviews or articles present an interesting subject of study; they surface the author's interpretation of the original source document. With the tool of human annotations, domain experts can extract the core constituents and surface implicit information in these various interpretations to make them comparable. We use these annotations to teach a model to interpret (see Figure 7.1).

The annotation provides structure to the model during training and generation. The different

aspects of interpretation are clearly marked with category labels, activating the relevant knowledge from training when the model generates the free-text information of a category. We conjecture that this technique of activating a context is a natural fit for language models that condition on previous words to generate the next. In the following section, we further describe the annotations, task, and dataset.

7.2 Semi-Structured Annotations

To guide the model during the interpretation process, domain experts annotate the different aspects of an interpretation, such as motive or temporal scope. First, the available operations for annotation have to be specified.

7.2.1 Standardizing Annotations

We aim to standardize the human annotations into a general but flexible semi-structured format which should make it possible for NLP models to process them. In order to do so, we first have to define the possible annotation operations.

Our annotations are created from two operations: (1) marking spans with a label in order to categorize them, and (2) optionally commenting on a marked span to give context, paraphrase, or make implicit information explicit.

7.2.2 Converting Annotations to Text

We convert annotations into a text-only format by inserting category-specific start and end tokens for each marked span. Overlapping or fully contained spans are allowed. We include the comments by adding them in parentheses (imitating a similar use in natural language) at the end of the respective marked span and before the category end token. An example annotation transformed to text format is shown in Figure 7.2.

7.2.3 Sequence-to-sequence Task

We propose the task of generating the interpretations, including the human annotations, from the source documents. This task can be formalized as a sequence-to-sequence generation task, with pairs of a single source document x_i and one or more target annotations y_{ij} in text format, with $1 \leq j \leq m_i$. The multiple target annotations are equivalent to multiple references in traditional text generation tasks, i.e. they are all equally valid solutions to the task. In total, there are n source documents and $m = \sum_{i=1}^{n} m_i$ targets.

The targets y_{ij} contain marked spans of categories c from a predefined set of categories C. Some categories occur in every target, and some are optional, as illustrated in the paragraph *Annotation Categories* below.

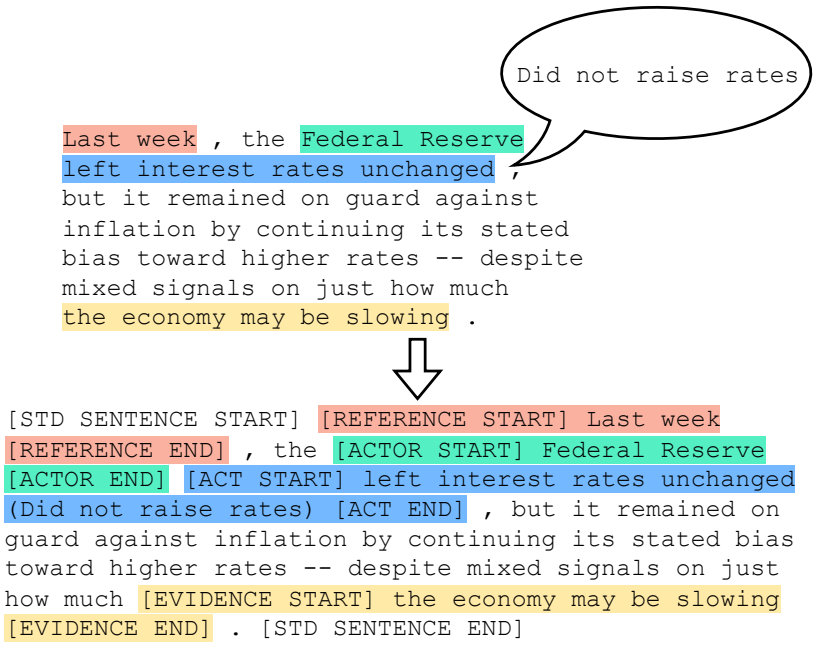


Figure 7.2: Example of the automatic conversion of an annotated interpretation into text format.

7.2.4 The FOMC Dataset

We now present our dataset constructed according to the guidelines above. The source documents and targets were selected and annotated by domain experts. They are on the topic of the monetary policy of the Federal Open Market Committee (FOMC) of the Federal Reserve, the central bank of the United States of America, in the years from 1967 to 2018. The source documents are policy announcements of the FOMC such as press releases, speeches, testimonies, Q&A sessions, or meeting minutes. The targets are sentences from news articles of the New York Times that conform to the requirements (see *Annotation Categories* below). The first example from the FOMC test set is shown below. It includes the source document filtered with FilterBERT, the target annotation, and BART's prediction. At the end, we show the generation scores for BART's prediction.

> **Filtered source document:** FEDERAL RESERVE press release For Use at 4:30 p.m. August 22, 1986 The Federal Reserve Board and the Federal Open Market Committee today released the attached record of policy actions taken by the Federal Open Market Committee at its meeting on July 8-9, 1986. Such records for each meeting of the Committee are made available a few days after the next regularly scheduled meeting and are published in the Federal Reserve Bulletin and the Board's Annual Report. The summary descriptions of economic and financial conditions they contain are based solely on the information that was available to the Committee at the time of the meeting. Attachment RECORD OF POLICY

ACTIONS OF THE FEDERAL OPEN MARKET COMMITTEE Meeting Held on July 8-9, 1986 Domestic policy directive The information reviewed at this meeting indicates that economic activity has expanded at a relatively slow pace recently. The intermeeting range for the federal funds rate was reduced to 5 to 9 percent. Other interest rates rose early in the period but then retreated amid signs of weakness in the economies of the United States and some of its major trading partners, renewing expectations of a discount rate cut in the near future. Since the May meeting short-term market rates had declined 10 to 40 basis points on balance. In their discussion of policy implementation for the weeks immediately ahead, Committee members took account of the likelihood that the discount rate would be reduced within a few days after the meeting. Against the background of sluggish expansion in economic activity and a subdued rate of inflation, most of the members believed that some easing was desirable and they indicated a preference for implementing the easing, at least initially, through a lower discount rate rather than through open market operations. In one view, a cut in the discount rate might need to be accompanied by some increase in the degree of pressure on reserve positions, pending evaluation of further economic and financial developments. The reduction was viewed as a technical adjustment that would provide a more symmetrical range around a lower federal funds rate that could be expected to emerge following the anticipated reduction in the discount 7/8-9/86 - 18 rate. Most short-term interest rates have declined on balance since the May 20 meeting of the Committee. In the implementation of policy for the immediate future, the Committee seeks to decrease somewhat the existing degree of pressure on reserve positions, taking account of the possibility of a change in the discount rate.

Target annotation: [STD SENTENCE START] Policymakers at the [ACTOR START] Federal Reserve [ACTOR END] [ACT START] decided at their July meeting to loosen credit conditions (Loosened monetary policy) [ACT END] [MOTIVE START] in an effort to stimulate the sluggish economy [MOTIVE END], according to [REFERENCE START] minutes [REFERENCE END] of the meeting released today. [STD SENTENCE END] [STD SENTENCE START] Members of the [ACTOR START] Federal Open Market Committee (Fed / FOMC) [ACTOR END] [REFERENCE START] voted [REFERENCE END] 10 to 1 to follow a strategy that would push interest rates lower, [ACT START] despite [ATTRIBUTION START] objections from one member (Should not loosen monetary policy) [ACT END] (One member of the FOMC) [ATTRIBUTION END] that [EVIDENCE START] such a course might threaten renewed inflation later [EVIDENCE END]. [STD SENTENCE END] [STD SENTENCE START] Thomas C. [ATTRIBUTION START] Melzer [ATTRIBUTION END], president of the St. Louis [ACTOR START] Federal Reserve Bank [ACTOR END], [ACT START] cast the single dissenting vote (Should not loosen money supply) [ACT END]. The minutes said Mr. Melzer [REFERENCE START] expressed concern [REFERENCE END] that [EVIDENCE START] looser Fed controls could initiate renewed inflation

[EVIDENCE END] and [EVIDENCE START] weaken the dollar on foreign exchange markets [EVIDENCE END]. [STD SENTENCE END]

BART prediction: [STD SENTENCE START] The [ACTOR START] Federal Reserve's Open Market Committee (Fed) [ACTOR END] [ACT START] voted unanimously at its July 8-9 meeting to ease monetary policy further (Might cut rates, in future) [ACT END], according to [REFERENCE START] minutes [REFERENCE END] of the session released today. [STD SENTENCE END]

ROUGE-1/2/L (including category markers): 31.68/17.00/28.71
ROUGE-1/2/L (excluding category markers): 30.38/15.38/27.85
BERTScore: 27.29
Novel bigrams: 84.38%
Closed correctly: 100.00%

Data collection. Domain experts searched the New York Times archives for articles on the monetary policy of the Federal Reserve. Candidate articles were searched for sentences that contain all mandatory categories described below. If a sentence is found, it is annotated by highlighting the categories and adding comments. All annotations are validated by a senior domain expert . Sentences from the same article referencing the same source document are collected in a single target annotation. If multiple articles reference the same source document, one target annotation is created per article.

Annotation Categories. A selected sentence is called a *standardized sentence* in the corpus terminology. The mandatory and optional categories, as well as their purpose, are listed below:

- **Standardized sentence:** Mandatory. Marks the start and end of a target sentence.
- **Act:** Mandatory. Most often contains a comment. Marks an action (or non-action) on monetary policy. Example: "left interest rates unchanged (Did not raise rates)".
- **Actor:** Mandatory. Marks the entity performing the act. By design, this is exclusively the Federal Reserve or FOMC. Example: "Fed".
- **Reference:** Mandatory. Provides a link to the source document, which can be opaque in the article, e.g. saying that something happened yesterday. That source is systematically tracked down by the domain experts. Example: "yesterday's meeting".
- **Attribution:** Optional. Marks the individual advocating for the Federal Reserve to perform a certain action. Example: "Greenspan".
- **Motive:** Optional. Can appear multiple times. States the goal of an act. Example: "to fight inflation".

	Train	Valid	Test
Source documents	1342	167	169
Target annotations	3246	364	380
Min targets/source	1	1	1
Median targets/source	1	1	1
Mean targets/source	2.42	2.18	2.25
Max targets/source	36	17	16

Table 7.1: Number of examples in each split in the FOMC dataset.

Count	Source documents	Targets
(Std) Sentences	262.6 (± 688.6)	1.6
Words	6054.1 (± 12639.4)	123.6
Start/end tokens	-	18.2
Total tokens	6054.1 (± 12639.4)	141.8

Table 7.2: Mean length of source documents and target annotations in the FOMC dataset.

- **Evidence:** Optional. Can appear multiple times. States an observation, e.g. about the current economic state, that served as an incentive for the act. Example: "high oil prices".
- **Scope:** Optional. Marks the temporal scope of an act. Example: "by the end of the year".

Dataset statistics. We now show dataset statistics. First, the number of examples in each split (80%/10%/10% for train/validation/test) is presented in Table 7.1. Second, the number of tokens in source and target texts is shown in Table 7.2.

Filtering source documents. As is evident from Table 7.2, the source documents are generally very long. In contrast, the maximum number of input tokens that state-of-the-art models are pretrained on, lies between 512 in BERT (Devlin et al., 2019) and 1024 in BART (Lewis et al., 2020). This limitation is due to the quadratic complexity of self-attention in the Transformer architecture (Vaswani et al., 2017) and its resulting strain on computational resources.

As a consequence, there are two ways to define the prediction task for the FOMC dataset. The first one is to condition on the full-text document but devise models capable of handling very long inputs, such as adding a filtering module (used below) or using appropriate architectures such as the Longformer (Beltagy et al., 2020). The second option is to condition on a specific filtering of the source documents which reduces them to a length that can be processed by the chosen model. Alongside the data, we provide a script that allows for selecting sentences from the source document, while satisfying the length restriction for a given tokenizer from the Hugging Face transformers library (Wolf et al., 2020). The selection logic can be set to either

pick sentences from the top of the source document (*Lead* strategy) or to use an oracle that greedily picks sentences that maximize the length-normalized ROUGE-2 recall gain (*Oracle* strategy).

7.3 Evaluation Modes

We train generative models on the sequence-to-sequence task on the FOMC dataset. We evaluate the trained models in two different modes, corresponding to their usage by the domain experts. We either generate the full interpretation or we complete a marked span given a prompt, in both cases conditioning on the source document.

7.3.1 Full Prediction

Predicting the whole interpretation from the source document corresponds to the standard evaluation setting in text generation tasks, such as abstractive summarization. The task remains the same as during training, as described in Section 7.2.3. We therefore measure the performance with standard text generation metrics.

7.3.2 Completion of a Marked Span

In a second evaluation mode, domain experts design prompts and ask a trained model to complete the interpretation. This allows the users to query the model with hypothetical and even counterfactual prompts. When studying the examples with which the experts queried the models, it became clear that the completion of a marked span of an annotation category is of special importance. In the following, we introduce the equivalence classes evaluation, a recipe for creating targeted evaluations of these completions with limited manual annotation effort.

7.4 Equivalence Classes Evaluation

We propose the equivalence classes evaluation as an efficient way of generating a large number of evaluation instances from domain experts' knowledge of the individual annotation categories.

7.4.1 Definition

An evaluation selects a category c that it wants to evaluate, which in turn consists of 2 or more equivalence classes. The members of an equivalence class are marked spans of category c in the dataset. Members of the same equivalence class are semantically interchangeable in the target annotation, with respect to the objective of the evaluation. The members of all

```
Evaluation: Act type
Category: Act
Equivalence class 1:
  - left interest rates unchanged (Did not raise rates)
Equivalence class 2:
  - decided to raise interest rates (Did raise rates)
  - voted to raise rates (Did raise rates)
Equivalence class 3:
  - lowered interest rates a quarter point (Cut rates)
```

Figure 7.3: Definition of an evaluation with its equivalence classes.

y_{prefix}	[REFERENCE START] Last week [REFERENCE END] , the [ACTOR START] Federal Reserve [ACTOR END] [ACT START]
$a^{(\text{pos})}$	left interest rates unchanged (Did not raise rates) [ACT END]
$a^{(\text{neg})}$	decided to raise interest rates (Did raise rates) [ACT END]

Figure 7.4: Equivalence classes evaluation instance with prefix y_{prefix}, a positive continuation $a^{(\text{pos})}$ and a negative continuation $a^{(\text{neg})}$.

equivalence classes must be syntactically interchangeable, such that replacing one for the other still results in a grammatically correct sentence. An example is given in Figure 7.3.

7.4.2 Creating Evaluation Instances

Evaluation instances are then created by searching target annotations in the validation/test set for a member of an equivalence class. If one is found, an evaluation instance is created consisting of (1) the prefix y_{prefix} up until the selected span, (2) the selected span $a^{(\text{pos})}$ as the true (positive) continuation, and (3) a randomly selected span $a^{(\text{neg})}$ from a different equivalence class as the false (negative) continuation. $a^{(\text{neg})}$ is chosen by uniformly sampling a negative equivalence class, and then uniformly sampling one of its members. An example is shown in Figure 7.4, where $a^{(\text{pos})}$ is in equivalence class 1 of the example evaluation in Figure 7.3, and $a^{(\text{neg})}$ has been sampled as the first member of equivalence class 2. Any other member of equivalence classes 2 or 3 could have been chosen as well.

For a single match of a positive span in the evaluation set, one can create a large number of evaluation instances by sampling negative continuations without replacement.

Optionally, the positive span $a^{(\text{pos})}$ can be replaced by a different member of the same equivalence class (for equivalence classes with more than one member). This can help mitigate lexical inaccuracies that can arise from replacing a span with another, which otherwise only exist for $a^{(\text{neg})}$.

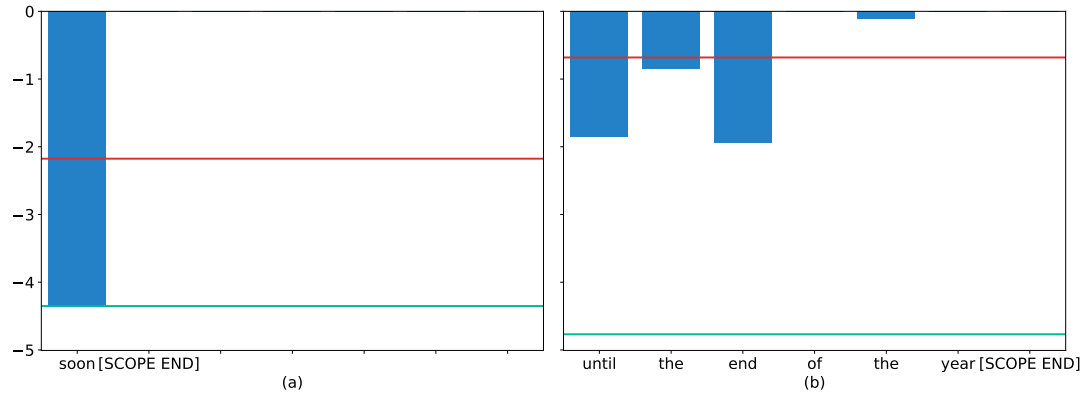


Figure 7.5: BART's log probabilities of a short (a) and long (b) continuation. The green line corresponds to the sum of log probabilities, the red line normalizes by the continuation's length.

Relation to Pattern-Exploiting Traning. In Schick and Schütze (2021a), Pattern-Exploiting Training (PET) is introduced. The concept of verbalizers is similar to equivalence classes. In their work, verbalizers are manually predefined single tokens that represent a class label.[1] Our equivalence classes consist of expert-selected multi-word spans from the data, which each represent the concept of their equivalence class. Equivalence classes are multi-faceted: They determine both a semantic concept and a grammatical structure and are always defined with respect to a certain aspect under evaluation.

7.4.3 Model Evaluation

To evaluate the generative model, we obtain the probability p_θ it assigns to $a^{(\text{pos})}$ and $a^{(\text{neg})}$ by getting its next-token probabilities given the source document x and prefix y_{prefix}. We apply teacher forcing and obtain the probabilities autoregressively, extending the prefix with the previous token at each turn. The probability of the entire span is computed as

$$p_\theta(a) = \prod_{i=1}^{l} p_\theta(a_i | x, y_{\text{prefix}}, a_{<i}) \tag{7.1}$$

where $a \in \{a^{(\text{pos})}, a^{(\text{neg})}\}$, and l is the length of a. The model solves an instance correctly if $p_\theta(a^{(\text{pos})}) > p_\theta(a^{(\text{neg})})$.

If the lengths of $a^{(\text{pos})}$ and $a^{(\text{neg})}$ are substantially different, the comparison of their p_θ could be determined by the difference in length. In natural language generation, it is common to compute p_θ as the sum of log-probabilities, normalized by the sequence length (Cho et al., 2014a). In Figure 7.5, we show BART's log probabilities of a short and long continuation, with the unnormalized sum as the green line, and the normalized sum as the red line. We see that

[1] In their follow-up work, they extend verbalizers to multiple tokens (Schick and Schütze, 2021b).

the long continuation ends with many high-probability tokens that have an outsized impact on the normalized log probability. We therefore decide to use unnormalized probabilities (Equation 7.1), but avoid introducing a length bias when sampling $a^{(\text{neg})}$ by restricting its word count to be within 2 of $a^{(\text{pos})}$.

7.4.4 In-Depth Analysis

The equivalence classes evaluation also allows for in-depth error analysis. First, we can test specific properties for a category, such as how well the model handles negation in acts. Second, we can break down an evaluation's score by combinations of equivalence classes, and identify the hardest combinations for the model. We show examples of such analyses in Section 7.6.3.

Data augmentation. As an added benefit, equivalence classes give rise to a simple training data augmentation method. We create additional training examples from equivalence classes by exchanging the ground-truth highlighted span with a different one from the same equivalence class. We postpone testing the efficacy of this data augmentation method to future work.

7.5 Experiments

In our experiments, we use the FOMC dataset described in Section 7.2.4.

7.5.1 Equivalence Classes

We propose equivalence classes from category annotations and validate them by our senior domain experts from Section 7.2.4. We create an evaluation for each of the following 5 categories: act, attribution, motive, evidence, and scope. We add one evaluation for the act comments, without the act itself. Furthermore, we create additional in-depth evaluation examples for modal verbs and negation, which our domain experts are especially interested in. We create separate evaluations for modal verbs in positive (e.g. *should*) and in negative formulation (e.g. *might not*), to avoid confounding with the effect of negation. These 3 evaluations (positive modal verbs, negative modal verbs, negation) are created for acts without comments, act comments, and acts concatenated with comments.

One evaluation instance is created for each example in the evaluation set that contains any member of the evaluation's equivalence classes. If the evaluation instances n have not reached 100 yet, $\lfloor \frac{100}{n} \rfloor$ negative spans $a^{(\text{neg})}$ are sampled per instance, such that the total number is close to 100. If more than 100 matches are found, all of them are included in the evaluation with one randomly sampled negative span. We do not replace positive spans. This procedure generates 1974 total evaluation instances from the validation set, and 2104 from the test set. The general evaluations of the 5 categories plus the act comments (excluding in-depth

evaluations) contain 818 evaluation instances from the validation set, and 886 from the test set. The smallest category (motive) has 74 and the largest (act labels) has 336 test evaluation instances.

7.5.2 Standard Text Generation Metrics

Since our task is a sequence-to-sequence task, we also report standard text generation metrics. If not mentioned otherwise, we compute the following metrics for generated annotations without special tokens (category start and end tokens).

ROUGE. ROUGE (Lin, 2004) is a textual overlap metric that is widely used in text summarization, a task with strong connections to ours. As is common in summarization, we report ROUGE-1/2/L as the unigram and bigram overlap, and the longest common subsequence, respectively. We compute ROUGE with and without special tokens, as we want to see both how well the model generates the annotations as well as the original article.

BERTScore. We use BERTScore (Zhang et al., 2020b) as a semantic similarity metric between the generated and reference target annotations. We do not use idf-importance weighting, and we use baseline rescaling.[2] If multiple target annotations are present, the maximum similarity is reported, as proposed by the authors.

Distinct bigrams. We report the distinct bigrams in the generated target annotations. This metric checks if the model produces overly generic and repetitive outputs. A higher number of distinct bigrams corresponds to higher lexical diversity in the output and is desirable.

Novel bigrams. Novel bigrams measure the percent of bigrams in a generated annotation that do not appear in the filtered source document that serves as input text. This metric measures the extractiveness of the model, i.e. its tendency to copy text from the input.

Annotation Category Metrics

We add annotation category-specific metrics to the text generation metrics. These metrics are designed to detect if any category or the target format is ignored by the model.

Category counts. We report the mean of each annotation category's occurrence over the generated target annotations.

[2] Evaluation hash: `roberta-large_L17_no-idf _version=0.3.11(hug_trans=4.6.1)-rescaled`

Categories correctly closed. This evaluation measures the percent of annotation spans that are correctly encompassed by a category start and end token. This evaluation shows whether the decoder correctly learned to generate in the target format.

7.5.3 Filtering Source Documents

As detailed in Section 7.2.4, the source documents are much longer than current Transformer models with quadratic self-attention complexity can process. However, we conjecture that only very specific parts of these documents are needed to generate the comparably very short target annotations (see Table 7.2). On top of mentioned filtering strategies, we train a filtering model. For this purpose, we finetune a BERT model (Devlin et al., 2019) for sequence classification.[3] We split long inputs at sentence boundaries into chunks of at most 512 tokens, and then predict whether to keep the sentences in the current chunk.

We train the model with a cross-entropy loss between the predictions and the oracle selection described in Section 7.2.4. We train with a batch size of 5 for 10 epochs but stop early when the F1 score on the validation set no longer improves. We use the same learning rate schedule as for the generative models described below, with a maximum learning rate of 1e−3. During inference, we select the sentences with the highest logits until we reach the token limit. The selected sentences are concatenated in the order in which they appear in the source document. We name this filtering model *FilterBERT*.

7.5.4 Generative Models

For our generative models, we rely on the Transformer architecture (Vaswani et al., 2017) and compare finetuning differently pretrained models.

Transformer. We use a randomly initialized Transformer encoder-decoder to test the effect of skipping pretraining. Our implementation of the Transformer is the same as the BERT model below.

BERT. We finetune a pretrained BERT encoder (Devlin et al., 2019) and train a randomly initialized Transformer decoder, as proposed in Liu and Lapata (2019b). Unless otherwise mentioned, we use the base model size.

Sentence planner. We finetune the sentence planner model from Chapter 3. We initialize the encoder with a pretrained BERT model, and the sentence and word generators with randomly initialized Transformer decoders. We use the same model size as BERT-base, and two layers for the sentence generator. From Table 7.2, we know that target annotations have an average

[3] We use the standard implementation in the Hugging Face transformers library (Wolf et al., 2020).

Model	Act	Act comments	Attribution	Motive	Evidence	Scope	Mean
Transformer	93.18%	93.45%	94.79%	66.22%	43.68%	50.00%	73.55%
BERT	97.73%	94.64%	97.16%	66.22%	45.98%	54.44%	76.03%
Sentence planner	97.73%	93.15%	95.73%	64.86%	44.83%	56.67%	75.50%
BART	98.86%	96.13%	97.16%	71.62%	81.61%	80.00%	87.56%

Table 7.3: Accuracy of main equivalence classes evaluations.

of 1.6 standardized sentences. Overall, 29.40% of examples have more than one standardized sentence, for which hierarchical decoding could be beneficial. These standardized sentences do not need to be consecutive in the original article, however.

BART. We finetune the BART model (Lewis et al., 2020) as a proponent of a jointly pretrained encoder and decoder.

Training details. Training steps and learning rate hyperparameters were selected on the validation set with a grid search with exponential step sizes. We train our models for a maximum of 10 (Transformer/BERT) or 20 (sentence planner/BART) epochs, which corresponds to 8000 or 16000 steps with a batch size of 4, respectively. We stop training early if the validation loss does not improve any further. We set the maximum learning rate to 1e−4 for randomly initialized parameters, and 1e−5 for pretrained ones. Exceptionally for BART, we use a learning rate of 1e−6 for the tied input/output embeddings. We warm up the learning rate for a tenth of the total epochs, with a linear increase from $\mathrm{lr}_{\max}/100$ to the maximum learning rate $\mathrm{lr}_{\max}$, and then a linear decay back down to the starting point. We use the Adam optimizer (Kingma and Ba, 2015).

Generation details. For our evaluation of text generation metrics (see Section 7.5.2), we generate text with beam search. We use 5 beams, a minimum generation length of 50 tokens and a maximum of 500, no length penalty, and no n-gram blocking (Paulus et al., 2018).

7.6 Results

In this section, we report the results of our equivalence classes and text generation evaluation, conduct an in-depth analysis of models' performance on acts with negation and modal verbs, ablate model sizes, filtering strategies and input lengths, and explore hallucinations in BART's outputs.

Model	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore	Distinct bigrams	Novel bigrams
References	-	-	-	-	9961	82.68%
Transformer	31.89	10.27	25.06	0.72	214	87.61%
BERT	41.09	18.31	31.63	19.00	965	82.75%
Sentence planner	41.58	18.02	31.35	19.49	1108	81.55%
BART	42.73	20.64	33.08	26.78	3011	73.62%

Table 7.4: Text generation evaluation results. ROUGE is computed on targets including special tokens.

Model	Std sent	Act	Actor	Reference	Attribution	Motive	Evidence	Scope	Closed
References	1.60	1.60	1.60	1.60	0.87	0.39	1.22	0.21	100.00%
Transformer	2.69	2.69	2.69	2.60	0.02	0.08	0.05	0.01	100.00%
BERT	1.63	1.63	1.63	1.63	0.73	0.43	0.07	0.14	99.97%
Sentence planner	1.31	1.31	1.34	1.31	0.64	0.41	0.34	0.18	100.00%
BART	1.55	0.79	1.22	1.37	0.29	0.02	0.13	0.05	69.44%

Table 7.5: Mean category counts.

7.6.1 Equivalence Classes Evaluation

Our main results for the general equivalence classes evaluations on the 5 categories plus act comments are shown in Table 7.3. The BART model with a jointly pretrained encoder and decoder substantially outperforms the Transformer, BERT, and sentence planner models. The act, act comments, and attribution evaluations are solved nearly perfectly, but the others are harder. For the evidence evaluation, Transformer, BERT and sentence planner perform substantially below the random baseline, which would achieve 50% in expectation. We analyze the case of the evidence evaluation further in Section 7.6.3.

7.6.2 Text Generation Metrics

We show the results of text generation metrics in Table 7.4. Again, BART outperforms the other models. The low scores in BERTScore and distinct bigrams (excluding special tokens) indicate that the Transformer fails to generate diverse and topical target annotation sentences. However, the comparably high ROUGE scores (including special tokens) show that it learns to generate the target format well, which is also supported by the last column of Table 7.5. The sentence planner performs on par with BERT. BART generates the most diverse and topical target annotations, and it is also the most extractive method, showing that it makes use of the input document.

In Table 7.5, we show the mean of each category's annotation counts for our three models. BERT produces outputs that stay closest to the number of category annotations of the reference target annotations. The sentence planner generates fewer but longer sentences, a result we have also observed for the BERT-large model size. BART under-generates all categories, which can be partially explained by it not having learned to open and close category spans reliably.

Model	Act negation	Act modals (pos)	Act comment modals (pos)	Act with comment modals (pos)
Transformer	89.58%	70.65%	93.81%	68.13%
BERT	93.75%	70.65%	93.81%	69.23%
Sentence planner	93.75%	73.91%	95.88%	78.02%
BART	95.83%	89.13%	97.94%	80.22%

Table 7.6: Accuracy on selected in-depth equivalence classes evaluations.

The combination of not having seen the format during pretraining and a lower decoder learning rate, which was helpful for the other tasks, explains why BART performs worse than the models with randomly initialized Transformer decoders.

7.6.3 In-Depth Analysis

Table 7.6 shows a selection of equivalence classes evaluations where equivalence classes were built for a specific purpose. In our evaluations, these measure performance on act modal verbs (e.g. *raised rates* vs. *might raise rates*) and act negation, both aspects that are of high importance to our domain experts. We can see that negation is handled well by all models, and that modals are substantially harder for acts, but not for act comments (where the act is part of the prefix). Acts with comments (last column) do not necessarily make the task easier than acts without comments (second column).

We also perform a qualitative in-depth analysis of the evidence evaluation for the BART model. To that effect, we count the percentage of evaluation instances the model gets wrong for each pair of equivalence classes, which is shown in the confusion matrix in Figure 7.6. The number in each square corresponds to the number of mistakes in the evaluation. Some of the mistakes occur for the following pairs of equivalence classes, where $a^{(\text{pos})}$ is taken from the first, and $a^{(\text{neg})}$ from the second:

- deflation – low/declining inflation
- cooling housing market – tightening credit market (full example shown in the paragraph below)
- high unemployment – high oil prices
- high unemployment – weak economic activity
- slowing growth – low money supply growth

The mentioned combinations of economic processes are correlated or even co-occurring, making it difficult for the model to distinguish the positive from the negative span. In these cases of close semantic similarity, the model may fall back to ranking candidate text spans higher based on e.g. their frequency in the training data, where inflation is one of the dominant

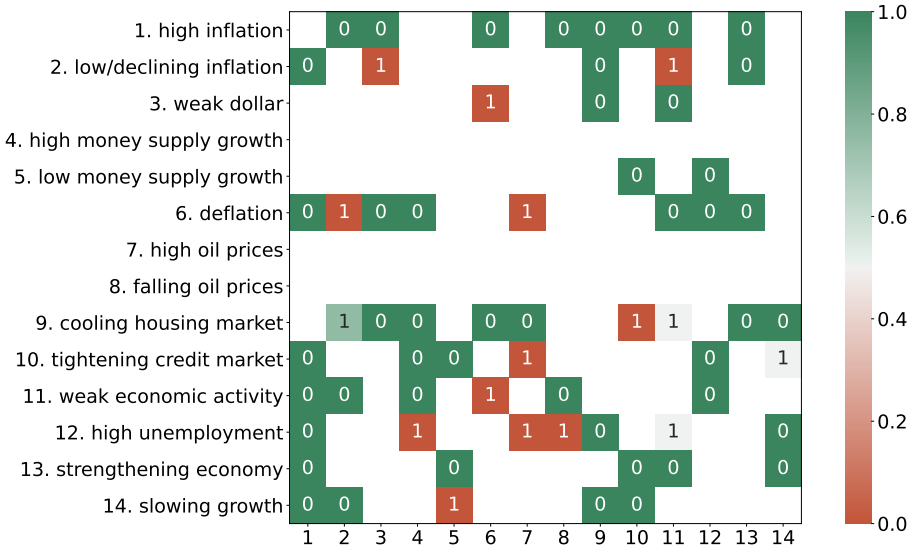


Figure 7.6: Confusion matrix of BART's accuracy on pairs of evidence equivalence classes. The equivalence class for $a^{(\text{pos})}$ is on the y-axis, and the one for $a^{(\text{neg})}$ is on the x-axis. Each cell contains the number of mistakes the model makes for that combination. Empty cells do not have a corresponding pair in the evaluation.

subjects. For other combinations, such as weak dollar – deflation, the model just makes mistakes.

Equivalence classes evaluation instance. We present the evaluation instance corresponding to the pair of true and distracting evidence mentioned before (cooling housing market – tightening credit market). BART got this example wrong, i.e. judged the negative continuation $a^{(\text{neg})}$ as more likely than the positive $a^{(\text{pos})}$. We also see this example in Figure 7.6, positive class 9, negative class 10.

$y_{\textbf{prefix}}$**:** [STD SENTENCE START] Ben S. [ATTRIBUTION START] Bernanke [ATTRIBUTION END] , the chairman of the [ACTOR START] Federal Reserve [ACTOR END] Board, [REFERENCE START] declared [REFERENCE END] on Friday that the central bank [ACT START] "stands ready to take additional actions as needed" (Might cut rates, in future) [ACT END] [MOTIVE START] to prevent the chaos in mortgage markets from derailing the broader economy [MOTIVE END] . Mr. Bernanke avoided any specific promise to lower the central bank's benchmark federal funds rate at its next policy meeting on Sept. 18. But he acknowledged [EVIDENCE START]
$a^{(\textbf{pos})}$**:** the dangers posed by the twin storms in housing and mortgage lending
$a^{(\textbf{neg})}$**:** credit was becoming harder to get for both consumers and businesses

Model	Parameters	EQ mean	ROUGE			BERTScore	Distinct bigrams
			R-1	R-2	R-L		
Transformer	247M	73.55%	31.89	10.27	25.06	0.72	214
BERT-base	247M	76.03%	41.09	18.31	31.63	19.00	965
BERT-large	771M	75.76%	41.26	17.91	31.39	19.30	1232
Sentence planner	281M	75.50%	41.58	18.02	31.35	19.49	1108
BART	406M	87.56%	42.73	20.64	33.08	26.78	3011

Table 7.7: Selected evaluation metrics for different model sizes.

Model	Filtering	EQ mean	ROUGE			BERTScore	Distinct bigrams
			R-1	R-2	R-L		
BERT	FilterBERT	76.03%	41.09	18.31	31.63	19.00	965
BERT	Lead	75.51%	41.27	18.74	31.44	19.59	1012
BERT	Oracle	75.15%	41.38	18.54	32.05	19.96	1010
BART	FilterBERT	87.56%	42.73	20.64	33.08	26.78	3011
BART	Lead	86.90%	41.56	19.79	32.09	25.15	1976
BART	Oracle	87.16%	44.04	21.84	33.87	26.98	3528

Table 7.8: Selected evaluation metrics for different filtering strategies.

7.6.4 Ablation Study

We perform ablation studies with respect to model size, filtering strategy, and source document input length.

Model sizes. In the results shown so far, BART has outperformed the Transformer, BERT, and the sentence planner. However, those models operate with 247 million (the size of BERT-base) and 281 million parameters, while BART has 406 million. In Table 7.7, we see that increasing BERT's parameters to the size of BERT-large only provides small to no benefits. Higher BERTScore and lexical diversity (distinct bigrams) are also achieved by the sentence planner with fewer additional parameters. BART outperforms BERT-large with almost half of the parameters. We believe its initialization from joint encoder-decoder pretraining is especially valuable on the FOMC dataset, which has comparably few training instances.

Filtering strategies. In Section 7.5.3, we introduced the FilterBERT model for identifying and selecting salient sentences from long source documents. As stated in Section 7.2.4, together with the dataset, we make available a script for filtering source documents with either the Lead or the Oracle strategy. The former selects sentences from the top of the source document, the latter selects those that most increase the length-normalized ROUGE-2 recall with the target annotations. In Table 7.8, we see that Oracle filtering generally performs best on generation metrics, but not on equivalence classes. The FilterBERT model outperforms the Lead strategy for BART but not for generation metrics on BERT. In general, the differences between the

Model	Filtering	Input tokens	EQ mean	ROUGE			BERTScore	Distinct bigrams
				R-1	R-2	R-L		
BART	Lead	512	86.90%	41.56	19.79	32.09	25.15	1976
BART	Lead	1024	87.12%	42.39	19.64	32.20	25.44	2421
BART	Oracle	512	87.16%	44.04	21.84	33.87	26.98	3528
BART	Oracle	1024	89.15%	44.81	21.36	34.77	27.79	3296

Table 7.9: Selected evaluation metrics for different source document input lengths.

generative models are much larger than between the filtering strategies.

Source document input length. Finally, since BART has the ability to process inputs of up to 1024 tokens in length, we evaluate how that compares to the input length of 512 tokens that we have used so far. The results in Table 7.9 show that for the Lead filtering strategy, longer inputs benefit all metrics except ROUGE-2. With Oracle filtering, ROUGE-2 and distinct bigram evaluations perform slightly worse with longer inputs, while the rest improve. In summary, the additional input sentences only make a small difference for BART.

7.6.5 Hallucinations

We perform an exploratory analysis of hallucinations in BART's outputs. To detect hallucinations, we use BART-GBP from Chapter 6. We run our method with the finetuned BART model on the FOMC test set without adaptations or further training. We then annotate the first 20 examples for hallucinations.

The most common type of hallucination is a made-up news article date, which we found in 55% of test outputs. The FOMC dataset's target annotations refer to the policy announcements with date expressions, such as *today*, *last week*, or *on Tuesday*, without grounding in the source document. BART learns to imitate this behavior. These hallucinations are irrelevant to our domain experts and the task, so we ignore them for the rest of the analysis.

Overall, we find that 45% of outputs have one hallucination, and 10% have two, for a total of 13 hallucinations. They can be clustered into four groups. First, the model tries to fit a decision to raise or lower interest rates into the act or motive (5 examples). Second, it makes up a vote when none is present in the source document (4 examples). Third, it hallucinates correct facts that it knows or has picked up during finetuning, such as that the FOMC consists of 12 members or that its meetings usually take two days (2 examples). Fourth, there are also 2 instances of intrinsic hallucinations, where information from the source document is combined in the wrong way.

Finally, we use BART-GBP to rank the output tokens according to their hallucination probability. For the examples that contain hallucinations, we find the rank of each hallucination in the

sorted output tokens. We then compute the mean reciprocal rank (MRR) over all 20 test examples. With an MRR of 0.67, BART-GBP achieves a high score, typically finding hallucinations at rank 1 or 2 of its hallucination candidate list.

7.7 Related Work

To the best of our knowledge, our setting, task, and evaluation have not been studied in prior work. We therefore describe studies with similarities to individual aspects of our work.

Evaluation. The closest approach to our equivalence classes evaluation is the concept of verbalizers in Pattern-Exploiting Training (PET) (Schick and Schütze, 2021a,b), the relation to which we already discussed in Section 7.4.2. The biggest difference to our approach is that PET's verbalizers are limited to a small, bounded set of predefined single tokens or few-token spans, while our equivalence classes are unbounded, and their members are collected from the data without restrictions on length or content.

Other work has also tried to make human annotations more efficient, e.g. for importance judgments of sentences in multi-domain summarization (Jha et al., 2020), or multi-task information extraction (Bikaun et al., 2022). AnnIE builds fact synsets to speed up open information extraction (Friedrich et al., 2022).

Annotations. A similar annotation scheme to ours has been used for training Galactica, an LLM for science (Taylor et al., 2022). They use annotations to mark citations, amino acid and DNA sequences, molecule structure, and computation. The annotations can be used for controllable generation and for offloading computation to an external model.

Aspect-oriented summarization. Interpretations may focus on certain aspects of the source documents, making them somewhat similar to aspect-oriented summarization. AspectNews (Ahuja et al., 2022) and SPACE (Angelidis et al., 2021) are two recent datasets with accompanying models.

News summarization. Since our interpretations are excerpts of New York Times articles, news summarization is relevant to our work as well. This is a very active field of research, with multiple large-scale datasets (e.g. CNN/DM (Hermann et al., 2015; Nallapati et al., 2016), XSum (Narayan et al., 2018a), NEWSROOM (Grusky et al., 2018), or Multi-News (Fabbri et al., 2019), among others). A lot of methods have been tried on these datasets. BART (Lewis et al., 2020) and PEGASUS (Zhang et al., 2020a) have shown some of the best results for finetuned models. Recently, zero-shot summarization from very large pretrained decoder-only language models has spurred a lot of interest, achieving performance close to finetuned models (e.g. PaLM (Chowdhery et al., 2022)).

7.8 Conclusion

We have devised a method to convert semi-structured human annotations into text format. We then introduced a task of predicting annotated interpretations of source documents that can be tackled with sequence-to-sequence models. We presented a human-annotated corpus about the monetary policy of the Federal Reserve. Our equivalence classes evaluation is an efficient technique to create a large number of targeted evaluation instances from a comparably cheap clustering by domain experts. We use this technique to evaluate state-of-the-art generative models on our task, and find that it shows larger differences between pretrained models than standard text generation metrics. In further in-depth analyses, the equivalence classes evaluation tests the models for specific properties, such as how they handle negation, and detects why models struggle to correctly rank alternative text spans of certain human annotation categories.

7.8.1 Value of Annotations for Language Models

In our evaluation, we saw that our annotations help the language models pick up on the important details of interpretations even with small training data. We believe that this observation could transfer to other low-resource settings.

Moreover, by structuring the different parts of target documents with beginning and end tags, the language model gets an explicit marker for the context. This is a strong signal of which words are more likely continuations, and a start marker can affect the next word probability distribution. A similar approach could also be used for controllable text generation, where tags can be used to select the aspect, sentiment, or any other desired property to be generated. The downside to such an approach is the expensive annotation that needs to be performed on the training data. It remains to be seen to what extent that annotation could be supported by automated methods while maintaining high data quality.

7.8.2 Subjectivity of the Annotation Process

Even though annotation protocols can be standardized and outputs aggregated over multiple annotators, the process of annotation remains subjective. In our interpretation task, social scientists extract and categorize information, providing additional context where necessary, and all annotations are validated by a senior domain expert. The models trained on the data will focus on the aspects that the annotators deemed important. This is not inherently a bad thing. Human annotation is a flexible tool that a different set of annotators could use to highlight other aspects of the data. Note that this is a separate consideration from the reproducibility of our results, which we enable by open-sourcing our data, code, and models.

7.8.3 Application to Other Domains

We have yet to establish the transferability of our allowed set of annotations and task setup to other domains. While we expect our procedure to be general enough to work in different areas, this chapter only uses a single corpus about macroeconomics. The reason for the limitation to one corpus is the high cost of finding relevant interpretation documents, performing the extraction and annotation, and standardizing the resulting annotations.

7.8.4 Equivalence Classes Creation

While the creation of equivalence classes is less expensive than directly creating evaluation examples, it still requires manual effort by domain experts, which is an expensive resource. This could be alleviated with an automatic method to obtain equivalence classes. In theory, the identification of candidate members of equivalence classes should be facilitated by the category annotations. The two member properties of (1) semantic interchangeability within equivalence classes and (2) syntactic interchangeability across equivalence classes could potentially be judged by a strong language model. When using this external model, care has to be taken that none of the sequence-to-sequence models gets an advantage, e.g. from being of the same model family as the external model.

7.8.5 Syntactic Structure of Equivalence Class Members

Syntactic interchangeability is a requirement on equivalence class members within one equivalence classes evaluation. This limits us to one syntactic construction per evaluation. We select the most common one in each category to obtain a large enough number of evaluation instances. As a consequence, the model will not be tested on different syntactic structures. Unfortunately, testing all possible syntactic constructions suffers from (1) a data sparsity problem, where not enough examples of the same construction occur in the data, and (2) a large increase in the manual effort required to construct one evaluation per syntactic structure.

8 Conclusion

In this thesis, we presented approaches that aim to improve diverse aspects of abstractive summarization. In Part I, we investigated interpretable representation learning. We devised a hierarchical Transformer decoder for predicting a latent plan for the next summary sentence before generating its words. We aimed to learn contextualized representations with an entailment interpretation, from which the entailment of the summary as well as the absence of hallucinations could be deduced. From unsupervisedly learning semantic text units, we hoped to gain further insights into the detection and extraction of salient phrases of source documents. In Part II, we used by-products of summary generation (attentions and decoding probabilities) to detect where the model is prone to generating hallucinations, due to the fusion of unrelated text segments and high-entropy guesses of phrases not copied from the source document. We applied sequence-to-sequence models in an interdisciplinary project on a novel task of interpreting policy announcements and found that detailed high-quality annotations enable the model to learn subtle cues from limited data. With a new evaluation technique based on the annotations, we comprehensively evaluated specific aspects of interpretation from a relatively cheap manual clustering by domain experts.

8.1 Summary of Contributions

We will now stress the most important contributions of this thesis, which could be interesting to the larger NLP research community.

In Chapter 3, we introduced a hierarchical Transformer decoder that generates a representation for the next summary sentence, which it then uses in the word generator to predict the sentence's words. As a result of this high-level plan for the summary sentences, the generated summaries turn out to be more abstractive. The increased abstractiveness does not decrease ROUGE scores. We also found that our hierarchical inductive bias is more effective than simply increasing the base model's parameters.

From our negative result of devising architectures with an entailment interpretation in Chap-

ter 4, we learned that these general-purpose architectures are very powerful, and small architectural changes, such as a different activation function, did not make a difference after hyperparameter tuning. The learning rate and its schedule proved to be the most important hyperparameter for the performance. While our entailment-based interpretation reached the same performance as the base architecture, it was not able to improve natural language inference or language modeling.

In Chapter 5, we aimed to induce semantic units of text with different object discovery mechanisms and analyzed the models' attentions carefully. Neither slot attention nor bottom-up attention were able to induce effective semantic units. Our trained models ignored our object representations or found other creative ways to circumvent our inductive biases, such as initialization, attention guidance, or the information bottleneck. Unsupervised learning of semantic text units remains a challenging and open-ended problem. We have provided a detailed account of conceptually promising approaches that failed to discover semantic units.

In Chapter 6, we detected hallucinations from the by-products of summary generation, namely encoder self-attentions, decoder cross-attentions, and decoding probabilities. Unlike prior work, our method does not require an external model to be trained and run. Additionally, we extended current hallucination detection research to CNN/DailyMail and to the token level. For this task, we provided two datasets for future research. We found that our method detects hallucinations more accurately than prior work. Still, intrinsic hallucinations remain challenging to detect for all methods.

Finally, in Chapter 7, we converted a dataset annotated with marked spans and comments into text format and proposed a sequence-to-sequence task. We released a dataset annotated with the aspects of interpretations. We then trained models to generate these semi-structured annotations. We introduced the equivalence classes evaluation, a scalable and fine-grained evaluation of model generations for the individual annotation categories. Our evaluation showed that our annotation scheme is a natural fit for language modeling and allowed state-of-the-art models to pick up on subtle details from low-resource data. We shared the best model with our collaborators from the Geneva Graduate Institute who are social scientists in the fields of economy and politics. They can query the model for particular aspects of an interpretation by providing a prompt and a category start token. The model completes the prompt with the most likely continuation. The scientists can also craft hypothetical and counterfactual scenarios by providing a corresponding prompt. They use it as a data-driven analysis and simulation tool for interpretations of policy announcements.

8.2 Inspirations for Future Work

As is probably not unusual, this thesis has left us with more ideas for further investigations than answered questions. We present the most promising in the following, starting with ideas for hallucination detection.

We would like to further extend the evaluation of entailment representations to check if they can be used to judge the faithfulness of summaries. As hinted at in Section 4.1, a summary (hypothesis) should be entailed by the source document (premise), unless it contains hallucinations. Entailment representations should benefit this task by enabling reasoning over the facts in their dimensions.

Our impression from our labeling efforts in Chapter 6 is that hallucination detection is underspecified. This was corroborated by the feedback from our human annotators for the TLHD-CNNDM dataset. There is a subjective component of what it means for a statement to be supported by the document, i.e. how much inference and world knowledge on the part of the reader is allowed. We believe that currently, applying the laws of physics to infer a statement would be acceptable, whereas knowing the name of a football stadium is not. Without explicitly defining these rules, it will be very hard for a model to identify the boundaries of acceptability. Consequently, a direction to benefit further research would be to come up with a more robust definition of what makes a hallucination.

If token- or phrase-level hallucination detection and correction are pursued, then an additional disambiguation step is required. If the source document does not support that a certain entity has performed a certain action, but both the entity as well as the action appear in the document, which one is the hallucination? We contemplated multiple solutions to this question, but editing-based considerations (we would like to require minimal edits) usually opposed consistent annotation.

A possible way out would be to switch to task-based evaluation (Zhang et al., 2023). If the task were defined as how effectively a model helps a human editor detect hallucinations, performance could be measured with the number of detected hallucinations per time. As an added benefit, this evaluation includes the inference cost of the models, an important part of practical application.

Models trained on CNN/DailyMail generate drastically fewer hallucinations compared to those trained on XSum. This has led us to question whether instead of detecting hallucinations and correcting model outputs, one could filter hallucinations in the training data to solve the problem. Even for LLMs, supervised finetuning on high-quality data has shown the fewest hallucinations (Ouyang et al., 2022). We speculate that this is infeasible due to two major factors. First, the filtering method would have to be automatic, since training datasets in use are too large for human annotation, and therefore some hallucinations will slip by the filtering process. Second, even with smaller but higher-quality datasets, scenarios similar to those seen in pretraining will still lead models to hallucinate.

An alternative solution could be to train more faithful models which stick closer to the input. If fuzzy memories from pretraining have less influence on generation, we conjecture that not just hallucinations would dwindle, but also other unwanted artifacts like biases and toxicity. One possible avenue is to answer the question of what it takes to make attention explanation (Jain and Wallace, 2019; Wiegreffe and Pinter, 2019; Brunner et al., 2020). We

temporarily investigated such an idea by making models' attentions more faithful to their generations, by sparsifying attention weights and imposing losses for too diverse attention.

We have also looked into using decoding probabilities to segment outputs. In Chapter 6, we briefly introduced a possible segmentation algorithm in Equation 6.3. The decoding probabilities and entropies identify the pivotal decisions during generation. A new decoding algorithm could use this information to replace word-by-word decoding with structured decoding, at a segment or phrase level, potentially with a smaller search space but a larger depth than currently (width $|\mathcal{V}|$, depth 1).

The annotation with category markers used by us and Galactica can be used to make better use of (low-resource) data. It remains an open question whether a limited budget for human annotation is better spent to increase the quantity or the quality of a dataset. It remains to be seen whether such annotations can be gathered more efficiently, e.g. with model-proposed tags that can be used to speed up human labeling or as data for weak supervision.

In either case, high-quality annotations enable controllable generation with a simple mechanism tailored to language models that generate the continuation based on the recent prefix. By inserting the desired category start token, one can guide the model to generate a continuation of that type. Depending on the annotation in the training data, this continuation can be single words or phrases, citations, or even placeholders for input from an external program (Taylor et al., 2022). The interactive operation mode for social scientists mentioned in Section 8.1 is a further promising direction for future work. In a time where models become larger and larger, it is an open question whether we will ever be able to fully understand how a prediction was made. The proposed approach offers a way to interact with this black box via natural language.

We are also curious to see whether new annotations can be picked up by LLMs through in-context learning. If only a modest amount of high-quality annotations are required for any given task, the few-shot setting of current LLMs would prove to be an even more enticing application for this annotation scheme.

8.3 Limitations

The black-box nature of neural networks also underlies the limitations of this thesis. In Chapter 3, for example, we have a plausible hypothesis why the sentence planner generates more abstractive summaries at higher ROUGE scores, but we cannot prove it without further insights into how the predictions are made. We did a thorough analysis to rule out typical explanations (increased number of parameters) and easy shortcuts (by checking the most common novel bigrams) but did not find conclusive evidence to answer that question.

Furthermore, evaluations are limited to the specific data distribution and metric they are evaluating, which makes it hard to draw general conclusions. We can harden the results by

performing many tests to arrive at a more nuanced understanding of a method's strengths and weaknesses, and we tried to do so in this thesis. An aspect that often goes untested in scientific studies is a method's robustness to unexpected inputs. Potentially, companies with real customers are better suited to take over that role.

Limitations of automatic and manual evaluation are discussed in Section 2.4. They also apply to this thesis. Additionally, due to cost and time constraints, our human evaluation in Chapter 3 is of small sample size and therefore larger variance. Considering this, the p-values in Table 3.9 are surprisingly low, suggesting that a larger human evaluation could increase the confidence in the results.

While we presented negative results in Chapter 4 and 5, it is still possible for these methods to work with adaptations or different hyperparameters. To facilitate future research, we tried to be accurate and verbose in describing what we tried.

In Chapter 6, and as already mentioned in Section 8.2, we presume that the task of token-level hallucination detection is underspecified, contributing to the difficulty of the models in uncovering hallucinations. The datasets are rather small, owing to the cost of human annotation.

The equivalence classes evaluation in Chapter 7 only evaluates the most common grammatical structures. This limits the robustness of the evaluation, which would be desirable for interactive prompting by domain experts, as detailed in Section 7.3.2. We hope that the contrastive nature of the evaluation helps avoid misleading conclusions. In almost all chapters of the thesis, pretraining is used to initialize models, or the models are used directly. We therefore inherit these models' limitations concerning generating biased, harmful, or toxic content as a result of its presence in the pretraining data. Models also exhibit overconfidence and assertiveness when in error, and hallucinate information. We refer the reader to the detailed limitations sections of more recent papers, which contain pointers to further studies on these subjects.

8.4 Ethical Considerations

The ethical implications of AI systems are an important societal topic. Summarization is one of the less harmful tasks to study. One danger of summarization in particular is the loss of information and context that comes with reducing the output length of a text. The loss of context can lead to oversimplification, misinterpretation, and in the worst case accidental misinformation by the summarization model. Summarization researchers aim to combat these issues by increasing the relevance and informativeness of their models' outputs.

Hallucinations are another source of misinformation. In this thesis, we contribute to the research area of hallucination detection. We find that this is still a hard topic, and propose avenues for future work.

In the larger field of text generation, additional ethical considerations are necessary. As mentioned in Section 8.3, pretrained language models reproduce biases present in their training data. For now, the goals of researchers are aligned with reducing these biases, as they negatively impact task performance in summarization (e.g. relevance, informativeness, faithfulness, factuality).

Improving language generation models in general, however, does also facilitate misinformation campaigns or propaganda of malicious actors. It will be even more important for people to verify the trustworthiness of their sources, and to become literate in the capabilities of language models. Clark et al. (2021) found that most of their evaluators underestimated language generation models.

The same danger holds for language models that are better aligned with their operator's intentions. Through improving controllable generation, it will become easier to produce targeted exploitative text, such as phishing emails, fake news, or political defamation. As of now, we do not have the tools to reliably control language models to resist attempts to create such text. This is showcased by the public release of ChatGPT, which resulted in a cat-and-mouse game between the developers trying to prevent certain prompts from being answered and users that created ever new prompts to circumvent the updated safeguards. If we could reliably detect machine-generated text, either through watermarking the output or by automatic tools, we could identify manipulation attempts. Unfortunately, this is not yet possible, and potentially never will be. As a result, further research on AI alignment and understanding natural language generation models is needed, combined with open and accessible information of the general population about the capabilities of AI.

Bibliography

Ahmed, K., Keskar, N. S., and Socher, R. (2017). Weighted transformer network for machine translation. *CoRR*, abs/1711.02132.

Ahuja, O., Xu, J., Gupta, A., Horecka, K., and Durrett, G. (2022). ASPECTNEWS: Aspect-oriented summarization of news documents. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6494–6506, Dublin, Ireland. Association for Computational Linguistics.

Angelidis, S., Amplayo, R. K., Suhara, Y., Wang, X., and Lapata, M. (2021). Extractive opinion summarization in quantized transformer spaces. *Transactions of the Association for Computational Linguistics*, 9:277–293.

Aralikatte, R., Narayan, S., Maynez, J., Rothe, S., and McDonald, R. (2021). Focus attention: Promoting faithfulness and diversity in summarization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6078–6095, Online. Association for Computational Linguistics.

Ba, L. J., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *CoRR*, abs/1607.06450.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., Joseph, N., Kadavath, S., Kernion, J., Conerly, T., Showk, S. E., Elhage, N., Hatfield-Dodds, Z., Hernandez, D., Hume, T., Johnston, S., Kravec, S., Lovitt, L., Nanda, N., Olsson, C., Amodei, D., Brown, T. B., Clark, J., McCandlish, S., Olah, C., Mann, B., and Kaplan, J. (2022). Training a helpful and harmless assistant with reinforcement learning from human feedback. *CoRR*, abs/2204.05862.

Baldassarre, F. and Azizpour, H. (2022). Towards self-supervised learning of global and object-centric representations. In *ICLR2022 Workshop on the Elements of Reasoning: Objects, Structure and Causality*.

Banerjee, S. and Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Behjati, M. and Henderson, J. (2021). Inducing meaningful units from character sequences with slot attention. *CoRR*, abs/2102.01223.

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Bhandari, M., Gour, P. N., Ashfaq, A., Liu, P., and Neubig, G. (2020). Re-evaluating evaluation in text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9347–9359, Online. Association for Computational Linguistics.

Bikaun, T., Stewart, M., and Liu, W. (2022). QuickGraph: A rapid annotation tool for knowledge graph extraction from technical text. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, Dublin, Ireland. Association for Computational Linguistics.

Bird, S., Loper, E., and Klein, E. (2009). *Natural Language Processing with Python*. O'Reilly Media, Inc.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Brunner, G., Liu, Y., Pascual, D., Richter, O., Ciaramita, M., and Wattenhofer, R. (2020). On identifiability in transformers. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M. M., and Lerchner, A. (2019). Monet: Unsupervised scene decomposition and representation. *CoRR*, abs/1901.11390.

Cao, M., Dong, Y., and Cheung, J. (2022). Hallucinated but factual! inspecting the factuality of hallucinations in abstractive summar ization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3340–3354, Dublin, Ireland. Association for Computational Linguistics.

Cao, M., Dong, Y., Wu, J., and Cheung, J. C. K. (2020). Factual error correction for abstractive summarization models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6251–6258, Online. Association for Computational Linguistics.

Celikyilmaz, A., Bosselut, A., He, X., and Choi, Y. (2018). Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1662–1675, New Orleans, Louisiana. Association for Computational Linguistics.

Chen, Y., Li, L., Yu, L., Kholy, A. E., Ahmed, F., Gan, Z., Cheng, Y., and Liu, J. (2020). UNITER: universal image-text representation learning. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXX*, volume 12375 of *Lecture Notes in Computer Science*, pages 104–120. Springer.

Chen, Y.-C. and Bansal, M. (2018). Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686, Melbourne, Australia. Association for Computational Linguistics.

Cheng, J. and Lapata, M. (2016). Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494, Berlin, Germany. Association for Computational Linguistics.

Child, R., Gray, S., Radford, A., and Sutskever, I. (2019). Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509.

Chitty-Venkata, K. T., Emani, M., Vishwanath, V., and Somani, A. K. (2022). Neural architecture search for transformers: A survey. *IEEE Access*, 10:108374–108412.

Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014a). On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Cho, S., Lebanoff, L., Foroosh, H., and Liu, F. (2019). Improving the similarity measure of determinantal point processes for extractive multi-docum ent summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1027–1038, Florence, Italy. Association for Computational Linguistics.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. (2022). Palm: Scaling language modeling with pathways. *CoRR*, abs/2204.02311.

Christiano, P. F., Leike, J., Brown, T. B., Martic, M., Legg, S., and Amodei, D. (2017). Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4299–4307.

Clark, E., August, T., Serrano, S., Haduong, N., Gururangan, S., and Smith, N. A. (2021). All that's 'human' is not gold: Evaluating human evaluation of generated text. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7282–7296, Online. Association for Computational Linguistics.

Cohan, A., Dernoncourt, F., Kim, D. S., Bui, T., Kim, S., Chang, W., and Goharian, N. (2018). A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana. Association for Computational Linguistics.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference d ata. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Crawford, E. and Pineau, J. (2019). Spatially invariant unsupervised object detection with convolutional neural networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence,*

AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, pages 3412–3420. AAAI Press.

Culp, L., Sabour, S., and Hinton, G. E. (2022). Testing glom's ability to infer wholes from ambiguous parts. *CoRR*, abs/2211.16564.

Curation (2020). Curation corpus base.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

Dang, H. T. (2005). Overview of DUC 2005. In *Proceedings of the Document Understanding Conference (DUC)*, pages 1–12, Vancouver, B.C., Canada.

Dangovski, R., Jing, L., Nakov, P., Tatalović, M., and Soljačić, M. (2019). Rotational unit of memory: A novel representation unit for RNNs with scalable application s. *Transactions of the Association for Computational Linguistics*, 7:121–138.

Daniluk, M., Rocktäschel, T., Welbl, J., and Riedel, S. (2017). Frustratingly short attention spans in neural language modeling. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, L. (2019). Universal transformers. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Deutsch, D. and Roth, D. (2019). Summary cloze: A new task for content selection in topic-focused summarization. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3720–3729, Hong Kong, China. Association for Computational Linguistics.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dhamdhere, K., Sundararajan, M., and Yan, Q. (2019). How important is a neuron. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Dhingra, B., Faruqui, M., Parikh, A., Chang, M.-W., Das, D., and Cohen, W. (2019). Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the*

57th Annual Meeting of the Association for Computational Linguistics, pages 4884–4895, Florence, Italy. Association for Computational Linguistics.

Dror, R., Baumer, G., Shlomov, S., and Reichart, R. (2018). The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

Durmus, E., He, H., and Diab, M. (2020). FEQA: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5055–5070, Online. Association for Computational Linguistics.

Durrett, G., Berg-Kirkpatrick, T., and Klein, D. (2016). Learning-based single-document summarization with compression and anaphoricity constraints. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1998–2008, Berlin, Germany. Association for Computational Linguistics.

Efron, B. and Tibshirani, R. J. (1994). *An introduction to the bootstrap*. CRC press.

Elman, J. L. (1990). Finding structure in time. *Cogn. Sci.*, 14(2):179–211.

Engelcke, M., Jones, O. P., and Posner, I. (2021). GENESIS-V2: inferring unordered object representations without iterative refinement. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 8085–8094.

Engelcke, M., Kosiorek, A. R., Jones, O. P., and Posner, I. (2020). GENESIS: generative scene inference and sampling with object-centric latent representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Erkan, G. and Radev, D. R. (2004). Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Eslami, S. M. A., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Kavukcuoglu, K., and Hinton, G. E. (2016). Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3225–3233.

Fabbri, A., Li, I., She, T., Li, S., and Radev, D. (2019). Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, Florence, Italy. Association for Computational Linguistics.

Fabbri, A. R., Kryściński, W., McCann, B., Xiong, C., Socher, R., and Radev, D. (2021). SummEval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409.

Falke, T., Ribeiro, L. F. R., Utama, P. A., Dagan, I., and Gurevych, I. (2019). Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2214–2220, Florence, Italy. Association for Computational Linguistics.

Filippova, K. (2020). Controlled hallucinations: Learning to generate faithfully from noisy data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 864–870, Online. Association for Computational Linguistics.

Friedrich, N., Gashteovski, K., Yu, M., Kotnis, B., Lawrence, C., Niepert, M., and Glavaš, G. (2022). AnnIE: An annotation platform for constructing complete open information extraction benchmark. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 44–60, Dublin, Ireland. Association for Computational Linguistics.

Gehrmann, S., Deng, Y., and Rush, A. (2018). Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.

Gers, F. A. and Schmidhuber, J. (2000). Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, IJCNN 2000, Neural Computing: New Challenges and Perspectives for the New Millennium, Como, Italy, July 24-27, 2000, Volume 3*, pages 189–194. IEEE Computer Society.

Gholipour Ghalandari, D., Hokamp, C., Pham, N. T., Glover, J., and Ifrim, G. (2020). A large-scale multi-document summarization dataset from the Wikipedia current events portal. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1302–1308, Online. Association for Computational Linguistics.

Gliwa, B., Mochol, I., Biesek, M., and Wawer, A. (2019). SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79, Hong Kong, China. Association for Computational Linguistics.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680.

Gopalakrishnan, A., Irie, K., Schmidhuber, J., and van Steenkiste, S. (2021). Unsupervised learning of temporal abstractions using slot-based transformers. In *Deep Reinforcement Learning Workshop*.

Goyal, T. and Durrett, G. (2020). Evaluating factuality in generation with dependency-level entailment. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3592–3603, Online. Association for Computational Linguistics.

Goyal, T. and Durrett, G. (2021). Annotating and modeling fine-grained factuality in summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1449–1462, Online. Association for Computational Linguistics.

Goyal, T., Li, J. J., and Durrett, G. (2022). News summarization and evaluation in the era of GPT-3. *CoRR*, abs/2209.12356.

Graff, D., Kong, J., Chen, K., and Maeda, K. (2003). English gigaword. *Linguistic Data Consortium*.

Graves, A. (2016). Adaptive computation time for recurrent neural networks. *CoRR*, abs/1603.08983.

Greff, K., Kaufman, R. L., Kabra, R., Watters, N., Burgess, C., Zoran, D., Matthey, L., Botvinick, M. M., and Lerchner, A. (2019). Multi-object representation learning with iterative variational inference. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2424–2433. PMLR.

Greff, K., Rasmus, A., Berglund, M., Hao, T. H., Valpola, H., and Schmidhuber, J. (2016). Tagger: Deep unsupervised perceptual grouping. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4484–4492.

Grusky, M., Naaman, M., and Artzi, Y. (2018). Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 708–719, New Orleans, Louisiana. Association for Computational Linguistics.

Gururangan, S., Swayamdipta, S., Levy, O., Schwartz, R., Bowman, S., and Smith, N. A. (2018). Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Gyllensten, A. C., Ekgren, A., and Sahlgren, M. (2019). R-grams: Unsupervised learning of semantic units in natural language. In *Proceedings of the 13th International Conference on Computational Semantics - Student Papers*, pages 52–62, Gothenburg, Sweden. Association for Computational Linguistics.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

Henderson, J. and Popa, D. (2016). A vector space for distributional semantics for entailment. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2052–2062, Berlin, Germany. Association for Computational Linguistics.

Hendrycks, D. and Gimpel, K. (2016). Gaussian error linear units (gelus). *CoRR*, abs/1606.08415.

Hermann, K. M., Kociský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701.

Hinton, G. E. (2021). How to represent part-whole hierarchies in a neural network. *CoRR*, abs/2102.12627.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O., and Sifre, L. (2022). Training compute-optimal large language models. *CoRR*, abs/2203.15556.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2020). The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Hua, X. and Wang, L. (2020). PAIR: Planning and iterative refinement in pre-trained transformers for long text generat ion. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 781–793, Online. Association for Computational Linguistics.

Huang, L., Cao, S., Parulian, N., Ji, H., and Wang, L. (2021). Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1419–1436, Online. Association for Computational Linguistics.

Huang, Y., Zhang, Y., Elachqar, O., and Cheng, Y. (2020). INSET: Sentence infilling with INter-SEntential transformer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2502–2515, Online. Association for Computational Linguistics.

Iida, R. and Tokunaga, T. (2012). A metric for evaluating discourse coherence based on coreference resolution. In *Proceedings of COLING 2012: Posters*, pages 483–494, Mumbai, India. The COLING 2012 Organizing Committee.

Ippolito, D., Grangier, D., Eck, D., and Callison-Burch, C. (2020). Toward better storylines with sentence-level language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7472–7478, Online. Association for Computational Linguistics.

Iskender, N., Polzehl, T., and Möller, S. (2021). Reliability of human evaluation for text summarization: Lessons learned and challenges ahead. In *Proceedings of the Workshop on Human Evaluation of NLP Systems (HumEval)*, pages 86–96, Online. Association for Computational Linguistics.

Jaegle, A., Gimeno, F., Brock, A., Vinyals, O., Zisserman, A., and Carreira, J. (2021). Perceiver: General perception with iterative attention. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 4651–4664. PMLR.

Jain, S. and Wallace, B. C. (2019). Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Jha, R., Bi, K., Li, Y., Pakdaman, M., Celikyilmaz, A., Zhiboedov, I., and McDonald, K. (2020). Artemis: A novel annotation methodology for indicative single document summarization. In *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*, pages 69–78, Online. Association for Computational Linguistics.

Jhamtani, H. and Berg-Kirkpatrick, T. (2020). Narrative text generation with a latent discrete plan. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3637–3650, Online. Association for Computational Linguistics.

Karimi Mahabadi, R., Mai, F., and Henderson, J. (2019). Learning entailment-based sentence embeddings from natural language inference. Idiap-RR Idiap-RR-20-2019, Idiap. project SNSF-LAOS.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. (2020). Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR.

Khan, S. H., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., and Shah, M. (2022). Transformers in vision: A survey. *ACM Computing Surveys*, 54(10s):200:1–200:41.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kipf, T., Elsayed, G. F., Mahendran, A., Stone, A., Sabour, S., Heigold, G., Jonschkowski, R., Dosovitskiy, A., and Greff, K. (2022). Conditional object-centric learning from video. In *The*

Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net.

Koh, H. Y., Ju, J., Zhang, H., Liu, M., and Pan, S. (2022). How far are we from robust long abstractive summarization? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2682–2698, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Koupaee, M. and Wang, W. Y. (2018). Wikihow: A large scale text summarization dataset. *CoRR*, abs/1810.09305.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114.

Kryscinski, W., Keskar, N. S., McCann, B., Xiong, C., and Socher, R. (2019). Neural text summarization: A critical evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 540–551, Hong Kong, China. Association for Computational Linguistics.

Kryscinski, W., McCann, B., Xiong, C., and Socher, R. (2020). Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346, Online. Association for Computational Linguistics.

Kryscinski, W., Rajani, N., Agarwal, D., Xiong, C., and Radev, D. (2022). BOOKSUM: A collection of datasets for long-form narrative summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6536–6558, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ladhak, F., Durmus, E., He, H., Cardie, C., and McKeown, K. (2022). Faithful or extractive? on mitigating the faithfulness-abstractiveness trade-off in abstrac tive summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1410–1421, Dublin, Ireland. Association for Computational Linguistics.

Ladhak, F., Li, B., Al-Onaizan, Y., and McKeown, K. (2020). Exploring content selection in summarization of novel chapters. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5043–5054, Online. Association for Computational Linguistics.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Li, N., Eastwood, C., and Fisher, R. B. (2020a). Learning object-centric representations of multi-object scenes from multiple views. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Li, W., Xiao, X., Liu, J., Wu, H., Wang, H., and Du, J. (2020b). Leveraging graph to improve abstractive multi-document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6232–6243, Online. Association for Computational Linguistics.

Li, X., Yin, X., Li, C., Zhang, P., Hu, X., Zhang, L., Wang, L., Hu, H., Dong, L., Wei, F., Choi, Y., and Gao, J. (2020c). Oscar: Object-semantics aligned pre-training for vision-language tasks. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXX*, volume 12375 of *Lecture Notes in Computer Science*, pages 121–137. Springer.

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Lin, H. and Bilmes, J. (2010). Multi-document summarization via budgeted maximization of submodular functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 912–920, Los Angeles, California. Association for Computational Linguistics.

Lin, J., Su, Q., Yang, P., Ma, S., and Sun, X. (2018). Semantic-unit-based dilated convolution for multi-label text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4554–4564, Brussels, Belgium. Association for Computational Linguistics.

Lin, Z., Wu, Y., Peri, S. V., Sun, W., Singh, G., Deng, F., Jiang, J., and Ahn, S. (2020). SPACE: unsupervised object-oriented scene representation via spatial attention and decomposition. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., and Shazeer, N. (2018). Generating wikipedia by summarizing long sequences. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Liu, Y., Fabbri, A. R., Liu, P., Zhao, Y., Nan, L., Han, R., Han, S., Joty, S. R., Wu, C., Xiong, C., and Radev, D. (2022a). Revisiting the gold standard: Grounding summarization evaluation with robust human evaluation. *CoRR*, abs/2212.07981.

Liu, Y., Gu, J., Goyal, N., Li, X., Edunov, S., Ghazvininejad, M., Lewis, M., and Zettlemoyer, L. (2020). Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Liu, Y. and Lapata, M. (2019a). Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, Florence, Italy. Association for Computational Linguistics.

Liu, Y. and Lapata, M. (2019b). Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Liu, Y., Liu, P., Radev, D., and Neubig, G. (2022b). BRIO: Bringing order to abstractive summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2890–2903, Dublin, Ireland. Association for Computational Linguistics.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019a). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Liu, Y., Titov, I., and Lapata, M. (2019b). Single document summarization as tree induction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1745–1755, Minneapolis, Minnesota. Association for Computational Linguistics.

Liu, Y., Wu, C., Tseng, S.-Y., Lal, V., He, X., and Duan, N. (2022c). KD-VLP: Improving end-to-end vision-and-language pretraining with object knowledge distillation. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1589–1600, Seattle, United States. Association for Computational Linguistics.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. (2020). Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Louizos, C., Welling, M., and Kingma, D. P. (2018). Learning sparse neural networks through l_0 regularization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Löwe, S., Lippe, P., Rudolph, M., and Welling, M. (2022). Complex-valued autoencoders for object discovery. *Transactions on Machine Learning Research*.

Lu, J., Batra, D., Parikh, D., and Lee, S. (2019). Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13–23.

Luong, T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Madsen, A. (2019). Visualizing memorization in rnns. *Distill*. https://distill.pub/2019/memorization-in-rnns.

Marfurt, A. and Henderson, J. (2021). Sentence-level planning for especially abstractive summarization. In *Proceedings of the Third Workshop on New Frontiers in Summarization*, pages 1–14, Online and in Dominican Republic. Association for Computational Linguistics.

Marfurt, A. and Henderson, J. (2022). Unsupervised token-level hallucination detection from summary generation by-products. In *Proceedings of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, pages 248–261, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Marfurt, A., Thornton, A., Sylvan, D., van der Plas, L., and Henderson, J. (2022). A corpus and evaluation for predicting semi-structured human annotations. In *Proceedings of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, pages 262–275, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Maynez, J., Narayan, S., Bohnet, B., and McDonald, R. (2020). On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Meister, C., Cotterell, R., and Vieira, T. (2020). If beam search is the answer, what was the question? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2173–2185, Online. Association for Computational Linguistics.

Merity, S., Keskar, N. S., and Socher, R. (2018). Regularizing and optimizing LSTM language models. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. (2017). Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Miculicich, L., Ram, D., Pappas, N., and Henderson, J. (2018). Document-level neural machine translation with hierarchical attention networks. In *Proceedings of the 2018 Conference on*

Empirical Methods in Natural Language Processing, pages 2947–2954, Brussels, Belgium. Association for Computational Linguistics.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

Miller, G. A. (1995). Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41.

Mitchell, M., van Deemter, K., and Reiter, E. (2010). Natural reference to objects in a visual domain. In *Proceedings of the 6th International Natural Language Generation Conference*. Association for Computational Linguistics.

Moniz, J. R. A. and Krueger, D. (2017). Nested lstms. In *Proceedings of The 9th Asian Conference on Machine Learning, ACML 2017, Seoul, Korea, November 15-17, 2017*, volume 77 of *Proceedings of Machine Learning Research*, pages 530–544. PMLR.

Nallapati, R., Zhai, F., and Zhou, B. (2017). Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 3075–3081. AAAI Press.

Nallapati, R., Zhou, B., dos Santos, C., Gulçehre, Ç., and Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Napoles, C., Gormley, M., and Van Durme, B. (2012). Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, pages 95–100, Montréal, Canada. Association for Computational Linguistics.

Narayan, S., Cohen, S. B., and Lapata, M. (2018a). Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Narayan, S., Cohen, S. B., and Lapata, M. (2018b). Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 1747–1759, New Orleans, Louisiana. Association for Computational Linguistics.

Narayan, S., Maynez, J., Adamek, J., Pighin, D., Bratanic, B., and McDonald, R. (2020). Stepwise extractive summarization and planning with structured transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4143–4159, Online. Association for Computational Linguistics.

Narayan, S., Simões, G., Zhao, Y., Maynez, J., Das, D., Collins, M., and Lapata, M. (2022). A well-composed text is half done! composition sampling for diverse conditional generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1319–1339, Dublin, Ireland. Association for Computational Linguistics.

Narayan, S., Zhao, Y., Maynez, J., Simões, G., Nikolaev, V., and McDonald, R. (2021). Planning with learned entity prompts for abstractive summarization. *Transactions of the Association for Computational Linguistics*, 9:1475–1492.

Nenkova, A. and Passonneau, R. (2004). Evaluating content selection in summarization: The pyramid method. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 145–152, Boston, Massachusetts, USA. Association for Computational Linguistics.

Nenkova, A., Passonneau, R., and McKeown, K. (2007). The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Trans. Speech Lang. Process.*, 4(2):4–es.

OpenAI (2023). GPT-4 technical report. *https://openai.com/research/gpt-4*.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., and Lowe, R. (2022). Training language models to follow instructions with human feedback. *CoRR*, abs/2203.02155.

Over, P. and Yen, J. (2003). An introduction to DUC-2003. In *Proceedings of the Document Understanding Conference (DUC)*, Edmonton, Canada.

Over, P. and Yen, J. (2004). An introduction to DUC-2004. In *Proceedings of the Document Understanding Conference (DUC)*, Boston, USA.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Pagnoni, A., Balachandran, V., and Tsvetkov, Y. (2021). Understanding factuality in abstractive summarization with FRANK: A benchmark for factuality metrics. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics:*

Human Language Technologies, pages 4812–4829, Online. Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Parker, R., Graff, D., Kong, J., Chen, K., and Maeda, K. (2011). English gigaword fifth edition. *Linguistic Data Consortium*.

Pasunuru, R. and Bansal, M. (2018). Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 646–653, New Orleans, Louisiana. Association for Computational Linguistics.

Pasunuru, R., Guo, H., and Bansal, M. (2017). Towards improving abstractive summarization via entailment generation. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 27–32, Copenhagen, Denmark. Association for Computational Linguistics.

Paulus, R., Xiong, C., and Socher, R. (2018). A deep reinforced model for abstractive summarization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Perez-Beltrachini, L., Liu, Y., and Lapata, M. (2019). Generating summaries with topic templates and structured convolutional decoders. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5107–5116, Florence, Italy. Association for Computational Linguistics.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Peyrard, M. (2019a). A simple theoretical model of importance for summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1059–1073, Florence, Italy. Association for Computational Linguistics.

Peyrard, M. (2019b). Studying summarization evaluation metrics in the appropriate scoring range. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5093–5100, Florence, Italy. Association for Computational Linguistics.

Poliak, A., Naradowsky, J., Haldar, A., Rudinger, R., and Van Durme, B. (2018). Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana. Association for Computational Linguistics.

Pöttker, H. (2003). News and its communicative quality: the inverted pyramid—when and why did it appear? *Journalism Studies*, 4(4):501–511.

Radev, D. R., Hovy, E., and McKeown, K. (2002). Introduction to the special issue on summarization. *Computational Linguistics*, 28(4):399–408.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training. *https://blog.openai.com/language-unsupervised*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *https://openai.com/blog/better-language-models*.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2016). Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Ravaut, M., Joty, S., and Chen, N. (2022). SummaReranker: A multi-task mixture-of-experts re-ranking framework for abstractive summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4504–4524, Dublin, Ireland. Association for Computational Linguistics.

Razumovskaia, E., Maynez, J., Louis, A., Lapata, M., and Narayan, S. (2022). Little red riding hood goes around the globe: Crosslingual story planning and generation with large language models. *CoRR*, abs/2212.10471.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 3856–3866.

Sajjadi, M. S. M., Duckworth, D., Mahendran, A., van Steenkiste, S., Pavetic, F., Lucic, M., Guibas, L., Greff, K., and Kipf, T. (2022). Object scene representation transformer. In *Advances in Neural Information Processing Systems*.

Sandhaus, E. (2008). The new york times annotated corpus. *Linguistic Data Consortium*.

Sarafianos, N., Xu, X., and Kakadiaris, I. A. (2019). Adversarial representation learning for text-to-image matching. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 5813–5823. IEEE.

Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ilic, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M., Tow, J., Rush, A. M., Biderman, S., Webson, A., Ammanamanchi, P. S., Wang, T., Sagot, B., Muennighoff, N., del Moral, A. V., Ruwase, O., Bawden, R., Bekman, S., McMillan-Major, A., Beltagy, I., Nguyen, H., Saulnier, L., Tan, S., Suarez, P. O., Sanh, V., Laurençon, H., Jernite, Y., Launay, J., Mitchell, M., Raffel, C., Gokaslan, A., Simhi, A., Soroa, A., Aji, A. F., Alfassy, A., Rogers, A., Nitzav, A. K., Xu, C., Mou, C., Emezue, C., Klamm, C., Leong, C., van Strien, D., Adelani, D. I., and et al. (2022). BLOOM: A 176b-parameter open-access multilingual language model. *CoRR*, abs/2211.05100.

Schick, T. and Schütze, H. (2021a). Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Compu tational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Schick, T. and Schütze, H. (2021b). It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Schuster, T., Fisch, A., Gupta, J., Dehghani, M., Bahri, D., Tran, V. Q., Tay, Y., and Metzler, D. (2022). Confident adaptive language modeling. In *Advances in Neural Information Processing Systems*.

Scialom, T., Dray, P.-A., Lamprier, S., Piwowarski, B., Staiano, J., Wang, A., and Gallinari, P. (2021). QuestEval: Summarization asks for fact-based evaluation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6594–6604, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Seitzer, M., Horn, M., Zadaianchuk, A., Zietlow, D., Xiao, T., Simon-Gabriel, C.-J., He, T., Zhang, Z., Schölkopf, B., Brox, T., and Locatello, F. (2023). Bridging the gap to real-world object-centric learning. In *International Conference on Learning Representations*.

Sellam, T., Das, D., and Parikh, A. (2020). BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Sharma, E., Li, C., and Wang, L. (2019). BIGPATENT: A large-scale dataset for abstractive and coherent summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2204–2213, Florence, Italy. Association for Computational Linguistics.

Shen, Y., Tan, S., Sordoni, A., and Courville, A. C. (2019). Ordered neurons: Integrating tree structures into recurrent neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Shrout, P. E. and Fleiss, J. L. (1979). Intraclass correlations: uses in assessing rater reliability. *Psychological bulletin*, 86(2):420.

Singh, G., Kim, Y., and Ahn, S. (2023). Neural systematic binder. In *The Eleventh International Conference on Learning Representations*.

Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum and weight decay. *US Naval Research Laboratory Technical Report*.

Smith, S., Patwary, M., Norick, B., LeGresley, P., Rajbhandari, S., Casper, J., Liu, Z., Prabhumoye, S., Zerveas, G., Korthikanti, V., Zheng, E., Child, R., Aminabadi, R. Y., Bernauer, J., Song, X., Shoeybi, M., He, Y., Houston, M., Tiwary, S., and Catanzaro, B. (2022). Using deepspeed and megatron to train megatron-turing NLG 530b, A large-scale generative language model. *CoRR*, abs/2201.11990.

So, D. R., Le, Q. V., and Liang, C. (2019). The evolved transformer. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5877–5886. PMLR.

So, D. R., Manke, W., Liu, H., Dai, Z., Shazeer, N., and Le, Q. V. (2021). Searching for efficient transformers for language modeling. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 6010–6022.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Sun, S., Shapira, O., Dagan, I., and Nenkova, A. (2019). How to compare summarizers without target length? pitfalls, solutions and re-examination of the neural summarization literature. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 21–29, Minneapolis, Minnesota. Association for Computational Linguistics.

Sundararajan, M., Taly, A., and Yan, Q. (2017). Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Sutton, R. (2019). The bitter lesson. *Incomplete Ideas (blog)*.

Suzgun, M., Scales, N., Schärli, N., Gehrmann, S., Tay, Y., Chung, H. W., Chowdhery, A., Le, Q. V., Chi, E. H., Zhou, D., and Wei, J. (2022). Challenging big-bench tasks and whether chain-of-thought can solve them. *CoRR*, abs/2210.09261.

Tan, H. and Bansal, M. (2018). Object ordering with bidirectional matchings for visual reasoning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 444–451, New Orleans, Louisiana. Association for Computational Linguistics.

Tan, H. and Bansal, M. (2019). LXMERT: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5100–5111, Hong Kong, China. Association for Computational Linguistics.

Tan, H. and Bansal, M. (2020). Vokenization: Improving language understanding with contextualized, visual-grounded supervision. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2066–2080, Online. Association for Computational Linguistics.

Tan, J., Wan, X., and Xiao, J. (2017). Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1181, Vancouver, Canada. Association for Computational Linguistics.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. (2023). Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):109:1–109:28.

Tay, Y., Dehghani, M., Tran, V. Q., Garcia, X., Bahri, D., Schuster, T., Zheng, H. S., Houlsby, N., and Metzler, D. (2022). Unifying language learning paradigms. *CoRR*, abs/2205.05131.

Taylor, R., Kardas, M., Cucurull, G., Scialom, T., Hartshorn, A., Saravia, E., Poulton, A., Kerkez, V., and Stojnic, R. (2022). Galactica: A large language model for science. *CoRR*, abs/2211.09085.

Tishby, N., Pereira, F. C., and Bialek, W. (1999). The information bottleneck method. In *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023). Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Tsvetkov, Y. and Wintner, S. (2011). Identification of multi-word expressions by combining multiple linguistic information sources. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 836–845, Edinburgh, Scotland, UK. Association for Computational Linguistics.

van Halteren, H. and Teufel, S. (2003). Examining the consensus between human summaries: initial experiments with factoid analysis. In *Proceedings of the HLT-NAACL 03 Text Summarization Workshop*, pages 57–64.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Vijayakumar, A., Cogswell, M., Selvaraju, R., Sun, Q., Lee, S., Crandall, D., and Batra, D. (2018). Diverse beam search for improved description of complex scenes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2692–2700.

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., and Titov, I. (2019). Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Voorhees, E. M. (2003). Overview of the TREC 2003 question answering track. In *Proceedings of The Twelfth Text REtrieval Conference, TREC 2003, Gaithersburg, Maryland, USA, November 18-21, 2003*, volume 500-255 of *NIST Special Publication*, pages 54–68. National Institute of Standards and Technology (NIST).

Wang, A., Cho, K., and Lewis, M. (2020). Asking and answering questions to evaluate the factual consistency of summaries. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5008–5020, Online. Association for Computational Linguistics.

Wang, A., Pang, R. Y., Chen, A., Phang, J., and Bowman, S. R. (2022). SQuALITY: Building a long-document summarization dataset the hard way. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1139–1156, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. (2018a). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Wang, J., Madhyastha, P. S., and Specia, L. (2018b). Object counts! bringing explicit detections back into image captioning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2180–2193, New Orleans, Louisiana. Association for Computational Linguistics.

Wang, R., Mao, J., Gershman, S., and Wu, J. (2021). Language-mediated, object-centric representation learning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2033–2046, Online. Association for Computational Linguistics.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. (2022). Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Weston, J., Chopra, S., and Bordes, A. (2015). Memory networks. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Wiegreffe, S. and Pinter, Y. (2019). Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China. Association for Computational Linguistics.

Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280.

Wiseman, S., Shieber, S., and Rush, A. (2017). Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao,

T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Wu, J., Ouyang, L., Ziegler, D. M., Stiennon, N., Lowe, R., Leike, J., and Christiano, P. F. (2021). Recursively summarizing books with human feedback. *CoRR*, abs/2109.10862.

Wu, J.-Y., Lin, Y.-J., and Kao, H.-Y. (2022). Unsupervised single document abstractive summarization using semantic units. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 954–966, Online only. Association for Computational Linguistics.

Xenouleas, S., Malakasiotis, P., Apidianaki, M., and Androutsopoulos, I. (2019). SUM-QE: a BERT-based summary quality estimation model. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6005–6011, Hong Kong, China. Association for Computational Linguistics.

Xie, S., Morcos, A. S., Zhu, S.-C., and Vedantam, R. (2022). COAT: Measuring object compositionality in emergent representations. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 24388–24413. PMLR.

Xin, J., Tang, R., Lee, J., Yu, Y., and Lin, J. (2020). DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251, Online. Association for Computational Linguistics.

Xu, J., Mello, S. D., Liu, S., Byeon, W., Breuel, T. M., Kautz, J., and Wang, X. (2022). Groupvit: Semantic segmentation emerges from text supervision. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 18113–18123. IEEE.

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., and Raffel, C. (2021). mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (2016). Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Yuan, W., Neubig, G., and Liu, P. (2021). Bartscore: Evaluating generated text as text generation. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 27263–27277.

Zhang, B., Titov, I., and Sennrich, R. (2021). On sparsifying encoder outputs in sequence-to-sequence models. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2888–2900, Online. Association for Computational Linguistics.

Zhang, J., Zhao, Y., Saleh, M., and Liu, P. J. (2020a). PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M. T., Li, X., Lin, X. V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P. S., Sridhar, A., Wang, T., and Zettlemoyer, L. (2022a). OPT: open pre-trained transformer language models. *CoRR*, abs/2205.01068.

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2020b). Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Zhang, T., Ladhak, F., Durmus, E., Liang, P., McKeown, K. R., and Hashimoto, T. B. (2023). Benchmarking large language models for news summarization. *CoRR*, abs/2301.13848.

Zhang, X., Liu, Y., Wang, X., He, P., Yu, Y., Chen, S., Xiong, W., and Wei, F. (2022b). Momentum calibration for text generation. *CoRR*, abs/2212.04257.

Zhao, Y., Khalman, M., Joshi, R., Narayan, S., Saleh, M., and Liu, P. J. (2023). Calibrating sequence likelihood improves conditional language generation. In *International Conference on Learning Representations*.

Zhong, M., Liu, P., Chen, Y., Wang, D., Qiu, X., and Huang, X. (2020). Extractive summarization as text matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6208, Online. Association for Computational Linguistics.

Zhong, Y. (2017). A theory of semantic information. *China Communications*, 14(1):1–17.

Zhou, C., Neubig, G., Gu, J., Diab, M., Guzmán, F., Zettlemoyer, L., and Ghazvininejad, M. (2021). Detecting hallucinated content in conditional neural sequence generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1393–1404, Online. Association for Computational Linguistics.

Zhou, Q., Yang, N., Wei, F., Huang, S., Zhou, M., and Zhao, T. (2018). Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th*

Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 654–663, Melbourne, Australia. Association for Computational Linguistics.

Zhu, L. and Yang, Y. (2020). Actbert: Learning global-local video-text representations. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 8743–8752. Computer Vision Foundation / IEEE.

Andreas Marfurt

Curriculum vitae

PhD candidate in NLP at Idiap Research Institute and EPFL,
Lecturer at Lucerne University of Applied Sciences and Arts

Education

11/2018–05/2023 **PhD candidate**, *Idiap Research Institute and EPFL*.
PhD candidate in the NLU group at Idiap with Dr. James Henderson

04/2016–03/2018 **PhD candidate**, *ETH Zürich*.
PhD candidate at the Institute for Machine Learning with Prof. Thomas Hofmann

09/2011–04/2013 **MSc**, *ETH Zürich*.
Master in Computer Science with a GPA of 5.58/6

09/2008–09/2011 **BSc**, *ETH Zürich*.
Bachelor in Computer Science with a GPA of 5.07/6

Research

supervisor Dr. James Henderson, `https://idiap.ch/~jhenderson/`

Idiap PhD thesis *Interpretable representation learning and evaluation for abstractive summarization*, focusing on summarization, hallucination detection and generation

ETH Zürich Sparse word embeddings, semantic document representations

Publications

profile `https://scholar.google.ch/citations?user=VdR7ECMAAAAJ`

papers A corpus and evaluation for predicting semi-structured human annotations, *Generation, Evaluation, and Metrics workshop (GEM), EMNLP 2022*

Unsupervised token-level hallucination detection from summary generation by-products, *Generation, Evaluation, and Metrics workshop (GEM), EMNLP 2022*

Sentence-level Planning for Especially Abstractive Summarization, *New Frontiers in Summarization workshop, EMNLP 2021*

Explaining away syntactic structure in semantic document representations, *arXiv 2018*

Andreas Marfurt
☎ +41 79 469 41 70 • ✉ andreas.marfurt@epfl.ch

Professional Experience

09/2022–now **Lecturer**, *Lucerne University of Applied Sciences and Arts*.
Teaching NLP and MLOps courses, supervising student projects and theses, responsible for the computation infrastructure for the AI/ML Bachelor's program.

01/2015–03/2016 **Software engineer**, *1plusX AG*.
First employee in machine learning startup. Tasks included:
- Creation and analysis of machine learning algorithms
- Design of system architecture for efficient and fault-tolerant processing of large data
- Implementation of end-to-end data processing pipeline with Apache Spark on AWS

07/2013–01/2015 **Research assistant**, *ETH Zürich*.
Main developer of the health data platform MIDATA.coop where users can import, visualize and control access to their personal health data.

Part-time jobs

07/2018–09/2018 **Data scientist**, *Binarium GmbH, Zürich*.
Analysis of a client's subscriber data for aspects of enlistment and retention

06/2011–08/2012 **Research assistant**, *ETH Zürich*.
Development of a new encryption algorithm for databases and comparison to existing encryption algorithms

Technical knowledge

python	Machine learning, deep learning with TensorFlow and PyTorch
scala	Large-scale data processing with Apache Spark
java	Master's thesis on Apache Hadoop and HDFS
aws	Cloud computing with Amazon Web Services (AWS)

Languages

german	**mother tongue**	
english	**proficient**	*Certificate of Proficiency in English (CPE), everyday use*
french	**intermediate**	*8 school years*

Extracurricular activities

03/2017–now Treasurer of local football club Celtic Lucerne FC

03/2012–03/2013 Board member of the computer science student's association at ETH Zürich, responsible for company relations

Interests

sports	Football, tennis, running, cross-country skiing
reading	Behavioral economics, decision theory, rationality
moocs	Game theory and gamification