EPFL

Thèse n° 10 383

Discovering meaningful units from text sequences

Présentée le 25 octobre 2024

Faculté de l'environnement naturel, architectural et construit
Intelligence Visuelle pour les Transports
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Melika BEHJATI

Acceptée sur proposition du jury

Prof. C. Gulcehre, président du jury
Prof. A. M. Alahi, Dr J. Henderson, directeurs de thèse
Prof. I. Titov, rapporteur
Prof. R. Sennrich, rapporteur
Prof. M. Jaggi, rapporteur

École polytechnique fédérale de Lausanne

2024

My knowledge has reached the point
where I know that I know nothing.
— Abu Ali Sina (Avicenna)

To my parents and my dearest,
Mohammadhossein. . .

Acknowledgements

I could hardly have imagined this day—the moment when I begin to acknowledge all the people who made this thesis possible.
First and foremost, I would like to express my deepest gratitude to my advisor, James Henderson, who patiently guided me through this long and challenging journey. From him, I not only learned how to conduct research but also how to think critically. His support and belief in my abilities gave me the confidence to keep moving forward, even during the most difficult moments. I am truly fortunate to have had the opportunity to learn from him, both professionally and personally.
I would also like to extend my sincere thanks to Alexandre Alahi for kindly accepting the role of my official supervisor at EPFL and for his support throughout these years.
I would like to extend my thanks to all my friends and colleagues at Idiap. In particular, I am deeply grateful to the members of the NLU & CCL labs: Andrei Coman, Fabio Fehr, Florian Mai, Laura Vasquez, Mete Ismayilzada, Andreas Marfurt, Rabeeh Karimi, Lesly Miculicich, Dina El Zein, Vincent Jung, Molly Peterson, and Lonneke van der Plas. A special thanks to Andrei and Fabio, who were not only colleagues but also kind friends. I truly appreciate the time we shared, from insightful scientific discussions to our fun coffee breaks.
I was fortunate to be a part of the NCCR Evolving Language project and grateful to have been funded by it. Through this community, I gained valuable insights into diverse fields of research, ranging from Primatology to Neuroscience. I would like to extend my thanks to the organizers of this project and to the group I closely collaborated with: the "Social Context" work package. In particular, I would like to acknowledge Miranda Dickerman and Joseph Mine, with whom I had interesting discussions about how our diverse research fields could intersect. A special thanks as well to Sabine Stoll and Simon Townsend, the principal investigators, for their leadership and support.
Pursuing a Ph.D. in a foreign country is not only about conducting research; it also involves leaving behind loved ones and adapting to an entirely different culture. I would like to express my gratitude to my parents for their lifelong support and encouragement. To my brothers, Ali and Amin, thank you for always supporting and understanding me. I am also deeply grateful to my parents-in-law and sister-in-law, Narges, for their constant support, as well as to my dear friends Tahereh, Sedreh, Mojgan, Howra, and Negar for being there for me.
Above all, to my husband, Mohammadhossein, whose unconditional support and belief in me kept me going, even in the toughest moments.

Lausanne, October 4, 2024 Melika Behjati

Abstract

In recent years, the field of Natural Language Processing has seen significant revolution by the introduction of Transformers, a stack of multiple layers of attention and non-linearity, capable of performing almost any task and the backbone for large foundation models. In contrast to traditional NLP pipelines, this architecture is able to learn the features required to perform a specific task without any assumptions about the structure of language. The only remaining hard-coded aspect is the way the text input is fed to these models. This preprocessing step, known as the tokenization step, divides the input into chunks which could be as fine-grained as bytes or as coarse-grained as words. The most popular approach is to use subwords, such as Byte Pair Encodings or word pieces which lie between characters and words. However, it has been shown that hard-coding the input representations, has its own drawbacks. In particular, it would lead to sub-optimal performance in downstream tasks. In addition, to perform different tasks we need different levels of representations. In this thesis, we define and address the novel task of inducing units from text in an unsupervised manner. This work is a step towards completely end-to-end models which can decide which level of representation is the most suitable for them to perform a specific task.

Our contributions are two-fold: First, we design models which are able to induce units without supervision at different levels. And second, since the task is novel, we need novel evaluations to show its effectiveness. Therefore, for every model we develop, we design and/or gather the set of tasks which evaluate and interpret the performance of our models.

In the first chapter, we design a model to induce morpheme-like units from a sequence of characters. We adapt a method from object discovery in vision, called Slot Attention for our purpose. We propose to evaluate this model by introducing bi-directional probing evaluation. In the second chapter, we design a model which induces word-like units from a sequence of characters by integrating non-parametric variational information bottleneck in the last layers of a transformer encoder. In the next chapter, we move to the multi-modal domain and starting from subwords, we design a model which induces phrases from image captions by aligning them to the objects in the image. Lastly, we explore a task-driven approach towards inducing entities.

Keywords: entity induction, object discovery, interpretability, grounding, representation learning, natural language understanding, NLP, deep learning, machine learning, artificial intelligence.

Résumé

Au cours des dernières années, le domaine du traitement automatique des langues naturel (TALN) a connu une importante révolution avec l'introduction des transformeurs, composés de multiples couches d'attention et de non-linéarité, capables de réaliser presque n'importé quelle tâche et constituant la base des grands modèles de fondation. Contrairement aux pipelines traditionnelles de TALN, cette architecture est capable d'apprendre les caractéristiques nécessaires pour réaliser une tâche spécifique sans aucune hypothèse sur la structure de la langue. Le seul aspect encore codé en dur est la manière dont le texte est introduit dans ces modèles. Cette étape de prétraitement, connue sous le nom d'étape de tokenisation, divise le text en morceaux qui peuvent être aussi petit que des octets ou aussi grand que des mots. L'approche la plus populaire consiste à utiliser des sous-mots, tels que les encodages "Byte-Pair-Encodings" ou les "word-pieces" qui se situent entre les caractères et les mots. Cependant, il a été démontré que la codification en dur des représentations le text présente ses propres inconvénients. Notamment, cela pourrait conduire à des performances sous-optimales dans les tâches en aval. De plus, pour réaliser différentes tâches, nous avons besoin de différents niveaux de représentations. Dans cette thèse, nous définissons et traitons la nouvelle tâche de l'induction d'unités à partir du texte de manière non supervisée. Ce travail est un pas vers des modèles "bout á bout" qui peuvent décider quel niveau de représentation leur est le plus approprié pour réaliser une tâche spécifique.

Nos contributions sont doubles : Premièrement, nous concevons des modèles capables d'induire des unités sans supervision à différents niveaux. Et deuxièmement, puisque la tâche est nouvelle, nous avons besoin de nouvelles évaluations pour démontrer son efficacité. Par conséquent, pour chaque modèle que nous développons, nous concevons et/ou rassemblons l'ensemble des tâches qui évaluent et interprètent la performance de nos modèles.

Dans le premier chapitre, nous concevons un modèle pour induire des unités semblables à des morphèmes à partir d'une séquence de caractères. Nous adaptons une méthode inspiré de la découverte d'objets en vision, appelée "Slot Attention" pour notre objectif. Nous proposons d'évaluer ce modèle en introduisant une évaluation de sondage bi-directionnelle. Dans le deuxième chapitre, nous concevons un modèle qui induit des unités semblables à des mots à partir d'une séquence de caractères en intégrant un bottleneck d'information variationnel non paramétrique dans les dernières couches d'un encodeur transformeur. Dans le chapitre suivant, nous passons au domaine multimodal et, en partant de sous-mots, nous concevons un modèle qui induit des phrases à partir de légendes d'images en les alignant sur les objets de l'image. Enfin, nous explorons une approche axée sur les tâches pour induire des entités.

Résumé

Mots-clés : induction d'entités, découverte d'objets, interprétabilité, ancrage, apprentissage de représentations, traitement automatique des langues naturel, TALN, apprentissage profond, apprentissage automatique, intelligence artificielle.

Contents

List of Figures

List of Tables

1 Introduction

Meaning is not a unique property of language, but a general characteristic of human activity ... We cannot say that each morpheme or word has a single or central meaning, or even that it has a continuous or coherent range of meanings ... there are two separate uses and meanings of language – the concrete ... and the abstract.
— Zellig S. Harris (Distributional Structure 1954)

1.1 Motivation

Prior to the deep learning era, feature engineering was a crucial step in designing models. This required humans to define and develop features of the input modality which were important, in their regard, in performing a specific task. With the rise of deep models trained on huge amounts of data, the need for engineering features has almost been eliminated. State-of-the-art models are able to learn generic, as well as task-specific, features without any prior assumptions.

However, one of the aspects which is still reliant on the designer's decision is how the input is fed to the model. In Natural Language Processing (NLP), this is the preprocessing step known as the *tokenization* step. The input sequence, which is an array of bytes or characters, passes through this step in order to output a more compressed version of it. While this is the dominant approach in most tasks, there is work suggesting to operate directly on bytes or characters. Feeding bytes or characters to the model has the advantage of allowing the model to learn the correlations between characters without any assumptions about the structure of the language. Though, this advantage brings its own drawbacks. First, the model should have more capacity in order to learn properties such as word boundaries which are trivial to humans (in most languages). Second, due to the limitations in compute, the model is able to operate on a shorter text context than a model working with words. As an example, if we have the sentence: "*John walked.*" it consists of 11 characters while including only 2 words. As a result of this limitation in the length of the context, the model is prone to missing longer dependencies between words and therefore, not learning the high-level semantics of a given text.

Tokenizing text sequences to words as units which carry individual meaning was popular in early NLP models. The major problem with this kind of tokenization is that since the vocabulary size of a model is limited, there will be words which are not in the vocabulary, known as OOV (Out Of Vocabulary). This problem is even more pronounced in agglutinative languages where new words can be generated by combining different morphemes. It will also limit the applicability of such models in real-life use cases such as machine translation. On the other hand, there are languages which do not have explicit word boundaries such as Chinese and Japanese where the words are inferred in the context.

Given words on one side and bytes or characters on the other side, tokenizing text into subwords was proposed as an intermediate solution. Subwords are groupings of contiguous characters or bytes, resulting in units which do not necessarily carry meaning, but can be combined to form a word. Word-pieces (Schuster and Nakajima, 2012) and their more recent variant sentence pieces (Kudo and Richardson, 2018a) are one line of obtaining such units. Byte Pair Encodings (BPEs) (Sennrich et al., 2015) use a frequency-based approach to this problem. These units resolve the OOV problem while reducing the character sequence length. While employing subwords is the standard approach in recent NLP models, including Large Language Models, there are some drawbacks to this method. They lead to sub-optimal representations of the input which needs the vocabulary size to be tuned (Gowda and May, 2020; Cao and Rimell, 2021), though it is often neglected. The model does not have information about the constituting characters of such units and it has to learn about them as part of its training (Kaushal and Mahowald, 2022). In addition, they are not robust to noise such as missing spaces or a character deletion or insertion (Belinkov and Bisk, 2017; Provilkov et al., 2020). The fact that tokenization into subwords is done as a preprocessing step, prior to training the model, constraints the model's performance. First in the sense that these units might not be the appropriate ones for a specific task and second, limiting their generalization to other domains (Cao and Rimell, 2021). Therefore, eliminating this step and integrating this step into the architectural design of the model itself will be beneficial.

In this thesis, we try to address the problem of inducing entities. That is, to let the model discover units which carry meaningful information and could lead to better representation of language for some tasks. Our contributions are two-fold. First, we propose different models which are able to induce units. Second, since this task is novel, we propose and/or gather different evaluation strategies which show the effectiveness of our models.

1.2 Contributions

Our contributions are towards answering the following research questions.

1.2.1 What is entity induction?

The topic of inducing entities as a standalone problem was not studied in the field when we started working on this thesis. To the best of our knowledge, we are the first to introduce this task and try to tackle it with different methodologies. We define the task as follows:

Given an input text sequence, which is represented at a specific granularity such as characters, subwords, or words, the goal is to represent this sequence in a more coarse-grained way such that the number of the resulting entities is less than the original input and every entity represents a more abstract concept. This should be performed in an unsupervised fashion, and that is why we use the term *induction* or *discovery* for it. For example, if we represent a sentence as a sequence of characters and a model is able to learn a representation of it which is at the level of subwords or words (more compact as well as more abstract), we refer to this model as an *entity induction model*.

1.2.2 Can we develop models that are capable of inducing entities?

We developed different models which are capable of inducing entities at different levels in this thesis.

In Chapter 2, we induce entities from a sequence of characters, where the entities are at the level of subwords. We learn our entities through creating a bottleneck in an auto-encoder. We first encode the sequence into a set of slots, where every slot is responsible for representing one entity. We adapt a module called Slot Attention (Locatello et al., 2020) from the vision domain for our purpose. This module is specifically designed to discover objects in an image via iterative top-down attention. We propose to initialize the slot vectors in a way which is more suitable for text inputs. Since the input sequence length varies among different examples, the number of entities should be dynamically adapted. Therefore, we utilize an L_0 regularization layer for pruning the unused entities. The stack of slots followed by the regularization layer creates our entity induction module. We show that our model is able to discover units which are at the same level as morphemes or subwords.

In Chapter 3, we induce entities similar to words, from a sequence of characters. Here, we take a different approach and utilize non-parametric variational information bottleneck (NVIB) layer (Henderson and Fehr, 2023) to encourage entity induction. We integrate this layer into the last Transformer encoder layers of an auto-encoder to induce sparse and smooth representations. We train our model with a noisy reconstruction objective and show that our model is able to induce units similar to words in its last encoder layer.

In Chapter 4, we try to address our task in a multi-modal setup. That is, given images and their captions, we aim to discover entities which are at the level of objects in the image, from the caption subwords. We learn to group the input tokens as part of the encoder architecture and then, align them to the objects in the image with a contrastive loss. As a result, we have a set of entities

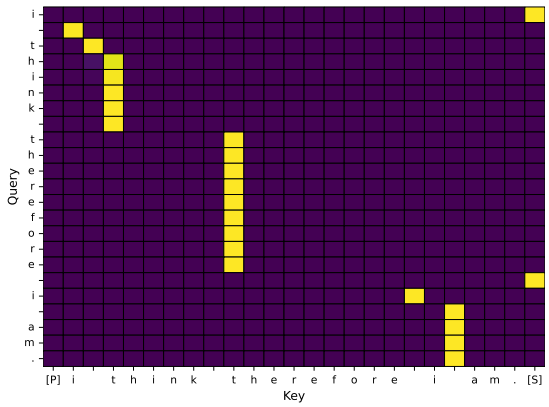


Figure 1.1: An example of the visualization of the attention maps in a model. For details refer to Chapter 3.

that are semantically grounded to the objects in the image and are at the level of phrases.

We explore a more direct task-driven approach towards discovering units from a character sequence in Chapter 5. We define the problem as predicting segmentation decisions with the goal of maximizing the generic downstream task of language modeling. In particular, we design two models: a segmentation model and a language model. The segmentation model predicts the subword boundaries and then, given the resulting subwords, we train the language model simultaneously. We utilize Reinforcement Learning to train our segmentation model. Our experiments using a supervised signal and also a fixed scorer for reinforcement learning demonstrate that the segmentation model can learn appropriate segment boundaries. However, we find that the signal from language modeling was not sufficient to learn good segmentations.

1.2.3 How can we evaluate the quality of the induced entities?

As stated above, we consider the task of inducing entities as a standalone problem. Since the task is novel and done without any supervised signal, we need to develop and gather different evaluation strategies to show the effectiveness of our proposed methods. In the following, we describe the strategies we find effective.

Visualization of the Attention maps

In order to evaluate the induced entities qualitatively, and since the Attention mechanism is the core operator in our models, visualization of the attention maps provides a good intuition about what the entities are corresponding to in a given model. In Figure 1.1, we visualize the resulting attention map after applying the last NVIB layer (see Chapter 3 for the details). We can observe that the vertical bands are approximately corresponding to words in the sequence. Therefore, this visualization can help us better understand and interpret what our model has learned. We utilize

this method in Chapters 2 to 4.

Greedy sequence segmentation

After inspecting the visualization of the Attention maps, we figured out that for some of our models, the vertical bands have clear boundaries. Hence, by taking the argmax over the attention weights, we greedily find segmentation boundaries of the input sequence. In Chapters 3 and 4, we obtain the resulting segmentation of the input sequence and then compute the overlap between the discovered segmentation and the annotations (for what we believe the units should be). For the example in Figure 1.1, we can compute the overlap between the discovered segments and words as targets.

Probing the induced entities

In the case of Chapter 2, the vertical bands' boundaries are not clear and fade towards the boundaries, therefore, we cannot evaluate them directly with greedy segmentation. Instead, we indirectly measure their correspondence to the units that we expect them to represent, with probing. We propose to do probing not only on predicting the target units from the entities but also to do it in the reverse direction. We call this approach *bi-directional probing*. The conventional way of probing measures to what extent the information we seek is available in the entities. However, since our goal is to show that the induced entities and the target units are at the same level of abstraction, we need to show that our induced units abstract away from extra information. Therefore, we do the probing from the target units to the induced entities to convey this.

1.2.4 Do these units lead to better representations of language?

In the previous section, we explained different strategies for intrinsically evaluating the quality of the induced units. In this section, we try to address the benefits of inducing units.

First, since our units are mainly obtained through creating a bottleneck and compression, we expect them to be more robust to noise than the initial input representation (see Chapter 3). That is, if we inject synthetic noise into the input such as deletion or insertion of tokens, how its performance get affected with and without entity induction.

Second, as the task definition describes, the induced entities are representing the sequence at a higher level of abstraction. Therefore, we expect them to be more linguistically informed than the original representation. We evaluate this by performing linguistic probing on the SentEval benchmark (Conneau and Kiela, 2018) (see Chapter 3).

Third, in order to take a more direct approach towards answering this question, we can probe the performance of our induced units in a downstream task. In Chapters 2 and 3, we evaluate our

induced units in a challenging ArXiv topic classification task (Hofmann et al., 2022) and show that they are more effective than utilizing the original character representation. In Chapter 4, we show that our induced entities have a better fine-grained knowledge about vision and language.

1.3 Outline

The thesis is structured into 5 additional chapters. In Chapter 2, we present our model for discovering morpheme-like units from character sequences with the use of an object discovery module. Then in Chapter 3, we discover word-like units from characters by integrating non-parametric variational information bottleneck into the last layers of a Transformer encoder. In Chapter 4, we take a multi-modal approach and group subwords into phrase-like segments which carry groundable information in the image. We explore a task-driven approach to discovering morpheme-segments from a sequence of characters in Chapter 5. Finally, in Chapter 6, we discuss our main findings through the thesis, and propose future directions for our work. We opt to discuss related work in each chapter, making them more specific to each contribution.

1.4 Publications

The following papers form the main content of this thesis.

1. M. Behjati, J. Henderson, *Inducing Meaningful Units from Character Sequences with Dynamic Capacity Slot Attention.*, Transactions on Machine Learning Research, 2023
2. M. Behjati [*], F. Fehr [*], J. Henderson, *Learning to Abstract with Nonparametric Variational Information Bottleneck.*, In Findings of EMNLP, 2023
3. M. Behjati, J. Henderson, *Discovering Meaningful Units with Visually Grounded Semantics from Image Captions*, under review, 2024

2 Discovering Meaningful Units with Dynamic Capacity Slot Attention

Characters do not convey meaning, but sequences of characters do. We propose an unsupervised distributional method to learn the abstract meaningful units in a sequence of characters. Rather than segmenting the sequence, our Dynamic Capacity Slot Attention model discovers continuous representations of the *objects* in the sequence, extending an architecture for object discovery in images. We train our model on different languages and evaluate the quality of the obtained representations with forward and reverse probing classifiers. These experiments show that our model succeeds in discovering units which are similar to those proposed previously in form, content and level of abstraction, and which show promise for capturing meaningful information at a higher level of abstraction.

This chapter is based on the following publication,

- M. Behjati, J. Henderson, *Inducing Meaningful Units from Character Sequences with Dynamic Capacity Slot Attention.*[1], Transactions on Machine Learning Research, 2023

[1] https://openreview.net/forum?id=m8U9rSs6gU

2.1 Introduction

When we look at a complex high-dimensional scene, we perceive its constituent objects, and their properties such as shape and material. Similarly, what we perceive when we read a piece of text builds on the word-like units it is composed of, namely *morphemes*, the smallest meaning-bearing units in a language. This chapter investigates deep learning models which discover abstract representations of such meaningful units from the distribution of character sequences in natural text.

In recent years, there has been an emerging interest in unsupervised object discovery in vision (Eslami et al., 2016; Greff et al., 2019; Engelcke et al., 2020; Elsayed et al., 2022; Seitzer et al., 2023). The goal is to segment the scene into its objects without supervision and obtain an object-centric representation of the scene. These representations should lead to better generalization to unknown scenes, and additionally should facilitate abstract reasoning over the image. Most of this work achieves its goal via modeling the scene as a composition of objects and learning the latent object representations through an auto-encoding objective. Locatello et al. (2020) proposed a relatively simple and generic algorithm for discovering objects called Slot Attention, which iteratively finds a set of feature vectors (i.e., slots) which can bind to any object in the image through a form of attention.

Inspired by this line of work in vision, our task is to learn a set of abstract continuous representations of the *objects* in text. We adapt the Slot Attention module (Locatello et al., 2020) for this purpose, extending it for discovering the meaningful units in natural language character sequences. This makes our proposed task closely related to unsupervised morphology learning (Creutz, 2003; Narasimhan et al., 2015; Eskander et al., 2020). However, there are fundamental differences between our task and morphology learning. First, we learn to represent a text with a set of vectors, which are not explicitly tied to the text segments. Second, our model learns its representations by considering the entire input sentence, rather than individual space-delimited words. These properties of our induced representations make our method appropriate for inducing meaningful units as part of deep learning models.

In particular, we integrate our unit discovery method on top of the encoder in a Transformer auto-encoder (Vaswani et al., 2017a), as depicted in Figure 2.1, and train it with an unsupervised sentence reconstruction objective. This setting differs from previous work on Slot Attention, which has been tested on synthetic image data with a limited number of objects (Locatello et al., 2020). We propose several extensions to Slot Attention for the domain of real text data. We increase the capacity of the model to learn to distinguish a large number of textual units, and add the ability to learn how many units are needed to encode sequences with varying length and complexity. Thus, we refer to our method as Dynamic Capacity Slot Attention. Additionally, as a hand-coded alternative, we propose stride-based models and compare them empirically to our slot attention based models.

To evaluate the induced representations, we both qualitatively inspect the model itself and

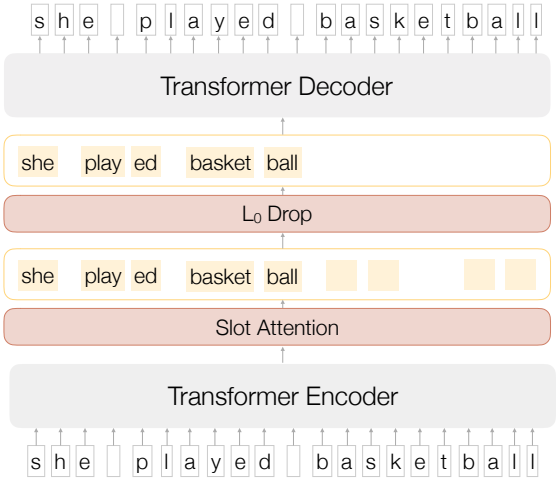


Figure 2.1: Dynamic Capacity Slot Attention. First, the Transformer encoder encodes the sequence and then, Slot Attention computes the slot vectors (highlighted text). Next, the L_0Drop layer dynamically prunes out the unnecessary slots and the decoder finally reconstructs the original sequence.

quantitatively compare it to previously proposed representations. Visualisation of its attention patterns shows that the model has learned representations similar to the contiguous segmentations of traditional tokenization approaches. Trained probing classifiers show that the induced units capture similar abstractions to the units previously proposed in morphological annotations (MorphoLex (Sánchez-Gutiérrez et al., 2018; Mailhot et al., 2020)) and tokenization methods (Morfessor (Virpioja, 2013), BPE (Sennrich et al., 2016)). To compare two representations, we propose to do this probing evaluation in both directions, to compare both informativeness and abstractness of the units. In addition, we show the potential benefits of our model in a downstream classification task. These evaluations show promising results in the ability of our models to discover units which capture meaningful information at a higher level of abstraction than characters.

To summarize, our contributions are as follows:

(i) We propose a novel model for learning meaningful units from a sequence of characters (Section 2.2).

(ii) We propose simple stride-based models which serve as strong baselines for evaluating such unsupervised models (Section 2.2.3).

(iii) We analyze the induced units by visualizing the attention maps of the decoder over the slots and observe the desired sparse and contiguous patterns (Section 2.4.4).

(iv) We propose a bi-directional probing method to evaluate such models by comparing the induced units to given units in terms of both their informativeness and their abstractness (Section 2.2.4).

(v) We show that our induced units capture meaningful information at an appropriate level of abstraction by probing their equivalence to previously proposed meaningful units (Section 2.4.5).

(vi) We evaluate the induced units in a downstream task to show its potential benefits over character-level and segmentation-based representations (Section 2.4.6).

2.2 Approach

The unsupervised induction of abstract meaningful units from sequences of characters is a novel task which requires both novel methods and novel evaluations. It should allow the induction of meaningful units as part of deep learning models. In this section we define the task, propose two types of models, and propose a probing-based evaluation.

2.2.1 Problem Formulation

Given a sequence of N characters $X = x_1 x_2 \dots x_N$, we want to compute a set of vectors $M = \{m_1, \dots, m_K\}$ which best represents X at a higher level of abstraction. We expect that each vector (i.e. slot) in the set M should represent a meaning-bearing unit in the text X, so that the induced representation M captures the desired level of abstraction. As an example, consider the sequence *"she played basketball"*, where we expect each of our vectors to represent one of the morphemes of the sequence, namely {*she, play, -ed, basket, -ball*}. Although the meaning-bearing units are often assumed to come from a fixed vocabulary, we do not assume this and compute the vectors dynamically from the full text.

2.2.2 Dynamic Capacity Slot Attention

We learn our representations through encoding the input sequence into slots and then reconstructing the original sequence from them. In particular, we use an auto-encoder structure where slots act as the bottleneck between the encoder and the decoder. Figure 2.1 shows an overview of our proposed model, Dynamic Capacity Slot Attention. First, we encode the input character sequence by a Transformer encoder (Vaswani et al., 2017a), which gives us one vector per character. Then, we apply our higher-capacity version of a Slot Attention module (Locatello et al., 2020) over the encoded sequence, to learn the slots. Intuitively, Slot Attention will learn a soft clustering over the input where each cluster (or respectively slot) corresponds to a candidate meaningful unit in the sequence. To select which candidates are needed to represent the input, we integrate an L_0 regularizing layer, i.e., L_0Drop layer (Zhang et al., 2021), on top of the slots. Since the maximum number of slots is fixed during the course of training, this layer ensures that the model only uses as many slots as necessary for the particular input. This stops the model from converging to trivial solutions for short inputs, such as passing every character through a separate slot. Finally, the Transformer decoder reconstructs the input sequence autoregressively using attention over

Algorithm 1 Slot Attention module (Locatello et al., 2020). q, k, v map the slots and inputs to a common dimension D and T denotes the number of iterations.

Require: inputs $\in \mathrm{R}^{N \times D_{input}}$, slots $\sim \mathcal{N}(\mu, diag(\sigma)) \in \mathrm{R}^{K \times D_{slots}}$
  inputs = LayerNorm(inputs)
  for i = 1 **to** T **do**
    slots_prev = slots
    slots = LayerNorm(slots)
    attn = Softmax($\frac{1}{\sqrt{D}}$ k(inputs).q(slots)^T, axis = 'slots')
    updates = WeightedMean(weights = attn + δ , values = v(inputs))
    slots = GRU(states = slots_prev, input = updates)
    slots += MLP(LayerNorm(slots))
  end for
  return slots

the set of slots.

Encoder

We use the Transformer encoder architecture for encoding our sequence (Vaswani et al., 2017a) and It consists of stacked layers of self-attention and non-linearity for building a new representation of the input sequence. We obtain the representation $X' = x'_1 x'_2 \dots x'_N$ from our input sequence X.

Slot Attention for Text

After encoding the character sequence, we use our extended version of Slot Attention for discovering meaningful units of the input character sequence. Slot Attention is a recent method for unsupervised object representation learning in vision (Locatello et al., 2020). It learns a set of feature vectors (slots) by using an iterative attention based algorithm. Algorithm 1 shows the pseudo code of this method. Abstractly, in every iteration, it takes the following steps. First, it computes an attention map between the slots and the inputs, with slots acting as queries and inputs as keys. Then, it normalizes the attention map over the slots, which makes the slots compete for representing each token of the input. Afterwards, it computes the slots' updates as the (normalized) weighted mean over the attention weights and the input values. Finally, it updates the slots through a Gated Recurrent Unit (GRU) (Cho et al., 2014) followed by a residual MLP. This process iterates a fixed number of times.

In Locatello et al. (2020), the slots are initialized randomly by sampling from a Normal distribution with shared learnable parameters μ and σ, i.e.,

$$\mathrm{slot}_i \sim \mathcal{N}(\mu_{\mathrm{shared}}, \sigma_{\mathrm{shared}}). \tag{2.1}$$

In other words, the initial value of slots are independent samples of a single Normal distribution with learnable parameters. We found in our experiments that this initialization method does not lead to good results in the text domain (see Section 2.4.2 for a comparison). In contrast with the experimental setting of Locatello et al. (2020), which used artificially generated data with a small number and range of objects, real language requires learning about a very large vocabulary of morphemes, and each example can have a large number of morphemes. This suggests that a model for language needs a more informative initialization with more trainable parameters, in order to have the capacity to learn about this large vocabulary and distinguish all the objects in a long sentence.

To investigate this issue, we propose another initialization for adapting Slot Attention to text. We consider a separate learnable μ per slot and we fix the σ to a predefined value for all the slots. Namely, the slots are initialized as

$$\text{slot}_i \sim \mathcal{N}(\mu_i, \sigma_{\text{constant}}). \tag{2.2}$$

By assigning a separate μ for each slot, the initialization has many more trainable parameters. This allows the model to learn about different kinds of units, such as ones that occur at different positions, or ones that have different types of forms, but we do not make any assumptions about what those differences might be. In addition, the intuition behind fixing the σ is to prevent it from collapsing to zero. In particular, since the number of possible n-grams in text is finite but the slots can have any continuous value in the space of $\mathrm{R}^{D_{slots}}$, the slots tend to learn an arbitrary mapping from n-grams in the input to the slots while turning σ to zero. In this case, there is no need for the slots to learn the underlying meaning-bearing units. By fixing the σ to a nonzero value, the initialization effectively introduces noise into slot vectors and thereby limits the amount of information which can be passed through each slot, from the information theoretic point of view. This bottleneck forces the slots to compress information in a meaningful way. As an alternative to having trained parameters for each slot, we could still distinguish all the slots at initialization simply using position embeddings. This would hand-code the assumption that meaningful units are distributed across the string and are local in the string. Thus, we try an alternative initialization of the following form.

$$\text{slot}_i \sim \mathcal{N}(\mu_{shared} + \text{PE}(i), \sigma_{shared}), \tag{2.3}$$

where $\text{PE}(i)$ is the positional encoding of the ith slot, using the same positional encoding function as inputs and distributing the slots evenly across the input sequence. With this formulation, we explicitly inject positional information into the slots. Moreover, we let the model to learn a shared σ, instead of fixing it, to increase the capacity of our model, since μ is also shared among all the slots.

Finally, We then obtain the set of slots $M = \{m_1 \dots m_K\} = \text{SlotAttention}(X')$.

Neural Sparsification Layer: L_0Drop

The number of units needed to represent a sequence varies among different sequences in the data. In contrast to the object discovery work where the data is generated synthetically and thereby, the number of objects in the scene is known beforehand, we do not make any assumptions about the number of units required. Therefore, we consider an upper-bound over the number of required units and prune the extra ones per input sequence. We accomplish this goal by using a neural sparsification layer called L_0Drop (Zhang et al., 2021). It allows our model to dynamically decide on the number of required units for every input sequence.

This layer consists of stochastic binary-like gates $\boldsymbol{g} = g_1 \dots g_K$ that for every input m_i works as

$$L_0\text{Drop}(m_i) = g_i m_i. \tag{2.4}$$

When g_i is zero the gate is closed and when it is one the whole input is passed. Each gate is a continuous random variable in the $[0, 1]$ interval, sampled from a hard-concrete distribution (Louizos et al., 2018). This distribution assigns most of its probability mass over its endpoints (i.e., 0 and 1) in favour of the sparsification goal. Specifically,

$$g_i \sim \text{HardConcrete}(\alpha_i, \beta, \epsilon), \tag{2.5}$$

where β and δ are the hyperparameters of the distribution. Moreover, α_i is obtained as A sample g_i from this distributaion is obtained from stretching and rectifying a sample from the BinaryConcrete distribution (Maddison et al., 2017; Jang et al., 2017):

$$\begin{aligned} s_i &\sim \text{BinaryConcrete}(\alpha_i, \beta), \\ \bar{s_i} &= s_i(1 + 2\epsilon) - \epsilon, \\ g_i &= \min(1, \max(0, \bar{s_i})), \end{aligned} \tag{2.6}$$

The BinaryConcrete variable s_i in $(0, 1)$ is stretched to $(-\epsilon, 1+\epsilon)$ and then a hard sigmoid function is applied to rectify it into the interval of $[0, 1]$.

α_i is predicted as a function of the encoder output m_i, i.e.,

$$\log \alpha_i = m_i w^T, \tag{2.7}$$

where w is a learnable vector. This allows the model to dynamically decide which inputs to pass and which ones to prune.

The $\mathscr{L}_0$ penalty, which yields the expected number of open gates, is computed as:

$$\mathscr{L}_0(M) = \sum_{i=1}^{k} 1 - p(g_i = 0 | \alpha_i, \beta, \epsilon), \tag{2.8}$$

where the probability of g_i being exactly 0 is provided in closed form in Louizos et al. (2018). We

follow the same approach as Louizos et al. (2018) at evaluation time and consider the expectation of each gate as its value,

$$\hat{g}_i = \min(1, \max(0, \sigma(\log \alpha_i)(1 + 2\epsilon) - \epsilon)). \tag{2.9}$$

We refer to the pruned slots after applying the L_0Drop layer as $M' = m'_1 \dots m'_K$.

Decoder

Lastly, we regenerate the input sequence from the set of slots by using a simple, shallow decoder. To this end, we use a one-layer Transformer decoder (Vaswani et al., 2017a) with a single attention head over the slots. A simple decoder forces the slots to learn representations with a straightforward relationship to the input, which we expect to be more meaningful. In other words, we do not use a powerful decoder because it would be able to decode even low-quality representations of the input, which are less meaningful (Bowman et al., 2015).

Training Objective

We train our model end-to-end by using Gumble trick for sampling HardConcrete variables (Maddison et al., 2017; Jang et al., 2017). The training objective is

$$\begin{aligned} \mathscr{L}_{\text{rec}}(X, M') + \lambda\, \mathscr{L}_0(M) &= -\log\left(\mathrm{E}_{\boldsymbol{g}}[p(X|M')]\right) + \lambda\, \mathscr{L}_0(M) \\ &\leq \mathrm{E}_{\boldsymbol{g}}[-\log p(X|M')] + \lambda\, \mathscr{L}_0(M) \\ &= \mathscr{L}(X), \end{aligned} \tag{2.10}$$

which consists of the reconstruction loss from the decoder (L_{rec}); equivalent to the Cross Entropy of the predictions and the input; and the L_0 penalty for the open gates. Hyperparameter λ, the sparsification level, controls the trade-off between the two losses. In practice, we find that in order to impose enough sparsity in the slots, we should slightly increase λ during the course of training using scheduling techniques.

Training Objective with Targeted Sparsity

Controlling the level of sparsity by setting λ can be difficult, so we provide an alternative version of the loss with an explicit target sparsity rate r (Mai and Henderson, 2022). We replace the L_0 term in Equation 2.10 with $\max(L_0, {}^1\!/_r \times N)$, to have

$$\mathscr{L}(X) = \mathrm{E}_{\boldsymbol{g}}[-\log p(X|M')] + \lambda \max(\mathscr{L}_0(M), {}^1\!/_r \times N) \tag{2.11}$$

where $1/r$ indicates the proportion of slots that we want to keep open and N is the maximum input length. In this setup, the model will stop closing more gates when L_0 reaches the target $1/r \times N$ [2].

2.2.3 Stride-based Models

We propose a simple hand-crafted alternative model to our induced units which can gain acceptable results in terms of performance. We design this model by replacing our Dynamic Capacity Slot Attention module with a linear-size bottleneck. In particular, we take 1 out of every k encoder outputs and down project them ($\mathrm{R}^{D_{input}} \rightarrow \mathrm{R}^{D_{slots}}$) to obtain the representations. In other words, we only pass certain encoder outputs based on their position and drop the rest.

$$M = \text{DownProject}(x'_1 x'_{k+1} x'_{2k+1} \dots x'_{nk+1})$$

where $n = \lfloor \frac{N-1}{k} \rfloor$. We can get different alternative models by varying the stride k. The training objective is the reconstruction loss $\mathscr{L}_{\text{rec}}(X, M) = -\log\, p(X|M)$. The idea of using stride-based models has also been used in Clark et al. (2022); Tay et al. (2022) in a different context and setup.

2.2.4 Bi-directional Probing Evaluation

Since the task is unsupervised and we do not evaluate using artificially generated data, there is no obvious gold standard for what units a method should learn. To quantitatively evaluate how well a model performs, we freeze the trained model and train probing classifiers to measure to what extent the discovered units capture the same information as previously proposed meaningful units. We use multiple previously proposed representations which have been shown to be good levels of abstraction for a range of NLP applications. If the induced representation separates the required information into units with a similar level of abstraction, then we can expect that they will also be effective in NLP applications.

We propose to measure whether two representations provide the same level of abstraction by measuring whether there is a one-to-one mapping between their respective units such that two aligned units contain the same information, meaning that each unit can be predicted from the other. Predicting the previously proposed unit from our induced unit is a probing task, as used in previous work (Belinkov and Glass, 2019), which we specify as *forward probing*. We refer to the prediction of our induced unit from the previously proposed unit as a *reverse probing* task, which we believe is a novel form of evaluation[3]. We compare various models on their trade-off between forward and reverse probing.

[2] Note that this term would be effective when $1/r \times N \leq K$, where K is the maximum number of slots.

[3] See Figure A.4 in the Appendix for an illustration of forward and reverse probing

Forward probing

This is the common way of probing where we want to measure if our induced units include the information about the target representation. We train a classifier with shared parameters on each of our slots individually and obtain a *set* of predictions $\{f(m'_1), f(m'_2), \ldots, f(m'_K)\}$, one per slot. As we are dealing with a set, we need to find a one-to-one matching between the classifier's predictions and the target tokens. Therefore, we follow Locatello et al. (2020) to use the Hungarian matching algorithm (Kuhn, 1955) for finding the match which minimizes the classification loss. The same alignment method is applied both at training and at testing time.

We consider the complete set of slots after applying the L_0Drop layer as the inputs to our classifier. Slots whose L_0Drop gate is closed are simply input as zero vectors. This gives us a fixed number of vectors. The two sides of matching should have the same size to obtain a one-to-one match, therefore, we add an extra target label (i.e., *empty*) for representing the pruned slots. Due to the fact that many slots are pruned out, considering a measure like accuracy could be misleading, since a classifier which outputs *empty* label will achieve very high accuracy. Therefore, we build a confusion matrix as follows. We consider all *non-empty* labels as positive and the *empty* ones as negative, and we report precision (P), recall (R) and F1 measure, to better reflect what the slots have learned.

Reverse probing

To evaluate whether the induced units capture the same level of abstraction, we also need to evaluate whether the induced units abstract away from the same information as the previously proposed units. We propose to measure this by training reverse probing classifiers, which predict each induced unit from its aligned target unit. Because the induced unit is a continuous vector, we predict the parameters of a d-dimensional Gaussian distribution with diagonal covariance, and measure the density function of the induced vector in this distribution. The probe is trained to minimise this negative-log-density function, and this loss on the test set is used as our evaluation measure.[4]

2.3 Related Work

Unsupervised Object Discovery.

There is a recent line of research in the image domain for discovering objects in a scene without explicit supervision, and building an object-centric representation of them. Most of this work is built around the idea of compositionality of the scenes. MONet (Burgess et al., 2019) and GENESIS (Engelcke et al., 2020) similarly use a recurrent attention network for learning the location masks of the objects. Greff et al. (2016, 2017, 2019); Van Steenkiste et al. (2018);

[4]Note that as we are modelling a continuous distribution, the density function can have values higher than 1 and therefore, the negative-logarithm of it could be negative.

Emami et al. (2021) model the scene as a spatial Gaussian mixture model. Furthermore, AIR network (Eslami et al., 2016) and its variants (Crawford and Pineau, 2019; Lin et al., 2020) model objects from a geometric perspective by defining three specific latent variables. Lately, Locatello et al. (2020) propose an attention-based algorithm (namely Slot Attention) to learn object representations (slots) which have been followed by Singh et al. (2022); Sajjadi et al. (2022); Seitzer et al. (2023); Singh et al. (2023a) and Kipf et al. (2022); Elsayed et al. (2022) further expanded it to the video domain. In contrast to this line of work in vision, our approach is specifically designed for text. We use additional components (e.g., L_0Drop layer) in our architecture to resolve the requirements of modeling textual data. Furthermore, our model is trained and evaluated on real text datasets, in contrast to these previous models which have only been shown to be effective on synthetic scenes.

Unsupervised morphology learning.

This subject has been of interest for many years in the NLP field (Elman, 1990; Creutz and Lagus, 2002; Baroni et al., 2002). Morphemes have strong linguistic motivations, and are practically important in many downstream tasks because they are the smallest meaning-bearing units in a language (Can and Manandhar, 2014). Many approaches have been proposed for discovering the underlying morphemes or morpheme segmentations. Morfessor variants are based on probabilistic machine learning methods (MDL, ML, MAP) for morphological segmentation (Creutz, 2003; Creutz and Lagus, 2002, 2005, 2007; Virpioja, 2013). Some researchers take a Bayesian approach for modeling word formation (Poon et al., 2009; Narasimhan et al., 2015; Bergmanis and Goldwater, 2017; Luo et al., 2017). Adaptor Grammars are another approach for modeling morphological inflections (Sirts and Goldwater, 2013; Eskander et al., 2016, 2019, 2020). In addition, Xu et al. (2018a, 2020) built their models upon the notion of paradigms, set of morphological categories that can be applied to a set of words. Moreover, Soricut and Och (2015a); Üstün and Can (2016) extract morphemes by considering the semantic relations between words in the continuous embedding space. Cao and Rei (2016) propose to learn word embeddings by applying a bi-directional RNN with attention over the character sequence. Furthermore, Ataman et al. (2020a) model word formation as latent variables which mimic morphological inflections in the task of machine translation.

Our work differs from the previous work in classical morphology learning in two ways. First, instead of explicitly discovering morphemes, we learn a set of continuous vector representations of the input, which would then need to be processed to extract the morphemes. The model itself has no explicit relation between these unit representations and segments of the input. Second, our model learns representations of an entire input sentence, rather than individual space-delimited words. This makes fewer assumptions about morphemes, and considers the context of the words in a sentence. Our work is similar to Ataman et al. (2020a) in modeling morphology implicitly in the latent space. However, we employ a self-supervised objective for our purpose which is more general compared to their supervised loss, as we do not need labeled data.

Unsupervised character segmentation.

Learning to segment a character sequence in unsupervised fashion is another relevant area to our work. Chung et al. (2017a) propose Hierarchical Multi-scale RNNs for modeling different levels of abstractions in the input sequence. In the language modeling task, they observe that the first layer is roughly segmenting the sequence into words, namely at space boundaries. Sun and Deng (2018) propose Segmental Language Models for Chinese word segmentation. Moreover, in (Kawakami et al., 2019), the authors design a model to learn the latent word segments in a whitespace-removed character sequence with a language modeling objective. As we mentioned earlier, we learn continuous vector representations of text which is different from explicitly detecting discrete character segments.

Learning intermediate representations from characters.

Recently, a line of work has been proposed to model language at the level of characters and then aggregate the characters into subword-like units and have the core Transformer layers on top of them. CANINE (Clark et al., 2022) learns these units by applying local attention and strided convolutions and proves the usefulness of their model in multilingual tasks. Charformer (Tay et al., 2022) learns a representation for each character by computing a weighted sum over character k-grams that this character is involved in and then performs mean pooling to downsample the representation size. The goal of our works differs from theirs as we are focusing on learning meaning-bearing units at the level of morphemes and we are not focusing on improving downstream tasks' performance directly.

Unsupervised speech unit discovery.

Unsupervised unit discovery has also been studied in the speech domain, where the speech data includes both acoustic and language information. Wav2vec 2.0 (Baevski et al., 2020) encodes the raw input using a convolutional neural network. Then, it learns units in the representation by discretizing the output of the feature encoder using product quantization (Jegou et al., 2010). HuBERT (Hsu et al., 2021) follows the same architecture as Wav2vec 2.0 but they discover acoustic units leveraging K-means algorithm. Authors in (Tjandra et al., 2020) extract subword units given speech audio by a variational autoencoder with a finite set of discrete latent variables (aka limited codebook) called vector quantized variational autoencoder (VQ-VAE) (Van Den Oord et al., 2017).

Subword discovery algorithms.

This set of algorithms have become a standard component of NLP models in recent years. Byte-Pair-Encoding (BPE) (Sennrich et al., 2016) iteratively merges the two consecutive tokens with the highest frequency. Word-piece (Schuster and Nakajima, 2012), sentence-piece (Kudo

and Richardson, 2018a) and unigram LM (Kudo, 2018) are other similar subword tokenization algorithms. In contrast to these methods, which mostly use local statistical information of the data, our model is trained over complete sentences to learn an abstract sophisticated representation.

2.4 Experiments

We train our models as auto-encoders and evaluate the resulting induced units. In addition to considering the ability of the induced representations to reconstruct the input (their supervised objective), we evaluate our unsupervised model both by visualizing attention maps to show what characters the slots correspond to (Section 2.4.4), and by bi-directional probing of the slot vectors (Section 2.4.5). In addition, we show the potential usefulness of our induced slots in a downstream task (Section 2.4.6).

2.4.1 Experimental Setup

Languages and training data. We apply our model to languages from different morphological typologies. We select English (EN), German (DE), French (FR), Spanish (ES) and Czech (CS) from the fusional family and Finnish (FI) from the agglutinative typology. For English we use the raw Wikitext2 dataset (Merity et al., 2016). For the rest, we use Multilingual Wikipedia Corpus (MWC) (Kawakami et al., 2017).

Probing data. We consider three target representations which either have linguistic or practical evidence of being effective in NLP applications. For the languages for which an in-depth linguistic morphological analysis is available, we compare to these morphemes (i.e., EN and FR) (Žabokrtský et al., 2022; Sánchez-Gutiérrez et al., 2018; Mailhot et al., 2020). Alternatively, as linguistically inspired units, we have the Morfessor (Virpioja, 2013) outputs, and BPEs (Sennrich et al., 2016) as frequency-based subwords.

Hyperparameters. As for the models, we use a standard Transformer architecture (Vaswani et al., 2017a) with model dimension 256. The encoder consists of 2 layers with 4 self-attention heads and the decoder consists of 1 layer with 1 self-attention head and 1 attention head over the slots. We use the same sinusoidal positional encoding function as in (Vaswani et al., 2017a). We feed in the sentences with less than 128 characters to our model and consider the maximum number of slots as 64 (half of the maximum input length). In addition, we take the dimension of slots as 128. We initialized the slots according to equation (2.2) in all the experiments except for Section 2.4.2, since it was the most effective form of initialization. We scheduled the λ parameter in the training loss to start with a low value and exponentially increase it every 10 epochs until it reaches a certain limit. We obtain this limit manually in a way that the final number of open gates roughly equals the average number of BPE tokens in a sequence[5]. Finally, we used Adam

[5]Note that this number is mainly used to tune the λ parameter. The final number of open gates is ultimately determined by the targeted sparsity rate which is treated as a hyperparamater in most of our experiments. Therefore, the vocabulary size of this tokenizer does not play a crucial role in the conclusions we make. However, as our

optimizer (Kingma and Welling, 2013) for training our models with a learning rate 10^{-4} and trained our models for 200 epochs. More details of the settings are available in the Appendix A.2.

For the forward probing classifier, we train a 2 layer MLP as our probing classifier f, which predicts the target token given the slot vector. For the reverse probing classifier, we first pass the target tokens into an Embedding layer and then apply a 2 layer MLP on top of each embedding individually to predict the parameters of a Gaussian distribution over slot vectors.

2.4.2 Slot Initialization Analysis

		BPE			Morfessor		
Initialization	Model	P	R	F1	P	R	F1
eq. (2.2)	untrained	0.71	0.09	0.17	0.71	0.11	0.19
(ours)	slot-attn	0.96	0.73	**0.82**	0.95	0.74	**0.83**
eq. (2.3)	untrained	0.79	0.11	0.20	0.78	0.14	0.23
	slot-attn	0.98	0.63	0.76	0.98	0.64	0.77
eq. (2.1)	untrained	0.80	0.10	0.18	0.78	0.12	0.20
	slot-attn	0.97	0.51	0.66	0.96	0.51	0.66

Table 2.1: Forward probing results of different slot initializations for the Spanish language.

We compare our proposed initialization with the original definition of Slot Attention, where slots are randomly initialized from a single shared distribution. Table 2.1 shows the probing results of our model under different slot initializations. Having a separate μ for each slot (Equation (2.2)) increases the capacity of the model and thus yields better results in comparison to sharing μ between the slots. The model especially achieves higher recall in this case, which implies better recovery of the BPE and Morfessor tokens as it can accurately model a greater number of units. However, we achieve similar results with fewer parameters when initializing slots with positional information (Eq 2.3). This positional information leads to large improvements compared to the model trained without it (Eq 2.1). But there is still a substantial performance gain from using our full model, with different μ per slot, compared to just using positional information.

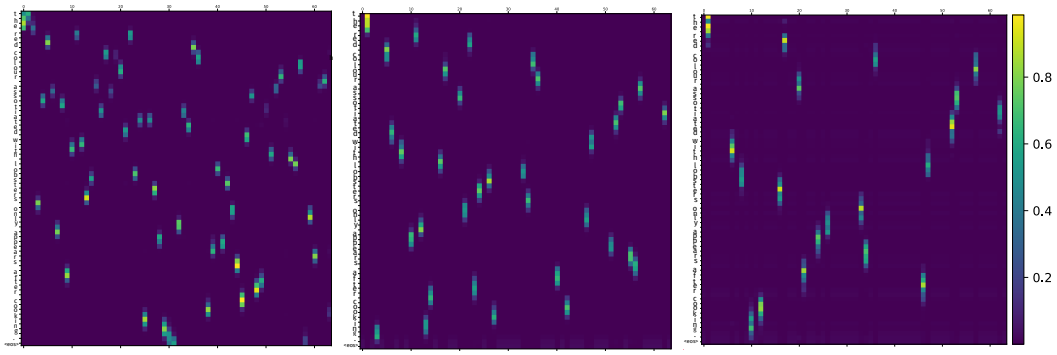
2.4.3 Reconstruction Quality

The models are trained to reconstruct the input character sequence, so the reconstruction error (reported in Table 2.2) indicates how effectively the model can compress information about the text in the available units' vectors. We compare the slot attention based model with targeted sparsity objective (Equation (2.11)), with the stride-based models (stride=6) to ensure that both models have the same average number of units. In all languages, slot attention based models are

experimental results indicate, the target number of units leads to slightly different behaviors.

	slot attention(r=6)		stride=6	
language	train	test	train	test
DE	**0.0022**	**0.0039**	0.0174	0.0213
EN	**0.0014**	**0.0033**	0.0120	0.0154
CS	**0.0021**	**0.0035**	0.0089	0.0137
FI	**0.0028**	**0.0045**	0.0097	0.0135
FR	**0.0043**	**0.0075**	0.0182	0.0229
ES	**0.0014**	**0.0036**	0.010	0.0149

Table 2.2: Reconstruction error on training and test set among different languages.



(a) epoch 5 (b) epoch 50 (c) epoch 125

Figure 2.2: Attention of the decoder over slots (x-axis) for generating every target character (y-axis) after different training epochs, for target sequence "*the red colour associated with lobsters only appears after cooking.*".

better able to reconstruct the sequence. This indicates that the slot attention vectors are better than the stride-based vectors at capturing information about the structure and meaning of the input, thereby providing better signals to the decoder to reconstruct the sequence.

2.4.4 Visualization

In order to show some qualitative results of our model, we visualize the attention maps for generating every output, shown in Figure 2.2. In particular, we show the attention of the decoder over slots when generating every output character [6]. Interestingly, although we do not impose any sparsity in the decoder's attention weights, the attention maps are quite sparse. Namely, at each generation step only a few slots are attended, and each slot is attended while generating only a few characters. In addition, although we do not impose any bias towards discovering segments of the input [7], the characters which are generated while attending to a given slot are contiguous in

[6] We observe the same pattern in the attention of slots over the input and have illustrated some examples in Appendix A.1.1

[7] This will give the model the flexibility to learn units for languages with non-concatenative morphology like Arabic.

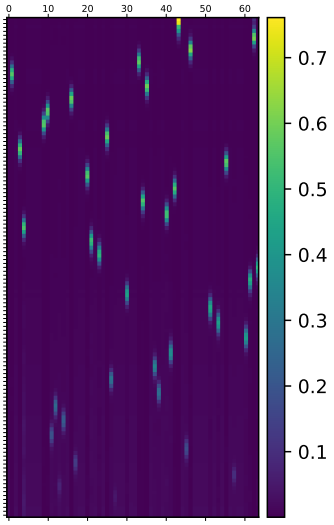


Figure 2.3: Average attention maps across all the test set samples. x-axis corresponds to slots and y-axis is the decoder's output position. The later positions (bottom) occur in fewer examples, but almost all slots show a clear preference for specific positions.

the string (the vertical bands in Figure 2.2). We believe that the emergence of contiguous spans is a result of the bottleneck we create with our Dynamic Capacity Slot Attention. This means that the model is trying to put correlated information about the input in the same vector, so that it can represent the string more efficiently. The strongest correlations in the character string are local in the input, so each slot tends to represent characters which are next to each other in the input.

In early steps of training, when the sparsity ratio (λ) is small, each slot tends to represent a bigram of characters (2.2a) and later on, trigrams (2.2b). These observations confirm the necessity of the L_0Drop layer for converging to better units. In particular, as the ratio increases, the number of active slots reduces and they become more specialized in representing contiguous meaning-bearing units of input. For instance, the word *cooking* in 2.2c is represented by two slots *cook* and *ing*. That these segments roughly correspond to the morphemes which we want the model to discover, is verified quantitatively in the probing experiments in Section 2.4.5. We provide attention map visualizations for other languages and stride-based models in Appendix A.1.1.

Learned Positional Information

We observed similar attention patterns for the slots across different input samples and thereby, we conjecture that there must be a correlation between what the slots have learned and the corresponding positions in the input sequence. To evaluate this, we average the attention maps between all the samples from the test set and visualize them in Figure 2.3. The averaged attention map verifies our hypothesis that the slots are highly correlated with the position of characters in the sequence. We think that this average is related to what the μ_i is learning, but that after the Slot Attention iterations the slots carry more information than just positions.

		BPE				Morfessor				Morphemes			
lang	Model	P	R	F1	Rev	P	R	F1	Rev	P	R	F1	Rev
EN	untrained	0.74	0.10	0.17	−47.12	0.71	0.11	0.19	−36.53	0.69	0.11	0.19	−50.29
	stride= 6	0.95	0.66	0.76	−105.1	0.94	0.66	**0.76**	−107.9	0.91	0.64	0.74	−116
	slot-attn($r = 6$)	0.96	0.68	**0.78**	**-150.1**	0.93	0.63	0.74	**-140**	0.91	0.70	**0.78**	**-123.7**
FI	untrained	0.74	0.09	0.17	−72.03	0.90	0.06	0.11	−39.23	-	-	-	-
	stride= 5	0.94	0.63	**0.74**	**-123.7**	0.89	0.43	0.57	**-112.6**	-	-	-	-
	slot-attn($r = 5$)	0.94	0.58	0.71	−104.7	0.89	0.50	**0.63**	−80.62	-	-	-	-
FR	untrained	0.75	0.08	0.14	−51.59	0.72	0.11	0.19	−28.05	0.74	0.11	0.19	−68
	stride= 5	0.94	0.69	0.79	−129	0.93	0.65	0.76	−120.3	0.91	0.68	0.77	**-204.3**
	slot-attn($r = 5$)	0.96	0.70	**0.80**	**-164.2**	0.94	0.68	**0.78**	**-160.8**	0.91	0.72	**0.80**	−190.2
ES	untrained	0.71	0.09	0.17	−37.85	0.71	0.11	0.19	−44.63	-	-	-	-
	stride= 5	0.94	0.67	0.77	**-103.5**	0.93	0.65	0.75	**-131.6**	-	-	-	-
	slot-attn($r = 5$)	0.94	0.69	**0.78**	−94.83	0.93	0.66	**0.76**	−125.7	-	-	-	-
DE	untrained	0.91	0.08	0.16	−46.91	0.76	0.08	0.15	−64.87	-	-	-	-
	stride= 5	0.95	0.66	0.77	−170.1	0.91	0.58	0.69	−156.1	-	-	-	-
	slot-attn($r = 5$)	0.97	0.68	**0.79**	**-190.8**	0.93	0.66	**0.76**	**-157.9**	-	-	-	-
CS	untrained	0.94	0.08	0.16	−22.98	0.86	0.05	0.10	−34.69	-	-	-	-
	stride= 5	0.94	0.63	**0.74**	−104.5	0.90	0.52	**0.65**	−87.16	-	-	-	-
	slot-attn($r = 5$)	0.96	0.62	**0.74**	**-164.5**	0.95	0.47	0.62	**-148.1**	-	-	-	-

Table 2.3: Forward and reverse probing results on different languages, with a target number of units. Note that human-annotated morphemes are only available for En and FR.

2.4.5 Probing Results

Given a target set of units and a trained model for finding units, we align the two sets of units, and use forward probing to see whether each induced unit contains as much information as its aligned target unit and use reverse probing to see whether each induced unit contains more information than its aligned target unit. Optimizing both measures requires having the same information as each target unit in its associated induced vector, and hence having the same level of abstraction.

The primary mechanism we have to control the amount of information in an induced representation is to control the number of units. More units means that there are more opportunities for the alignment to find a unit which correctly predicts each target unit, but makes it harder to predict all the units from the available target units. We take two approaches to controlling the number of induced units. First, we fix the number of units to a target number, which allows an efficient comparison of models. Second, we vary the number of units, which allows us to evaluate how well different models can match the informativeness trade-off of the target representation.

Targeted informativeness.

As a less computationally expensive initial evaluation, we use hyperparameters to set the number of induced units to approximately the same as the number of target BPE tokens. For the stride-

based models, we simply set the stride according to the average target number of tokens (i.e., stride= 6 or = 5). For the slot attention models, we set the r in Equation (2.11) to be equal to the stride value.

We use probing to compare slot attention and stride-based models to an uninformative baseline representation (untrained), thereby controlling for the power of the probing classifier to learn arbitrary mappings (Conneau et al., 2018; Oord et al., 2018a). This baseline is the set of slot vectors output by a randomly-initialized slot attention model without any training.

Table 2.3 shows the results of the forward and reverse probing tasks on different languages. As the results show, the trained slots achieve much higher performance in all the tasks in comparison to the random baselines. Our model achieves very high precision in predicting the *non-empty* labels. Its performance is weaker on the recall side, but here the improvement over the untrained model is even more pronounced. This seems to be due to the imbalance between *empty* and *non-empty* labels in the training set, where the *empty* labels comprise around 66% of the data for the probing classifier. For this reason, below we will use F1 as our overall performance measure for forward probing.

For most languages and targets, our slot attention model performs better than the comparable stride-based model. In the cases where the stride-based model has a higher F1, its worse score for reverse probing suggest that this may just be the result of an informativeness trade-off. For the majority of cases where the slot attention model has a higher F1, its reverse probing score is also better. The performance gain of our slot attention models over the stride-based models is particularly noticeable for the two languages where we have real morpheme annotations, which implies that our slot attention based method is more effective in discovering linguistically meaningful units. The agglutinative language (Finnish) is generally harder than the fusional languages, but slot attention does relatively well at capturing the linguistically-motivated Morfessor targets compared to the stride-based model.

We further show some examples from the predictions of the forward probing classifiers . Table 2.4 shows two examples of the probing classifiers' predictions given the learned slots. As explained above, the model is quite precise in predicting *non-empty* labels.

Informativeness trade-off.

The advantage of our two-directional probing evaluation is that we can directly compare the informativeness trade-off of different models by plotting their probing results in two-dimensional graphs. To vary this trade-off, we train stride-based models with different strides, stride=$\{2,3,4,5,6\}$, and slot attention based models with different r, where r = stride, for the same strides. This leads to the two models having the same range of values for their number of valid units.

Figure 2.4 plots this informativeness trade-off for the same languages and target representations

input	target	probing classifier prediction
una razón de su auge fue su aparente éxito en tratar enfermos por epidemias infecciosas .	BPE	una razón de su auge fue su aparente éxito en tratar enfermos por epi#as# inf#ocosasón
	Morfessor	una razón de su auge fue su aparente éxito en tratar enfermcio por epidemias#s .rs
außerdem wurde er zum besten spieler des turniers gewählt .	BPE	außerdem wurde er zum sten spieler des #ur#ierers gewähl# .
	Morfessor	außerdem wurde er zutur besten spieler des turniers gewählt #

Table 2.4: Input and output pairs from Spanish and German datasets. The predictions are sorted based on their matching target. The empty label is shown as '#' and wrong predictions are shown in red.

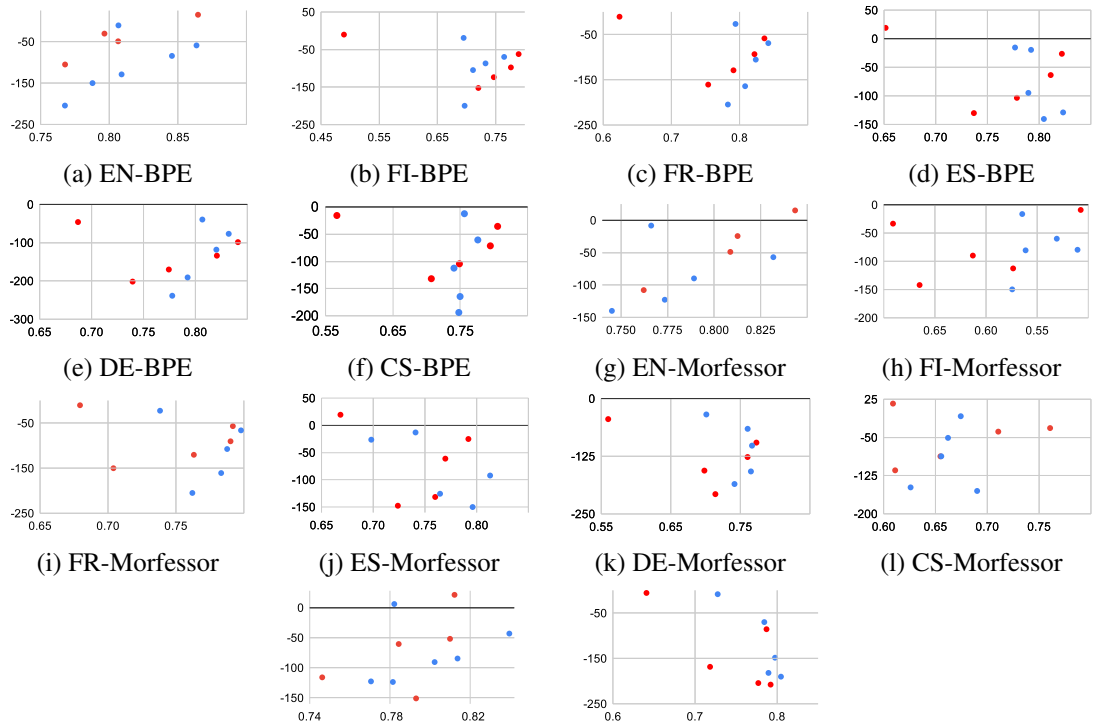
as in Table 2.3. Better results are nearer to the lower right corner of the plot. Overall, the slot attention based models provide better representations than the stride-based models. About half the plots show a clear trend in favour of slot attention based models, and none of the plots show a clear trend in favour of the stride-based models. But many of the plots show a substantial amount of variability.

These results justify our design choices for the slot attention based models to achieve the goal of discovering meaningful units which have the same level of abstraction as the previously proposed units. In addition, although our stride-based models fall behind the slot attention based models in many cases, they can be utilized as strong baselines for evaluating such unsupervised models and to find acceptable abstract units which can omit the need for tokenizing text as a preprocessing step. We provide the complete tables of results in Appendix A.1.2.

2.4.6 Downstream Task Evaluation

The focus of this work is developing the analogue of visual object discovery methods in the text domain, and developing an intrinsic evaluation framework for this line of work. We showed above that the information captured by our induced units are similar to the information in previously proposed representations, and thus there is every reason to believe that the demonstrable effectiveness of these predefined representations will also apply to our induced representations. To confirm the effectiveness of our induced meaningful units in downstream tasks, we report here results using our induced representations in a downstream task which we believe to be representative of their potential.

As our downstream task, we used the challenge set released by Hofmann et al. (2022), which they argue serves as a benchmark for pretrained language models' tokenizers. The task is to classify



(a) EN-BPE (b) FI-BPE (c) FR-BPE (d) ES-BPE

(e) DE-BPE (f) CS-BPE (g) EN-Morfessor (h) FI-Morfessor

(i) FR-Morfessor (j) ES-Morfessor (k) DE-Morfessor (l) CS-Morfessor

(m) EN-Morphemes (n) FR-Morphemes

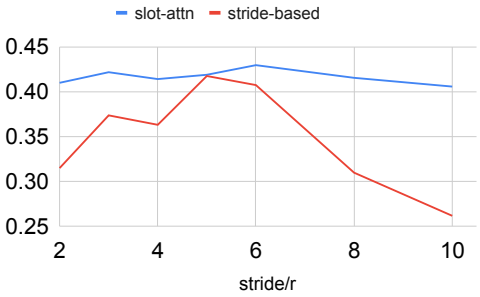
Figure 2.4: 2D graphs showing the informativeness trade-off for slot attention models (blue points) and the stride-based models (red points). The x-axis shows the F1 measure for the forward probe and the y-axis shows the negative log-density function loss for the reverse probe, so better models are lower and further to the right.

each ArXiv title into its corresponding sub-area (within a category of 20) in the three subjects of CS, Maths, Physics. The dataset requires a challenging generalization from a small number of short training examples with highly complex language (Hofmann et al., 2022). For this reason, we believe it is a good benchmark for evaluating our unit induction models.

In order to evaluate the quality of the induced units, we first freeze the representations we get from our trained models and then train classifiers for this downstream task. Since our models output a set of vectors as the high-level representation of the character sequence, we design an attention-based classifier to perform the task. In particular, after applying a shared 2 layer MLP (with ReLU non-linearity) on top of each of the representation's vectors, we apply attention over the resulting keys and values using a learnable query vector. Then, we linearly project the resulting vector to compute the score for each sub-area, and apply softmax with cross-entropy loss. Following Hofmann et al. (2022), we use F1 as our evaluation metric and average across the subjects. We compare our Slot Attention model with Stride-based model with stride 6 and a character-based model (stride=1) in addition to a BPE-based model, as shown in Table 2.5.[8] In

[8] Note that our numbers are not comparable to the ones reported in Hofmann et al. (2022) because we use a completely different setup (in terms of both pre-training data and architecture).

	Dev			
	Cs	Maths	Physics	Avg
stride=1	0.3053	0.3315	0.3867	0.3411
stride=6	0.3919	0.3976	0.468	0.4191
BPE-based	0.3196	0.3383	0.3693	0.3424
slot-attn(r=6)	0.3911	0.4229	0.4742	**0.4294**
	Test			
	Cs	Maths	Physics	Avg
stride=1	0.2934	0.3181	0.3839	0.3318
stride=6	0.3833	0.3863	0.4529	0.4075
BPE-based	0.3095	0.345	0.3578	0.3374
slot-attn(r=6)	0.397	0.424	0.4682	**0.4297**



(a) Results with varying stride/r on the test set.

Table 2.5: Arxiv-L classification results without fine-tuning the models.

order to be fair in comparison, we train the BPE-based baseline model under the same setup as a character-based model (stride=1) by only modifying the inputs and targets to be BPE tokens of the sentence. These results indicate that the representations induced by our proposed models work much better than a character-based and BPE-based model on this challenging task, confirming that they could be a useful replacement for tokenization methods in downstream tasks. The slot attention representations also perform better than the stride-based representations. The BPE-based model does not work as well as other models and we believe this is due to the fact that BPE struggles to find good tokenization of the rare technical words found in ArXiv topics and usually splits them into meaningless tokens.

Since there is no target representation in this evaluation, we also consider varying the target percentage of vectors in the induced representations, shown in Figure 2.5a. Interestingly, the best performance is achieved with induced units which are around the same granularity as those used in the BPE target representation (stride=6). For the stride-based models, results are very dependent on finding the right hand-coded level of granularity, whereas our slot attention model is more robust to the choice of the target level of granularity. Perhaps this is due to the ability of its L_0Drop layer to adjust the number of units depending on the content of each individual example.

2.5 Conclusions

In this chapter, we propose Dynamic Capacity Slot Attention for discovering meaningful units in text in an unsupervised fashion. We use an auto-encoder architecture for encoding a character sequence into a set of continuous slot vectors and decoding them to reconstruct the original sequence. We enforced the set of slots to act as a bottleneck in transferring information between the encoder and the decoder by adding a constant noise to their vectors and integrating an L_0

regularizing layer on top of the slots which only retains the necessary vectors. In addition, we propose a set of stride-based models which could serve as an alternative to our main model. We evaluate our model by probing the equivalence between the pruned slots and predefined tokens. In particular, we propose to do reverse probing as well as the normal way of probing. Our experiments show that our representations effectively capture meaningful information at a higher level of abstraction. Moreover, we show the usefulness of our induced units in a challenging classification task.

This work is a first step towards replacing hand-coded abstract units in text with representations induced within a deep learning architecture, demonstrating that it is feasible to improve over hand-coded units, even in the text domain. More generally, it addresses the issue of entity induction, which is a fundamental property of cognitive representations but is currently poorly understood. We believe that the proposed models and evaluation method will lead to greater progress on this fundamental problem.

Future work. One of the advantages of our unsupervised approach to inducing abstract units is that the method is not specific to characters, or indeed to any observable level of representation. Thus it can be trained in an end-to-end fashion with different training objectives, it can be used to induce any level of abstraction, and it can be stacked to induce multiple levels of abstraction. This could potentially avoid the need to hand-code different levels of abstraction, such as vectors associated with tokenization and sentence markers. Replacing hand-coded model design choices with choices which are induced within a deep learning architecture is a central objective of this work.

Limitations. We learn a set of abstract continuous vector representations of the input and we do not explicitly discover morphemes or morpheme segments. Although the visualized attention maps show interesting patterns of input segmentation (see Figure 2.2), the vertical bands are fading near the end-points. More specifically, it is not straight-forward to determine boundaries between the bands as the transition is done smoothly due the continuous nature of the attention function. For this reason, we could not obtain good segmentations by employing simple heuristics on the attention maps.

3 Discovering Meaningful Units with Nonparametric Variational Information Bottleneck

Learned representations at the level of characters, sub-words, words and sentences, have each contributed to advances in understanding different NLP tasks and linguistic phenomena. However, learning textual embeddings is costly as they are tokenization specific and require different models to be trained for each level of abstraction. We introduce a novel language representation model which can learn to compress to different levels of abstraction at different layers of the same model. We apply Nonparametric Variational Information Bottleneck (NVIB) to stacked Transformer self-attention layers in the encoder, which encourages an information-theoretic compression of the representations through the model. We find that the layers within the model correspond to increasing levels of abstraction and that their representations are more linguistically informed. Finally, we show that NVIB compression results in a model which is more robust to adversarial perturbations.

This chapter is based on the following publication,

- M. Behjati [*], F. Fehr [*], J. Henderson, *Learning to Abstract with Nonparametric Variational Information Bottleneck.*[1], In Findings of EMNLP, 2023

Both Melika Behjati and Fabio Fehr contributed equally to this work. The candidate (Melika) was responsible for the general development of the idea and the design and implementation of the evaluation experiments. Fabio was responsible for the design and implementation of the model and contributed expertise in NVIB. In this chapter, we focus mainly on the candidate's contributions. Other details are reported in the appendices.

[1] https://aclanthology.org/2023.findings-emnlp.106/

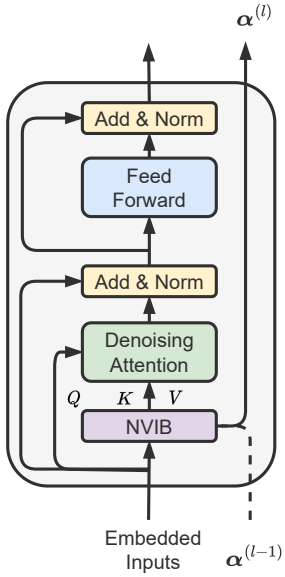


Figure 3.1: Transformer encoder layer (l) including the NVIB layer and Denoising self-attention module.

3.1 Introduction

Learning representations of language using self-supervision has become a cornerstone of NLP (Pennington et al., 2014; Peters et al., 2018; Devlin et al., 2019, *inter alia*). However, these representations are specific to their tokenisation (e.g. Byte-Pair (Sennrich et al., 2016), WordPiece (Schuster and Nakajima, 2012), SentencePiece (Kudo and Richardson, 2018b), characters (Al-Rfou et al., 2019), and even bytes (Xue et al., 2022)), which restricts the level of abstraction from the input text which their representations are able to convey. Work like CANINE (Clark et al., 2022) and Charformer (Tay et al., 2022) avoid problems with tokenisation by modeling individual characters or bytes, and thereafter use a stride-based downsampling to reduce the representation length. The stride pattern is fixed and thus can't be considered as learning to abstract. Behjati and Henderson (2023) recently introduced the task of learning a higher level of abstraction in a set-of-vector space by proposing Dynamic Capacity Slot Attention. In this work, we propose a novel character-level model of representation learning which learns different levels of abstraction in different layers of the same model.

We adapt the Nonparametric Variational Information Bottleneck regulariser (NVIB) (Henderson and Fehr, 2023) for application to self-attention in the stacked layers of a Transformer encoder.[2] The resulting model has greater abstraction than a standard Transformer due to selectively dropping some vectors in higher attention layers. Interestingly, we observe that the learned abstract units are intuitive, often corresponding to words. By employing different analysis

[2]The code is publically available at:
https://github.com/idiap/nvib
https://github.com/idiap/nvib_selfattention

methods, we demonstrate that our model is better at encoding semantically and linguistically meaningful information than a standard Transformer baseline. Moreover, it exhibits an enhanced level of robustness, further consolidating its advantage.

3.2 The Model

Our model consists of standard Transformer encoder-decoder layers (Vaswani et al., 2017b), where the encoder block has been augmented with an NVIB regulariser on the self-attention layers, as seen in Figure 3.1.

3.2.1 NVIB for Self-Attention

Nonparametric Variational Information Bottleneck is an information-theoretic regulariser for attention-based latent representations (Henderson and Fehr, 2023). It has been shown to induce smooth and sparse latent representations in the cross-attention layer of a Transformer encoder-decoder, where Henderson and Fehr (2023) used it to define a Variational Auto-Encoder (VAE) (Kingma and Welling, 2014). It generalises attention over a set of vectors to *denoising attention* over a mixture of impulse distributions, and uses Bayesian nonparametrics to handle the fact that the number of vectors grows with the length of the text. NVIB uses Dirichlet Processes (DPs) to define distributions over these mixture distributions, and controls the information in the latent representation by sampling a mixture distribution from the attention layer's DP, thereby adding noise which removes information.

We extend the previous work by adapting NVIB for use in the stacked self-attention layers of a Transformer encoder. By extending NVIB's information-theoretic regularisation to the series of latent representations inside the Transformer encoder, we see increasingly abstract interpretable representations in the higher layers.

NVIB layer

As with a standard attention layer, an NVIB layer maps a set of n vectors to an attention function. It first maps the n vectors $\boldsymbol{Z} \in \mathbb{R}^{n \times p}$ to the parameters of a DP, which are a total pseudo-count for its Dirichlet distribution and a mixture of Gaussians for its base distribution. Each of the n vectors is individually projected to a pseudo-count $\boldsymbol{\alpha} \in \mathbb{R}^n$ and a Gaussian component ($\boldsymbol{\mu} \in \mathbb{R}^{n \times p}, \boldsymbol{\sigma} \in \mathbb{R}^{n \times p}$) of the base distribution. The model can drop entire vectors by setting their pseudo-counts to zero, thereby making the representation sparse. In addition, there is an $n{+}1^{th}$ component of the base distribution for the prior, with parameters $\alpha^p{=}1$, $\boldsymbol{\mu}^p{=}\mathbf{0}$ and $\boldsymbol{\sigma}^p{=}\mathbf{1}$. The individual pseudo-counts are both summed to get the DP's total pseudo-count and normalised to weigh the components of the DP's base distribution. The NVIB layer then uses denoising attention to access either a set of weighted vectors sampled from the DP (at training time), or the base distribution of the DP (at testing time).

Henderson and Fehr (2023) use ReLU, linear and exponential activation functions to compute $\boldsymbol{\alpha}$, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, respectively. To adapt NVIB for stacked layers of self-attention, our model replaces the activation for the pseudo-count parameters with an exponential activation, and includes a multiplicative skip connection from the previous layer $l-1$, as shown in Figure 3.1:

$$\boldsymbol{\alpha}^{(l)} = \exp(\boldsymbol{w}\boldsymbol{Z}^T + b + \log(\boldsymbol{\alpha}^{(l-1)})), \tag{3.1}$$

where $\boldsymbol{w} \in \mathbb{R}^{1\times p}$ and $b \in \mathbb{R}$ form the linear projection. The exponential activation allows the model to be more stable in training.[3] The skip connection in between layers $l-1$ and l helps coordinate the importance of vectors across layers. Keeping the pseudo-count parameters in log-space prevents overflow and improves precision when the parameters get larger. This results in a multiplicative skip connection which emphasizes the communication between layers.

To compute self-attention, the DP parameters projected from all the individual vectors together define a single DP, and we take a single sample from this DP which all the individual vectors access via denoising attention. The queries for this denoising self-attention are computed from the original n vectors $\boldsymbol{Z} \in \mathbb{R}^{n\times p}$, before the NVIB layer. See Appendix B.4 for the exact attention functions.

Training objective

The NVIB loss regularises the attention-based representations so that the size of the representation at each layer is appropriate for the complexity of the representation being encoded at that layer. It has three terms, a reconstruction loss L_R, and two KL divergence terms: L_D for the pseudo-counts of the Dirichlet distributions, and L_G for the parameters of the Gaussian components. The L_R term is the supervised learning objective, which tries to make the latent representation informative enough to predict the original text. The L_G term tries to make the individual Gaussian components less informative, as in vector-space VAEs (Kingma and Welling, 2014). The L_D term tries to push down the total pseudo-count, which pushes some of the individual pseudo-counts to zero, thereby effectively dropping their vectors and reducing the number of vectors in the latent representation. See Appendix B.3 for loss equations.

To apply NVIB to stacked self-attention layers, we want to allow the lower layers to compute with more vectors while encouraging the upper layers to compress to fewer vectors, thereby encouraging abstraction at the higher layers. We therefore weight the loss terms differently at each layer:

$$\mathscr{L} = L_R + \beta^{(l)}(\lambda_D L_D + \lambda_G L_G) \tag{3.2}$$

$$\beta^{(l)} = \frac{l}{\sum_{l=0}^{N} l} \quad \text{for } l \in \{1, ..., N\} \tag{3.3}$$

[3] Since the exponential function is never exactly zero, we threshold small values to introduce sparsity. See Appendix B.1.

where $\beta^{(l)}$ controls the degree of NVIB regularisation for layer l, linearly increasing it for higher layers. If a vector is dropped in the last self-attention layer (i.e. zero pseudo-count), then we also drop that vector in the cross-attention layer to the decoder, but otherwise there is no NVIB regularisation of the cross-attention.

During preliminary experiments, instead of the above formula for $\beta^{(l)}$ we considered a uniform weight, as well as a doubling weight, per layer. These regularisation weights were either too weak or too strong, respectively. The values we considered for the hyperparameter λ_D are given in Appendix B.2. When we increase this regularisation, the characters are grouped into fewer and fewer vectors until all characters are compressed into a single vector, much like a sentence embedding. If we over-regularise, the representations collapse to the uninformative prior representation.

3.3 Related Work

Modeling language at the level of characters has the advantage of providing an end-to-end framework for the models to operate, without the need for tokenization as a preprocessing step (Xue et al., 2022; Ataman et al., 2020b; Choe et al., 2019; Al-Rfou et al., 2019; Kawakami et al., 2017). This is at the cost of longer sequence lengths and the need for greater model depth to reach the understanding level of subword-based models. While CANINE (Clark et al., 2022) and Charformer (Tay et al., 2022) are some attempts to bypass these shortcomings, they do so by fixed architectural design choices. Our work differs in that it allows the model to learn how to abstract and compress the input without a hard-coded abstraction structure. Our inspiration comes from Behjati and Henderson (2023) who introduced the task of learning a higher level of abstraction and proposed a method based on Slot Attention (Locatello et al., 2020) for this purpose. Our work is also related to HM-RNNs (Chung et al., 2017b) as it tends to learn a hierarchy of units within its layers, though it does not make discrete decisions on unit boundaries. Our approach to learning meaningful disentangled abstractions by encouraging the models to learn compressed representations through a bottleneck is shared with VAEs (Kingma and Welling, 2014) and other work in that line (Alemi et al., 2017; Higgins et al., 2017).

3.4 Experiments

Our proposed model's abstractness is analyzed qualitatively through attention visualisations (Section 3.4.2) and quantitatively through a challenging sub-topic classification task (Section 3.4.3). Each layer is probed to analyse the linguistic information captured (Section 3.4.3) and finally we examine the models' robustness to adversarial, synthetic noise (Section 3.4.4). We provide additional details of these experiments in the Appendices B.5 and B.6.

	P	R	F1
Transformer	**95.51**	56.51	64.52
NVIB	85.23	**79.02**	**78.86**

Table 3.1: Word segmentation performance [%].

3.4.1 Experimental Setup

Data

We train all models on the Wikitext-2 (Merity et al., 2017) encyclopedia dataset at the character level, with a noisy character deletion reconstruction objective (Lewis et al., 2020).

Models

We compare the self-attention representations from a standard Transformer encoder layer and our Transformer encoder layer with NVIB regularisation. We consider models consisting of six stacked Transformer encoder layers to be in line with the base model from Vaswani et al. (2017b). For the Transformer decoder we use only 2 layers so that the decoder is not able to compensate for poor embeddings from the encoder. For simplicity of implementation and interpretation, we use only a single attention head. For the NVIB models, we only apply NVIB to the final three layers. To ensure comparability between our model and the baseline, we train the baseline to have the same denoising capability and thus the same validation cross-entropy when evaluated on noised examples. For further details see Appendices B.1 and B.2.

3.4.2 Attention Map Visualisations and Analysis

To qualitatively evaluate the model's ability to learn interpretable abstractions, we visualise the self-attention maps. Figure 3.2 compares the self-attention patterns of the the last 3 layers of: a Transformer with 6 layers of standard attention (left); and a Transformer with 3 layers of standard attention followed by 3 layer of denoising attention with NVIB (right).

Despite being trained solely on noisy reconstruction at the character level, the NVIB layers compress the self-attention representations through the layers into distinct groups. At lower levels, the model uses nearly all vectors (i.e. ~99%) and learns position-local information, shown as a diagonal pattern. At higher levels the model drops some vectors (the blank columns) and groups characters (the vertical bars) in ways which strongly resemble subword units or even words. The last level retains only an average of ~35% of vectors. This is because the stronger NVIB regularisation at higher layers encourages the grouping of correlated characters, to reduce redundant information, and the strongest correlations are within words. We provide further examples in Appendix B.7.

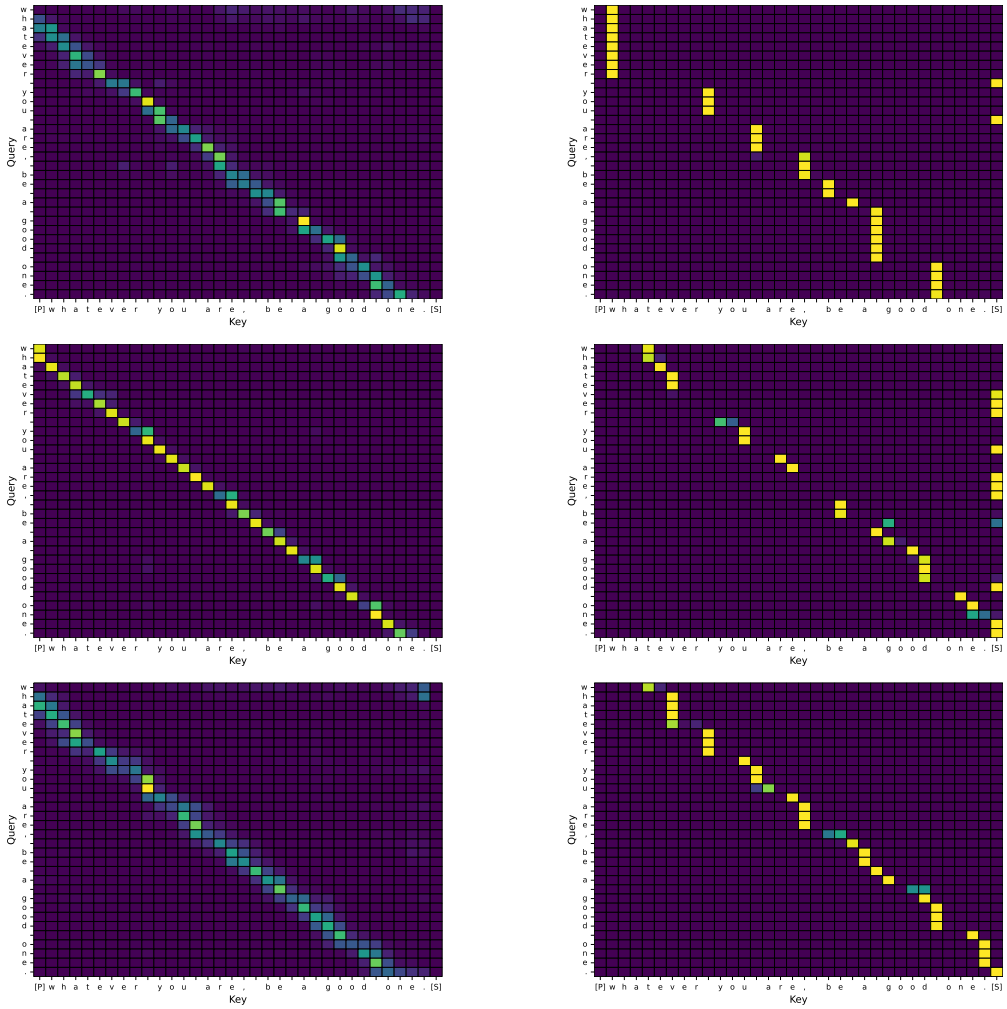


Figure 3.2: Self-attention patterns of the last 3 layers of 6-layer Transformer encoders from bottom to top. **Left**: Standard self-attention. **Right**: With NVIB regularisation. **Sentence**: "Whatever you are, be a good one." Dark purple is 0 and light yellow is 1 for attention.

Quantification of Word Resemblance

We quantify the resemblance of the final-layer self-attention maps to words by extracting contiguous segments from the maps and computing the F1 measure between our segments and the words in the sequence.

First, we take the $argmax$ over the Key dimension of an Attention map and extract the contiguous segments from the resulting vector. Then, we compute the intersection between the set of obtained segments and the set of words in the sequence. In particular, for each segment and word, we compute the number of intersecting characters (i.e., the length of the longest common substring) as a measure of their overlap. This would lead to a rectangular matrix of scores. Then, we perform the Hungarian matching algorithm (Kuhn, 1955) to find the best 1-1 match between the

two sets. Afterward, for each matched word and segment, we compute Precision (P), Recall (R) as follows,

$$P = \frac{\text{longest common substring length}}{\text{segment length}} \tag{3.4}$$

and

$$R = \frac{\text{longest common substring length}}{\text{word length}}. \tag{3.5}$$

We report the average macro $F1$, P, R over the validation set of our training data. Table 3.1 compares the performance of our model to the Transformer baseline. This impressive unsupervised performance (F1 of 78.86%) concurs with the attention visualisations and quantitatively verifies that our model has learned to abstract to the level of words. For the baseline Transformer, as it usually predicts units of length one or two which are within a single word the P is high as opposed to its R value.

3.4.3 Probing Analysis

This section uses different probing tasks to quantitatively evaluate the abstraction capabilities of our model and analyse the linguistic information captured by the layers. To evaluate the level of abstraction of our representations, we train probing classifiers to identify the sub-topic from short, technical input sentences (Hofmann et al., 2022). To analyse the linguistic features captured by the models layers, we train probing classifiers at each layer of the model and evaluate our representations on the suite of linguistic evaluations SentEval (Conneau and Kiela, 2018).

Probing Classifiers

We used two types of probing classifiers to perform our tasks. First, we employ an attention-based probing classifier to operate on top of the set of representations for predicting the specified property. This would be similar to having a learnable [CLS] token which is used as the representation of the sequence in BERT-like models. This is also in line with the findings of Pimentel et al. (2022) that the way we do the probing should resemble how the model itself would use the information within its architecture. In particular, we first map the representations into a new space with a 2 layer MLP. Then, we compute the attention with a learnable query vector. Finally, we linearly project the resulting vector into the number of classes for each task. We refer to this probe as *Attention-based probe*. Second, we tried a less complicated and more common type of probing in which we first aggregate the set of representation vectors by mean and then apply a 2 layer MLP with ReLU non-linearity to perform the task. We refer to this probe as *Aggregating probe*.

Task	Transformer	NVIB
Computer science	42.33	44.47
Mathematics	44.02	47.13
Physics	48.83	52.32
Average	45.06	**47.97**

Table 3.2: F1 score [%] on Arxiv-L classification task.

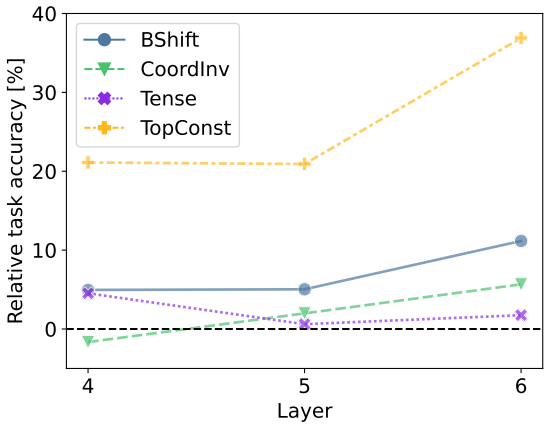


Figure 3.3: Relative performance of NVIB over Transformer for a subset of SentEval tasks.

ArXiv Topic Classification

The ArXiv topic classification task (Hofmann et al., 2022) is a challenging task consisting of short input sentences with long technical words. For each subject, the classifier should classify the topic into 20 possible sub-areas. Following Behjati and Henderson (2023), we train an attention-based probe on the final layer of the models and report the F1 measure for performance on the ArXiv-L dataset. Without finetuning the models, this classification task serves as probing high-level abstract linguistic properties (Hewitt et al., 2021). As shown in Table 3.2, the NVIB layer results in the model learning more information about the meaning and semantics in the abstract representations than characters and therefore provides better units for performing the task.

Linguistic Probing

The SentEval task set is specifically designed to examine the linguistic information available in a sentence representation at different levels, ranging from surface-level to semantic-level tasks (Conneau et al., 2018; Conneau and Kiela, 2018). We probe for linguistic information of our model and the baseline Transformer, across all layers. In general, the performance improves in deeper layers and increases further with the inclusion of NVIB in the layers. We highlight the results of four tasks in Figure 3.3, which to perform well in these tasks the representations must

	Layer	**CoordInv**	**ObjNum**	**TreeDepth**	**TopConst**	**BShift**	**Tense**	**SubjNum**
Chance		0.5	0.5	0.125	0.05	0.5	0.5	0.5
Transformer	1	0.5023	0.6498	0.2200	0.2880	0.5006	0.7306	0.6500
	2	0.5144	0.7255	0.2350	0.3724	0.4994	0.7891	0.7131
	3	0.5190	0.7547	0.2594	0.4261	0.5055	0.8263	0.7297
	4	0.5196	0.7687	0.2692	0.4368	0.5108	0.8114	0.7545
	5	0.5196	0.7737	0.2736	0.4369	0.5304	0.8320	0.7435
	6	0.5227	0.7756	0.2736	0.4212	0.5465	0.8384	0.7683
NVIB	1	0.5037	0.7646	0.2349	0.3323	0.5007	0.8344	0.7285
	2	0.5069	0.7859	0.2511	0.4243	0.5108	0.8379	0.7777
	3	0.5110	0.7963	0.2589	0.4453	0.5466	0.8606	0.7844
	4	0.5111	0.7879	0.2655	0.5290	0.5361	0.8481	0.7943
	5	0.5299	0.7660	0.2651	0.5283	0.5571	0.8371	0.7793
	6	**0.5523**	**0.8207**	**0.2923**	**0.5766**	**0.6075**	**0.8531**	**0.8038**

Table 3.3: Performance on Senteval tasks.

capture latent syntactic structures (**BShift**), cluster them by constituent types (**TopConst**), or have an understanding of semantics (**Tense**) or broad discourse and pragmatic factors (**CoordInv**) (Conneau et al., 2018). The inclusion of our NVIB layers increases the relative performance over the Transformer baseline, showing it to be more linguistically informed.

We report the complete set of results in Table 3.3 on a subset of 7 of the 10 SentEval tasks as sentence length (**SentLen**), word content (**WC**) and semantic odd man out (**SOMO**) tasks are too challenging for our models when encoding from a character level.

3.4.4 Robustness Analysis

We analyze the robustness of our models to synthetic noise injected into the input sequences (Belinkov and Bisk, 2017; Durrani et al., 2019). Namely, we evaluate the reconstruction quality when the inputs are perturbed by swapping, deleting, inserting, and substituting characters (Morris et al., 2020). We expect our model to be more robust due to its compressed representations. Figure 3.4 shows the performance of our model in comparison to a Transformer when the probability of injecting noise into the input increases. The results indicate that our model is more robust to adversarial noise than a standard Transformer, with an increased advantage as the level of noise increases. Building a model that is more robust to noise by design is a valuable contribution towards building reliable models.

3.5 Discussions

The methods that we propose in this chapter and chapter 2 are similar in that they both induce entities through compression, though they are different in two aspects. First, in this chapter,

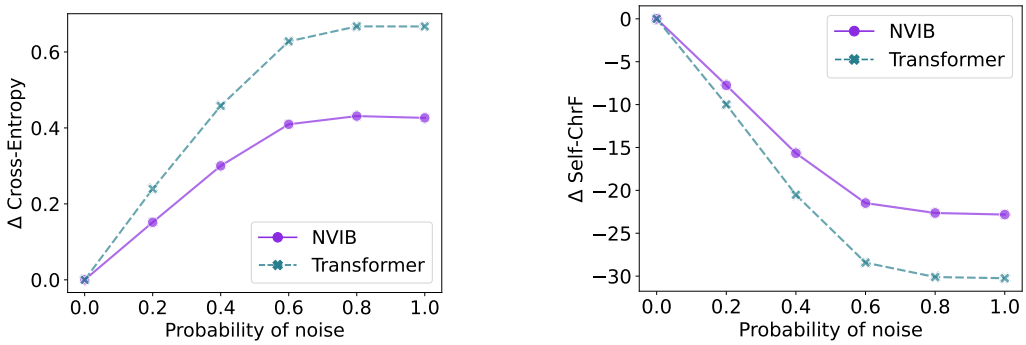


Figure 3.4: Robustness plots showing relative performance change over increasing input perturbations.

we induce entities at the level of words, while we induce entities at the level of morphemes in chapter 2. As a result, the previous literature and the experiments are different. Second, the entity induction models are trained differently in regard to the number of Transformer layers and the training objectives. Therefore, a direct comparison between the results is not possible (specifically for the ArXiv topic classification task which is shared between the two chapters). However, we believe that by tuning the hyperparameters of either of the models, we can induce entities at other levels.

3.6 Conclusions

In this chapter, we propose a novel method for inducing abstract representations of text. We adapt the Nonparametric Variational Information Bottleneck (Henderson and Fehr, 2023) regulariser for application to self-attention in the stacked layers of a Transformer encoder. Our model learns how many vectors are needed at each layer, thereby inducing different levels of abstraction in different layers of the same model. We find that these abstract units are intuitive, more robust, and better at encoding semantically and linguistically meaningful information.

Limitations

While the models and training data are reasonable in size, the experiments do not include the very large scale training often found in work on representation learning in text. We anticipate that the advantages of NVIB on self-attention layers will only increase as the models and data are scaled up, since this should allow even more abstract representations to be learned. In addition, the experiments are only done on English, but we would expect more improvements with more morphologically rich languages. In future work, we plan to explore fine-tuning NVIB for sparsity and downstream performance, and consider different tokenizations beyond characters only.

4 Discovering Meaningful Units with Visually Grounded Semantics

Fine-grained knowledge is crucial for vision-language models to obtain a better understanding of the real world. While there has been work trying to acquire this kind of knowledge in the space of vision and language, it has mostly focused on aligning the image patches with the tokens on the language side. However, image patches do not have any meaning to the human eye, and individual tokens do not necessarily carry groundable information in the image. It is groups of tokens which describe different aspects of the scene. In this work, we propose a model which groups the caption tokens as part of its architecture in order to capture a fine-grained representation of the language. We expect our representations to be at the level of objects present in the image, and therefore align our representations with the output of an image encoder trained to discover objects. We show that by learning to group the tokens, the vision-language model has a better fine-grained understanding of vision and language. In addition, the token groups that our model discovers are highly similar to groundable phrases in text, both qualitatively and quantitatively.

This chapter is based on the following article:

- M. Behjati, J. Henderson, *Discovering Meaningful Units with Visually Grounded Semantics from Image Captions*, under review, 2024

4.1 Introduction

Vision-language models have been shown to be less effective at capturing fine-grained information about the images described by the captions (Bugliarello et al., 2023; Kamath et al., 2023; Yuksekgonul et al., 2022). This information is crucial for the models to obtain a better understanding of the real world. While there has been work trying to acquire this kind of knowledge in the space of vision and language, it has mostly focused on aligning the image patches with the tokens on the language side (Yao et al., 2022; Wang et al., 2022; Zeng et al., 2022a; Mukhoti et al., 2023). However, image patches do not have any meaning to the human eye, and individual tokens often do not carry information groundable in the image. Minimally, it is groups of image patches which represent objects and the group of tokens in the text that refer to those objects. For this reason, there has been an active line of research in vision investigating the unsupervised discovery of objects by learning to assign image patches to their representative object slots (Locatello et al., 2020; Sajjadi et al., 2022; Wu et al., 2024). Recently, Xu et al. (2022) integrated an object discovery module into their vision-language model to learn the object entities. They showed that representing the image at the level of its constituent objects improves the performance of their model in downstream tasks. In this paper, we investigate the unsupervised discovery of groundable phrases on the language side to get better correspondence with objects on the vision side. We hypothesise that finding these meaningful units in language representations will improve the fine-grained understanding of image-caption semantic relationships. As far as we are aware, we are the first to investigate this possibility.

We base our model on the recent model of visual object discovery using image caption pairs proposed by Xu et al. (2022). We freeze the image side of the model, and introduce analogous deep learning mechanisms to discover *objects*[1] on the language side. We investigate two types of losses, one which promotes the correspondence between representations of the language side and representations on the vision side, and one which promotes the ability to reconstruct the text from the language representations. We find that training with both these losses leads to better fine-grained understanding of the image-text relationship, and discovers units which are highly similar to groundable phrases in text, both qualitatively and quantitatively. Further analysis finds that optimising the image-text correspondence alone does not lead to the discovery of meaningful units on the language side, and while this model does learn a good fine-grained understanding of the image-text relationship, it does not represent the semantics of objects as well as the model which does represent groundable phrases. We also find that optimising the reconstruction loss alone does lead to the discovery of meaningful units on the language side, but they have a slightly worse similarity to groundable phrases than the model which includes grounding information, and do not capture image-text relationships.

Our contributions are as follows,

- We develop a novel model to discover meaningful units from the image captions in the

[1]We use the terms *objects*, *entities*, *groups* and *units* interchangeably.

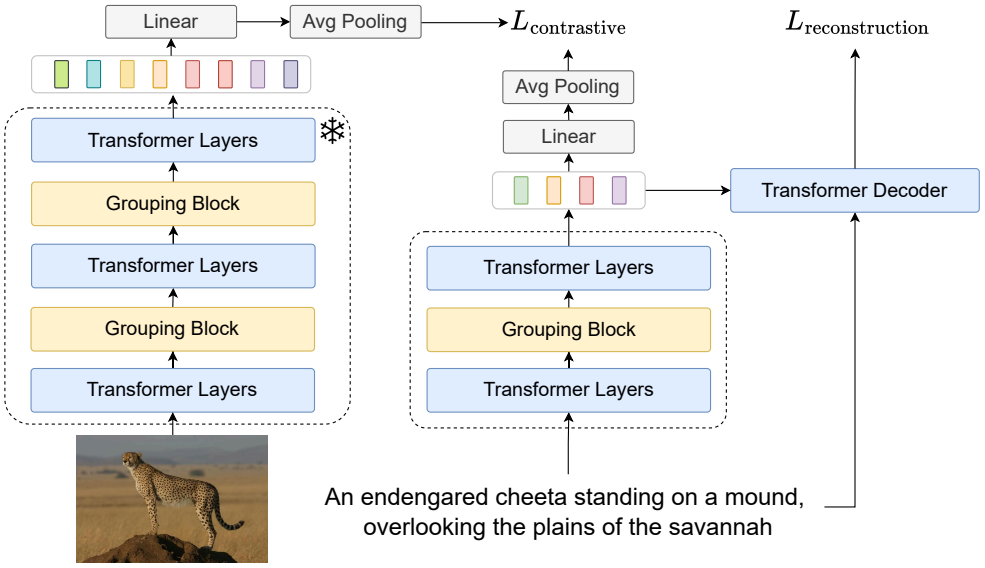


Figure 4.1: Overview of the model. We freeze the image encoder and only train the text encoder, decoder and the linear projection heads. The image passes through Transformer layers followed by the grouping blocks. The output of the image encoder is a set of groups which are approximately representing the objects. The caption also passes through the same set of blocks and the output of the text encoder is a set of groups representing units in language. The two modalities interact via a contrastive loss. There is also a reconstruction loss where the decoder decodes the text groups into the original input.

vision language setup (Section 4.2.1).

- We improve the fine-grained vision and language understanding of our model compared to a single-vector representation of text, under two different benchmarks (Section 4.4.2).
- We show that the segments that our model discovers are meaningful both qualitatively, and in terms of accordance with groundable phrases (Appendix A.1.1).

4.2 Method

To facilitate learning the fine-grained semantics of image-text relationships, we propose a model for learning text representations whose granularity matches the granularity of objects in the image, meaning that it is neither as course-grained as having a single vector for embedding the entire text[2] nor as fine-grained as having a different vector for every token. Given a dataset of image-caption pairs, $\mathrm{D} = \{(I_i, T_i)\}_{i=1,\ldots,N}$, we want to learn a representation of each caption in the form of groups of tokens which are aligned with the semantic space of objects in its image. To do so, we freeze the image encoder which has been trained to output the objects in the image and only train the text encoder and the projection heads. In particular, if the input representation of

[2]This is the common way of representing text in dual-stream vision-language models like CLIP (Radford et al., 2021).

language is at the level of subwords, we aim to find a higher level representation of them which would approximately represent groundable phrases. More specifically, let $T_i = [t_{i1}, \ldots, t_{iM}]$, where t_{ij} is a subword of T_i and M is the total number of subwords in T_i. We would like to group the subword tokens t_{ij}s into non-overlapping groups $T_i = \{g_{i1}^T, \ldots, g_{iK}^T\}$ where $K < M$. This would lead to a more compact abstract representation of T_i.

4.2.1 Model

We illustrate an overview of our model in Figure 4.1 and describe each of its components in the following sections.

Text Encoder: Text Group Transformer

We design our text encoder to learn semantic units of language. The key idea is to have shared learnable group vectors which can bind to different tokens of input (Xu et al., 2022). At each stage the groups carry the information from the previous layer to the next layer. To initiate the binding, the groups are appended to the input tokens they need to bind, and they all interact via several Transformer encoder layers to allow the groups and tokens to exchange information. Then, by performing a top-down attention mechanism shown as the Grouping block, the groups bind to different parts of the input.

More specifically, we first embed the input tokens and add learned positional encodings to them. Then, we append the learnable group vectors, $[g_{ik}^T]_{k=1\ldots K}$, to these embedded inputs, $[t_{ij}]_{j=1\ldots M}$, and pass the resulting vectors through some Transformer encoder layers, allowing them to interact with each other. We denote the encoded tokens and groups as $\hat{t}_{ij}$ and $\hat{g}_{ik}$. Then the grouping happens in a grouping block. In this block, the groups act as the queries and the encoded inputs as keys and values through a top-down attention mechanism. As with standard attention, the raw attention scores are computed as

$$A_{kj}^{\text{raw}} = \frac{Q(\hat{g}_{ik}^T)K^\intercal(\hat{t}_{ij})}{\sqrt{d}} \tag{4.1}$$

where d is the dimension of the model and Q and K are linear query and key projections. In order to have discrete assignments of inputs to the groups, GroupViT actually performs a hard assignment over A^{raw} by utilizing Gumble softmax (Jang et al., 2017; Maddison et al., 2017). Namely,

$$A' = \text{Gumble Softmax}(A^{\text{raw}}). \tag{4.2}$$

In top-down attention, instead of normalizing over the keys in the softmax function, the A' weights are first normalized over the queries, which are the groups. This will make the groups compete for representing different inputs (Locatello et al., 2020) and has been shown to be the most important component in discovering objects (Wu et al., 2023). After the normalization, the hard assignment happens and the gradient is backpropagated with the straight through trick (Van

Den Oord et al., 2017), that is:

$$A = \text{one-hot}(\text{argmax}_{\text{groups}}(A')) - \text{sg}(A') + A' \tag{4.3}$$

where sg is the stop gradient operator. Finally, the group vectors get updated as

$$\bar{g}_{ik}^{T} = \hat{g}_{ik}^{T} + W(\sum_{j} \frac{A_{kj}}{\sum_{j} A_{kj}} V(t_{ij})) \tag{4.4}$$

where V and W are the linear projections for values and outputs respectively.

After the grouping block, the updated group vectors serve as inputs to subsequent Transformer encoder layers. Finally, these refined groups represent the fine-grained semantics of the text in our model.

Image Encoder

We use the image encoder of Xu et al. (2022), which follows the same architecture as the text encoder, but with two stacked levels of transformer encoder layers and grouping blocks. As its input, the images are first divided into patches and then linearly projected. The encoder then extracts the set of image groups denoted as $\{\bar{g}_{ik}^{I}\}$. Due to the computational cost, we freeze the image encoder and assume that the image groups are representing objects in the image.

4.2.2 Training Objectives

Our model is trained with two different losses, i.e., a contrastive loss and a reconstruction loss, which we will explain in the following. The two losses are combined with a hyperparameter λ which controls the ratio between the two terms.

$$L_{\text{total}} = L_{\text{contrastive}} + \lambda L_{\text{recontruction}} \tag{4.5}$$

Contrastive Loss

The image and text modalities interact via a contrastive loss. First, the final groups for each modality are mapped into a common space with a Linear projector (Φ^{T}), i.e., $z_{ij}^{T} = \Phi^{T}(\bar{g}_{ij}^{T})$. Then, we average pool over them to obtain the global features for each modality ($\hat{z}_{i}^{T}$). We compute the InfoNCE loss (Oord et al., 2018b) for every modality separately. Given a batch size of B and a similarity function (sim), the infoNCE loss for the image to text is

$$L_{\text{I-T}} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{e^{\text{sim}(\hat{z}_{i}^{T}, \hat{z}_{i}^{I})/\tau}}{\sum_{j=1}^{B} e^{\text{sim}(\hat{z}_{j}^{T}, \hat{z}_{i}^{I})/\tau}}, \tag{4.6}$$

and respectively for the text to image is

$$L_{\text{T-I}} = -\frac{1}{B}\sum_{i=1}^{B}\log\frac{e^{\text{sim}(\hat{z}_i^T,\hat{z}_i^I)/\tau}}{\sum_{j=1}^{B}e^{\text{sim}(\hat{z}_i^T,\hat{z}_j^I)/\tau}}. \tag{4.7}$$

The final contrastive loss is calculated by averaging the two losses,

$$L_{\text{contrastive}} = \frac{1}{2}(L_{\text{I-T}} + L_{\text{T-I}}). \tag{4.8}$$

As for the similarity function sim(a,b), we consider the cosine similarity between the vectors.

Reconstruction Loss

In order to encourage the model to group the tokens into meaningful units, we incorporate a reconstruction loss from a text decoder. This loss encourages the model to assign tokens to different groups in order to spread information about the text across multiple vectors, and thus make better use of the available vectors.

We employ a simple shallow Transformer decoder to reconstruct the original input conditioned on the text groups. The shallow decoder has to rely on the information in the groups for decoding. Thus, it enforces the encoder to better encode the information into the groups (Bowman et al., 2015).

The output of this layer is

$$\overline{T}_i = \text{TransformerDecoder}(T_i|\{g_{i1}^T \dots g_{iK}^T\}). \tag{4.9}$$

The probabilities from these predictions are then used to define the reconstruction loss:

$$L_{\text{reconstruction}} = \sum_{i=1}^{B}\text{CE}(\overline{T}_i, T_i|\{g_{i1}^T,\dots,g_{iK}^T)\}) \tag{4.10}$$

where CE is the cross entropy between the output probabilities of the decoder and the original input given the discovered groups.

4.3 Related Work

Our work is related to different tasks in vision and language, which we will explain in this section.

Object discovery.

Here the task is to discover the objects in an image or video without any supervision. Slot-based object discovery (Locatello et al., 2020) has become popular due to the simplicity of the method

(Singh et al., 2022; Sajjadi et al., 2022; Singh et al., 2023b; Seitzer et al., 2023; Singh et al., 2023a; Wu et al., 2023, 2024). We have a novel adaptation of this method in discovering units similar to phrases in language with visually grounded semantics.

Weakly supervised visual grounding.

Visual grounding refers to the tasks where a phrase or expression is grounded in the image. In the weakly supervised setup, the only information used is the pairing of the image with its caption. In weakly supervised phrase grounding, the phrases are predetermined and no discovery happens on the language side (Datta et al., 2019; Gupta et al., 2020; Wang et al., 2020; Chen et al., 2022). In referring expression comprehension and referring image segmentation, the model must identify a specific part of the image described in a single expression. Kim et al. (2023) addressed the task of referring image segmentation by employing a slot-based object discovery module and merging relevant slots by cross attending over them with the textual query to build the final segmentation.

Vision language models with vision and language alignments.

While many large-scale vision language models have been developed, it has been shown that they fall short in understanding fine-grained details in the image. This is especially more pronounced in the dual-stream Vision Language Models (VLMs), where the modalities interact only via a single-vector representation. Therefore, there has been efforts to align language and vision at the level of patches and tokens (Yao et al., 2022; Wang et al., 2022; Mukhoti et al., 2023). Zeng et al. (2022b) use additional supervision from the phrase grounding annotations to help the model learn the alignments. (Bica et al., 2024) aligns tokens and patch embeddings at different levels of granularity simultaneously. (Li et al., 2022) learns the semantic alignment from the perspective of game-theoretic interactions.

Object detection.

The objective of this task is to detect the object boundaries in an image. Our work is related to query-based object detection, such as the approach in (Carion et al., 2020; Kamath et al., 2021), where, at decoding time, learnable object queries attend to the input features and encode an object. Liu et al. (2023) extend this approach by proposing a dual query model, demonstrating that simultaneously learning phrases and their corresponding objects improves the module's groundable understanding. The main difference between our model and this line of work lies in the weakly supervised nature of our approach.

Zero-shot open-vocabulary semantic segmentation.

Semantic segmentation is a well-established task in computer vision. Recently, with the rise of VLMs, these models have demonstrated promising zero-shot capabilities in the semantic

segmentation task as well. (Xu et al., 2022) propose a hierarchical grouping architecture that learns to group image regions without pixel-level annotations, relying solely on paired image and text data. Patel et al. (2023) expanded on image-text alignment, suggesting to not only align an image to the corresponding text but also to the text from visually similar samples. Additionally, Mukhoti et al. (2023) propose aligning patch tokens from a vision encoder with the <cls> token from a text encoder to enhance the model's performance.

Unit discovery in language.

Lately, discovering language units as part of the model architecture has been explored. These models operate on top of characters, where the units are usually at the level of subwords or words. The purpose is to optimize model efficency (Dai et al., 2020; Nawrot et al., 2022, 2023; Sun et al., 2023) or to skip the tokenization step of preprocessing and develop an end-to-end model (Clark et al., 2022; Tay et al., 2022; Cao, 2023; Behjati and Henderson, 2023; Behjati et al., 2023). Our research aligns with these developments by also focusing on language unit discovery. However, it differs in that these units are semantically grounded to vision.

4.4 Experiments

In this section, empirically evaluate our proposed model. We will first evaluate the quality of the discovered segments quantitatively by their accordance with the groundable phrases in Section 4.4.4, and probe the fine-grained vision-language understanding of our proposed text encoder under two benchmarks in Section 4.4.2. Then, we show the effectiveness of our model in finding meaningful units by visualizing the attention maps in Appendix A.1.1. We also analyse the contributions of different aspects of our model with a series of ablation studies in Section 4.4.5.

4.4.1 Experimental Setup

Datasets:

We trained our models on the training split of GCC3M dataset which consists of around 3 million image-caption pairs collected from the web (Sharma et al., 2018). The average caption length in this dataset is 10.5 tokens. We will explain the datasets we used for evaluation in their corresponding sections.

Parameters:

We first resize the images to 224×224 and then divide them into patches of size 16×16. The image encoder has 12 Transformer encoder layers with the hidden dimension of 384 and two grouping blocks at the 6th and 9th layers. The number of groups in the first block is 64 and 8 in

the second block. We load the weights from the GroupViT released checkpoint[3] (Xu et al., 2022) and keep it frozen during training.

For the text encoder, we have 6 Transformer encoder layers followed by a grouping block[4] and then another 3 Transformer encoder layers. Each self-attention layer has 4 heads. We experiment with $K = 1, 2, 4, 8, 16$ as the number of groups. We report the performance and results of the model trained with 4 groups as it has the best performance, and study the effect of having different numbers of groups in our ablations. The text decoder has only 1 Transformer decoder layer consisting of one self-attention and one cross attention layer, each with 1 attention head. We tie the weights between the token embeddings in the encoder and the decoder. Both the encoder and the decoder have a model dimension of 128. The linear projection heads map each modality's feature vector to 256 dimension. We fix the τ to 0.07 in our contrastive losses and λ equals to 1. We use Byte Pair Encodings (Sennrich et al., 2016) as our tokenizer with a vocabulary size of around 50k tokens and the maximum number of tokens is set to $M = 77$ following previous work (Radford et al., 2021; Xu et al., 2022). We train our models with a batch-size of 4096 for 25 epochs and use the GradeCache library (Luyu Gao, 2021) to obtain this batch size on a single RTX3090 GPU[5]. We trained our models with AdamW optimizer (Loshchilov and Hutter, 2019) with a learning rate of 0.0016 with linear warmup for 2 epochs and cosine annealing decay.

Baselines:

We compared our model against a text encoder with 9 Transformer layers, where the final text representation is taken from the <eos> token. This is the architecture used in GroupViT and other dual-stream vision-language models (Radford et al., 2021) and has approximately the same number of parameters as our proposed model. We train this model under the same training setup as our own model.

In addition, we report the results of the trained GroupViT model with its own text encoder and 2 layer projection heads. Note that this model has many more parameters and has been trained on 10x more data.

4.4.2 Fine-grained Vision-Language Understanding Probes

We evaluate the fine-grained vision and language understanding of our model by employing different benchmarks which are specifically designed for this purpose. We will explain each of these benchmarks and the zero-shot performance of our models in the following sections. In each case, the zero-shot classifier ranks the image-text pairs by their similarity scores $\text{sim}(\hat{z}_j^T, \hat{z}_i^I)$, which is the cosine between the pooled embeddings on the image and text sides. We refer to the

[3]We take the checkpoint trained on GCC3M (Sharma et al., 2018), GCC12M (Changpinyo et al., 2021) and YFCC14M (Thomee et al., 2016) datasets.

[4]Our prilimenary experiments with two blocks did not lead to reasonable results.

[5]It takes around GPU 48 hours for every model to train.

Model	subj	verb	object	overall
random	50	50	50	50
groupvit	81.6	77.3	91.7	81.0
transformer	80.5	69.5	89.0	75.3
ours (4 groups)	80.3	70.1	90.4	76.0

Table 4.1: The zero-shot pairwise ranking accuracy of different models under SVO probes.

score obtained from this zero-shot classifier as pair-wise ranking accuracy.

SVO Probes

Hendricks and Nematzadeh (2021) designed a benchmark where they pair every sentence with two images, one positive and one negative. The negative images are selected in a controlled fashion where only either subject, verb or the object of the image is different from the original one. The test split of this dataset contains around 30k examples.

Table 4.1 shows the results of the zero-shot performance of different models under this benchmark. We observe that our model has a better overall performance compared to the Transformer baseline, which verifies our hypothesis that representing the language in a fine-grained and meaningful manner helps the fine-grained vision and language understanding of the model. The Transformer's single-vector representation succeeds in capturing information about subjects, but our multi-vector representation does a much better job of representing objects, and to a lesser extent verbs. Both of these models are well above the random baseline. The results for GroupViT's Transformer model are not comparable because it is trained on much more data, but we see that the resulting increase is much higher on verbs than on the the groundable phrases (subjects and objects) that our model is designed to represent as separate vectors.

FOIL-COCO

Shekhar et al. (2017) propose FOIL-COCO dataset where for every image there is a correct caption and a "foil" one. The foil caption is different from the original caption by altering one of the nouns in the original caption into a foil one. We evaluate the zero-shot performance of our model with pairwise ranking accuracy in Table 4.2 on the test split of this benchmark which has around 99k examples. We observe that our model demonstrates a remarkably good performance, outperforming the transformer model. This indicates that the noun understanding of our model has improved by learning fine-grained representations. Additionally, despite being trained on substantially less data than the GroupViT text encoder, our model performs nearly as well.

Model	accuracy
random	50
groupvit	82.5
transformer	80.91
ours (4 groups)	81.68

Table 4.2: The zero-shot performance of different models under the FOIL-COCO benchmark.

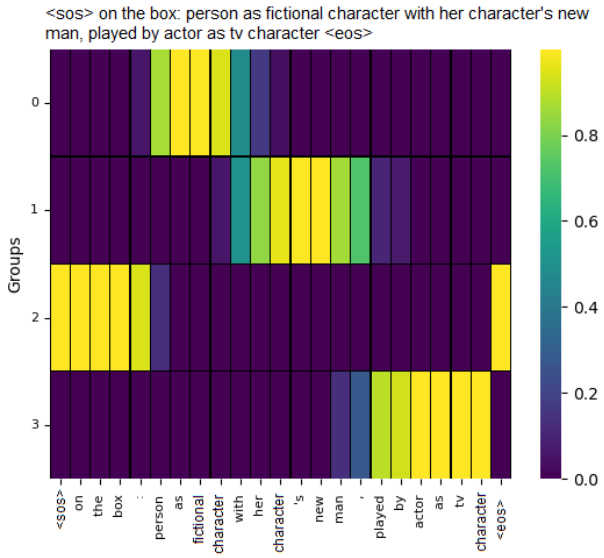


Figure 4.2: Soft attention of the groups over the input tokens. It shows that contiguous segments have emerged which capture phrase-like units.

4.4.3 Attention Visualization

In order to understand what each group is representing, we visualize the soft attention weights of the groups over the input subwords in Figure 4.2. Interestingly, we can observe that contiguous segments have emerged, without imposing any contiguity constraints in the groupings. We believe that this is due to the fact that usually in language the contiguous tokens capture highly correlated information and that's why our model is grouping them together as part of its compression. Moreover, we can see that the emerging segments are meaningful in that they capture phrase-like units. We quantitatively evaluate the phrase discovery performance of our model in the following section (Section 4.4.4). In our examination of a sample of attention maps, we observe that a given group tends to bind to similar positions in the text, but that the boundaries between groups vary.

Model	tIoU	P	R	F1
random	42.15	61.51	60.03	54.54
k-means	52.77	61.82	64.87	59.55
spectral-clustering	38.88	49.81	52.82	45.52
mean shift	50.38	**99.64**	51.73	65.13
ours (4 groups)	**76.42**	87.25	**85.83**	**83.72**

Table 4.3: Phrase segmentation performance of different models under different evaluation metrics.

4.4.4 Zero-shot Segmentation Evaluation

In order to evaluate the emerging segments in the attention maps quantitatively, we propose a metric similar to Intersection-over-Union (IoU) in the visual object detection literature which we call "tIoU". We first compute the soft attention weights of the groups over the input tokens. Then, by taking the argmax over the inputs, we have an assignment matrix of every input to a group. Given a gold segmentation, we can compute the IoU for each discovered group of tokens and each gold segment. For the computation of IoU, the intersection is equal to the number of overlapping tokens. For the union, we do not count the tokens which were not annotated in the dataset, as the annotators did not have the constraint to include all the tokens in their annotation. This gives us a matrix where by applying the Hungarian matching algorithm (Kuhn, 1955) maximizing this metric, we can obtain a 1-1 mapping between the discovered groupings and the gold segments. By having the mappings, we can compute precision, recall and F1 as well as IoU for each paired group and gold segment. In reporting the results, we first average every metric for the text input and then report the average over all examples.

For the gold segmentation, we use the annotations in Flickr30k Entities (Plummer et al., 2015) where groundable phrases are human-annotated. We report the results on the validation set of this dataset which has around 5000 examples. The number of annotated phrases in this dataset is on average 3.5.

In Table 4.3, we report the results of our evaluation. We compare our model against multiple baselines, including an untrained, randomly initialized model. We also report the performance of applying different clustering methods over the encoded features of our transformer baseline. In particular, we apply k-means, spectral clustering (Shi and Malik, 2000) and mean shift (Comaniciu and Meer, 2002) with 4 clusters. We observe that our model surpasses all the baselines by a large margin in all the metrics. Specifically, the high tIoU indicates that our model is indeed very good at discovering groundable phrases in the captions.

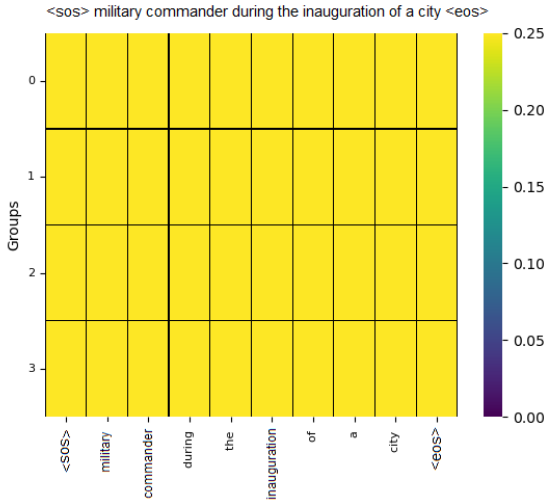


Figure 4.3: Soft attention of the groups over the input tokens for a model trained without the reconstruction loss. It shows a uniform attention map and lack of segmentation.

4.4.5 Ablation Study

In this section, we study the effect of different design choices on the performance of our models both in terms of groundable phrase discovery and fine-grained vision and language understanding.

		SVO					
model	tIoU	subject	verb	object	overall	FOIL	Noun Undr.
ours	**76.42**	**80.3**	70.1	**90.4**	76.0	**81.68**	**84.12**
w/o contrastive loss	76.18	51.4	49.7	50.8	50.2	42.59	48.26
w/o reconstruction loss	40.80	78.2	**72.6**	89.1	**76.9**	78.66	81.98

Table 4.4: The performance of our model compared to the ablated ones on multiple datasets. Noun understanding refers to the average of performance on noun phrases (i.e. subjects, objects and FOIL-COCO).

Training Losses

In Table 4.4 we see the different effects of the two types of loss on our multi-vector model. Without the contrastive loss, the model has no training on the image-text relationship, so it is not surprising that the image-text semantic evaluations are very low. More surprisingly, although it still segments in a meaningful way, without contrastive loss, the segmentation corresponds slightly less well to groundable phrases. This suggests that semantic grounding in images actually helps the model discover meaningful units of text.

Interestingly, without the reconstruction loss, the model fails to segment in a meaningful way. We can see this both in the tIoU score and in the uniform attention pattern shown in Figure 4.3. This

lack of segmentation in turn affects the fine-grained understanding of the image-text relationship. The holistic representations indicated by Figure 4.3 are relatively good at representing verbs, because verb understanding combines information across multiple objects. But if we only consider the noun phrases (i.e. subjects, objects categories from SVO probes and FOIL-COCO), averaged in the last column, then segmenting the representation according to semantic objects, as indicated in Figure 4.2, results in much better understanding of the image-text relationship.

# of groups	tIoU	SVO	Foil
1	43.55	74.99	80.23
2	53.12	75.30	80.01
4	**76.42**	**76.0**	**81.68**
8	63.93	74.8	80.56
16	52.54	72.4	79.43

Table 4.5: The performance of our model trained with different number of groups.

Number of Groups

In Table 4.5, we report the performance of our model trained with different numbers of groups. We can see that the model trained with 4 groups achieves the best results in all our evaluations. This implies that having too many or too few groups hurts the performance of our model.

4.5 Conclusions

In this work, we developed a novel model for discovering meaningful units that are semantically aligned to the objects in the image. We freezed an image encoder which outputs groups that approximately represent objects and employ an analogous architecture on the text side to discover units that are at the level of phrases. While many dual-stream VLMs represent text as a single vector, we hypothesise that learning to represent language at a finer granularity will improve their fine-grained vision and language understanding.

We verified our hypothesis by employing two specifically designed probing benchmarks, namely, SVO probes and FOIL COCO. In addition, we showed that the segments that appear in the attention maps of groups attending to tokens are meaningful both qualitatively and quantitatively, in terms of overlapping with groundable phrases. Moreover, we ablated the effect of our losses on learning these units and concluded that both are necessary for having meaningful and semantically aligned units.

Limitations

We have performed our experiments on the datasets and benchmarks in English. However, we do not make any language dependent assumptions in developing our model. Therefore, we believe that our method is generalizable across other languages as long as enough data for training is available.

We were not able to perform our experiments at scale due to the computational limitations. We expect that training the image and text encoder simultaneously from scratch would lead to better alignment between the two modalities, which should in turn improve our results.

5 Exploring a Task-driven Approach to Discovering Meaningful Units

In this chapter, we explore a task-driven approach to inducing meaningful units without supervision. We consider language modeling as a generic task that we want to optimize our units for. Here we take a more direct way of modeling the unit discovery task with a model which decides to either segment or not, at certain positions in a sequence of characters. We train our model by utilizing reinforcement learning.

Before trying our main idea, we perform a preliminary experiment in which we train a language model at different input representation levels. The results verify our assumption that representing language at the level of subwords leads to better performance and is more time-efficient. We then perform our main experiments. While our experiments with supervised signals show that the model is capable of outputting meaningful segmentation boundaries, we find that training both the segmentation model and the language model does not work.

The work that we present in this chapter was our first attempt towards entity induction in this thesis, where we wanted to directly find segments which are useful for a language model. As we did not find the proposed approach effective, we took an alternative approach to model entities with continuous vectors while compressing the original input sequence in an end-to-end manner.

5.1 Introduction

In linguistics, *morphemes* are considered to be the smallest meaningful units of a language. The combination of morphemes creates words e.g. the word *friendly* consists of the morphemes *friend* and *ly*. Discovering morphemes becomes an important issue in morphologically rich languages such as Turkish and Finnish, where a huge amount of rare words may appear by combining different morphemes. These languages suffer from high 'Out Of Vocabulary' (OOV) rate while being modeled at word-level; however, this problem exists in other languages as well with lower frequency. Therefore, by considering morphemes as the smallest units, we can mitigate this problem and take advantage of most of the data we have. Moreover, modeling languages at character-level cannot capture long-term dependencies, and some word-level information such as word boundaries is also disregarded.

In this chapter, we aim to discover morphemes from a sequence of characters in an unsupervised manner since labeled data is not always available, especially for low-resourced languages. We assume that by representing a language at the morpheme-level, we can improve performance on most relevant downstream tasks. Therefore, since our task is unsupervised, we consider improvement in a language model (generic downstream task) as our objective. In general, our approach consists of a main model that extracts the morphemes of a sequence and a language model that scores the quality of the generated morphemes. The scores of the language model are transferred to the main model by utilizing Reinforcement Learning approaches.

We performed several experiments. First, we simplified our problem to word-level and sentence-level prediction of morpheme boundaries, using an off-the-shelf tool's outputs as gold standards. After achieving reasonable results with supervised learning; indicating the capability of our model in learning the boundaries; we moved to our main problem setting. We tried different reward functions and RL techniques in our experiments.

5.2 Method

In this section, we describe our proposed method for discovering morphemes from a sequence of characters. Consider a character sequence $X = x_0 x_1 \dots x_n$ as input and the morpheme sequence $M = m_0 m_1 \dots m_k$ as the output. For instance, the output of the sequence 'John walked' would be 'John', 'walk','-ed' which '-' denotes the middle of a word.

We hypothesize that discovering morphemes, as the smallest meaningful units of language, can improve the performance of most relevant possible downstream tasks such as language modeling, machine translation, parsing, etc. As we are interested in performing the task in an unsupervised manner, we could consider improvement in a downstream task as our objective. A reasonable choice could be language modeling objective, which is considered a general task in NLP (Devlin et al., 2018).

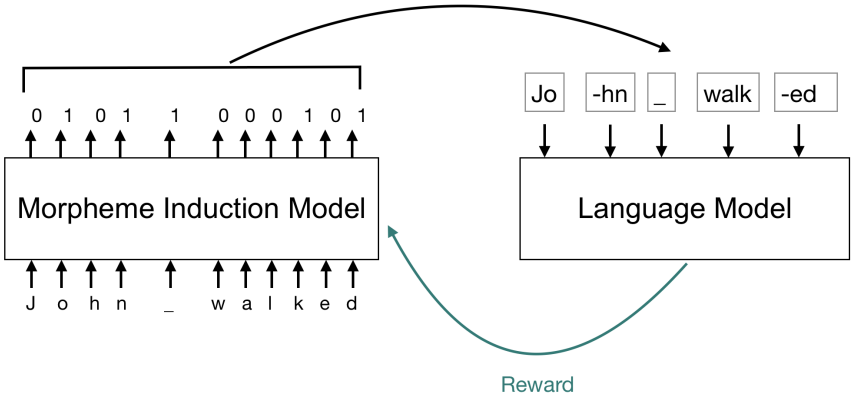


Figure 5.1: A sketch of our proposed method in a setting where the morpheme induction model outputs either 0(no segment) or 1(segment with no change).

In particular, our proposed method consists of two models:

1. Morpheme induction model: Our main model to perform the task of inducing morphemes from a sequence of characters.
2. Language model: Our downstream task model which could be improved by better performance of our main model.

Therefore, we aim to train the morpheme induction model to maximize the score of the language model. In order to achieve this goal, the language model should also be trained according to the outputs of the morpheme induction model. Figure 5.1 shows a sketch of our proposed method.

Let $X = x_0 x_1 \dots x_n$ be the input character sequence of the model and $S = s_0 s_1 \dots s_n$ the segmentation-rule probabilities for every adjacent pair. The segmentation rules R consists of the following rules (similar to Narasimhan et al. (2015)):

- Do not segment
- Segment
  - no change
  - repeat e.g. created = creat**e**+ed
  - delete e.g. planning = pla**n**+ing
  - modify e.g. studied = stud**y**+ed

We also assume that the word boundaries are known. Thus, the model outputs $p(S|X)$, and since the act of segmenting each position is independent of the previous positions, the probability could

be decomposed to $p(S|X) = \Pi_{i=1}^{n} p(s_i|X)$. Lastly, the morpheme sequence $M = m_0 m_1 \dots m_k$ could be obtained by applying a **sample** of S to the input X. Sampling is required to deterministically define the morpheme boundaries so that we can pass on the morphemes to the language model.

As described previously, we use a language model to score ($L(S, X)$) the quality of the generated morphemes. In which we consider the input as $M = m_0 m_1 \dots m_k$ and the output as the log-likelihood of the sequence normalized by the length of the character input (Equation 5.1).

$$L(S, X) = \frac{1}{|X|} \log p(M = m_0 m_1 \dots m_k) = \frac{1}{|X|} \sum_{i=0}^{k} \log p(m_i | m_0 \dots m_{i-1}) \tag{5.1}$$

In order to synchronize the morpheme induction model and the language model, we pretrain the language model on the outputs of the morpheme induction model initialized randomly. Then, both the language model and the morpheme induction model are updated simultaneously. In this setting, since both S and $L(S, X)$ are smooth, we can assume that small changes in S can cause small reasonable changes in L.

Since the process of inducing morphemes contains sampling from the segmentation-rule probabilities, it is not possible to pass the gradient of the language model directly to the morpheme induction model. Therefore, we use Reinforcement Learning to find the best parameters for our main model. In this setting, we consider the state to be the input character sequence and the action to be segmenting the sequence (which is done independently at the same time). Therefore, the policy $\pi_\theta = p(S|X)$ where θ defines the parameters of the morpheme induction model. The reward for this state and action would be language modeling's score. We use the policy gradient method REINFORCE (Sutton et al., 2000) to update the parameters of our main model. In our task, If we consider $U(\theta) = \mathbb{E}_{S,X \sim p(S,X,\theta)}[L(S, X)]$ as the utility of policy π_θ, then the $\nabla_\theta U_\theta$ could be calculated as in Equation 5.2 where m denotes number of samples.

$$\begin{aligned} \nabla_\theta U(\theta) &= \frac{1}{m} \sum_{i=1}^{m} L(S_i, X_i)\, \nabla_\theta \log p(S_i | X_i, \theta) \\ &= \frac{1}{m} \sum_{i=1}^{m} (\sum_{j=1}^{n} \nabla_\theta \log p(S_{ij} | X_i, \theta))\, L(S_i, X_i) \end{aligned} \tag{5.2}$$

In order to reduce the variance of the $\nabla_\theta U(\theta)$ we used a baseline b for the score function and thus:

$$\nabla_\theta U(\theta) = \frac{1}{m} \sum_{i=1}^{m} (L(S_i, X_i) - b)\, \nabla_\theta \log p(S_i | X_i, \theta) \tag{5.3}$$

The baseline is a simple Multi Layer Perceptron (MLP) which inputs X and outputs the value of it.

Finally, θ gets updated by gradient ascent algorithm with the learning rate α.

$$\theta^{t+1} = \theta^t + \alpha \nabla_\theta U(\theta^t) \tag{5.4}$$

5.2.1 Reward

In general, when training a sequential model using RL, one score is calculated as the reward for the whole output sequence (y) (Ranzato et al., 2016; Chen and Jin, 2020; Bahdanau et al., 2017) as in Equation 5.5. We refer to this strategy of reward as *sequence-level reward*.

$$\mathscr{R}([s_1, s_2, \ldots, s_n]) = k \tag{5.5}$$

where $k \in \mathrm{R}$. One shortcoming of this strategy is that it is often hard for the model to understand how the output decisions contribute to the final reward. For instance, consider y_i as a neutral action and y_j is a critical one with respect to the final reward, while both of them are treated similarly.

A way for mitigating this problem, especially when the sequence is too long, is to use a Monte-Carlo-Tree-Search algorithm for assigning a reward for each of the actions individually (Yu et al., 2017). We refer to this strategy as *per-action reward*.

5.3 Related Work

In this section, we discuss the previous work on morphology induction with an emphasis on unsupervised methods. In addition, we review some common subword discovery algorithms used in recent works on Natural Language Processing (NLP).

Some early works in inducing morphemes are based on a probabilistic view of the problem. In Creutz and Lagus (2002); Creutz (2003), the authors introduced a method to segment words into morpheme-like units. They also released their models as a tool called *Morfessor* (Creutz and Lagus, 2005), which is considered a baseline for most recent work. The core idea is to maximize the posterior probability of the discovered morpheme-like segments given the corpus during an iterative search algorithm. Another view of the problem was proposed in Narasimhan et al. (2015) and followed by Bergmanis and Goldwater (2017). They considered a log-linear model, based on semantic and syntactic features, which predicts the parent of a given word and generates a morphological chain of child-parent transitions. They also considered possible changes to the words during these transitions, such as repetition, deletion, and modification of the last character of the parent word. This allows them to move closer to discovering the exact morphemes of a word in comparison to *Morfessor* (Creutz and Lagus, 2005), which seeks the best segmentation.

Another approach to the problem is to extract morphemes as the common affixes or suffixes between pairs of words. In Soricut and Och (2015b), they considered morphemes as vector

transformations between pairs of words in the embedding space (e.g. *walk* to *walked* is the same as call to *called*). In Xu et al. (2018b), the authors proposed an algorithm to identify statistically reliable *paradigms*, set of morphological categories that can be applied to the set of words from the morphological segmentation result of a probabilistic model.

A different way of solving the problem was proposed in Cao and Rei (2016). They learned word embeddings by attending the hidden states of a bi-directional RNN on character-level input. Then, according to the attention weights, they were able to split the word into morpheme-segments and identify the segments which have more impact on the meaning of the word.

In order to broaden our studies in the field, we also reviewed some of the recent work on *supervised* morphology learning where labeled data is available. In Kann et al. (2016), they performed canonical segmentation using a character-level neural encoder-decoder structure (e.g. *questionably* → *question+able+ly*). They also included a neural reranker which rescores the canonical segments using morpheme-level and lexical information. In Cotterell and Schütze (2018), the authors proposed a probabilistic model of word formation which includes both segmenting the word into morphemes and also, the synthesis of the meaning of that word from the meanings of those segments.

Apart from the methods proposed for discovering morphemes as the smallest meaningful units of a language, some algorithms were introduced in the literature (Sennrich et al., 2015; Wu et al., 2016) for segmenting words into appropriate subword units. These subwords are not intended to carry on meaning as morphemes do. A common subword discovery algorithm called Byte-Pair-Encoding (BPE) was proposed in Sennrich et al. (2015). The algorithm starts with a set of characters and merges frequent pairs until reaching a specified limit. Another widely used algorithm is Word-Piece (Wu et al., 2016). It was designed to deal with infinite vocabulary in Japanese and Korean voice search systems. It is similar to BPE with the difference that pairs are chosen based on increasing the likelihood of a language model.

5.4 Preliminary Experiment

In this section, we describe the experiments we perform to verify our idea. Our proposed method is based on the assumption that morpheme-level representation of text can lead to better downstream task performance. Therefore, we design an experiment to validate this assumption.

Consider language modeling as our downstream task. We compare the effect of different levels of representation on the output of this model. In order for the loss to be comparable in all cases, we consider the negative of log-likelihood of the input sequence as the loss function (the negative of Equation 5.1).

5.4.1 Experimental Setup

Dataset

We used Penn Tree Bank dataset prepared by (Mikolov et al., 2010), which is both available in word-level and char-level. The text is lowercased and preprocessed to have a vocabulary size of 10000, and out of vocabulary words are replaced with '<UNK >' token for both levels. The word-level dataset contains 888k words for training, 70k for validation, and 79k for testing. The char-level dataset contains 5.01M characters for training, 393k for validation, and 442k for testing.

Language Model

We used a simple one-directional RNN with Long Short Term Memories (LSTMs) to do the language modeling. It is a one-layer RNN with an embedding size of 300 with 300 hidden-layers.

Representation Levels

We perform our experiment on four levels of representation, as shown in Table 5.1. We use the Polyglot[1] wrapper for Morfessor to get the morpheme segmentations. For BPEs we used YouTokenToMe[2] implementation of the algorithm.

Level	Vocabulary size
Character	53
BPE	5000
Morpheme	4970
Word	10000

Table 5.1: The investigated levels of representation and their vocabulary size.

5.4.2 Results

Figure 5.2 shows the testing scores for different levels of representation as training proceeds. As it shows, BPE and morpheme level representation work better than the character level one. Although word-level representation converges faster, BPE and morpheme level curves achieve lower scores after their convergence to the minimum point. We can conclude that subword (BPE and morpheme)-level representations result in better performance of the language model. And, importantly, subwords converge faster to a lower minimum than characters. Therefore, this validates our hypothesis that by extracting morphemes from a sequence, the downstream tasks' performance can be improved. Therefore, we implement our proposed method for inducing

[1] https://polyglot.readthedocs.io/en/latest/MorphologicalAnalysis.html
[2] https://pypi.org/project/youtokentome/

morphemes in the following section.

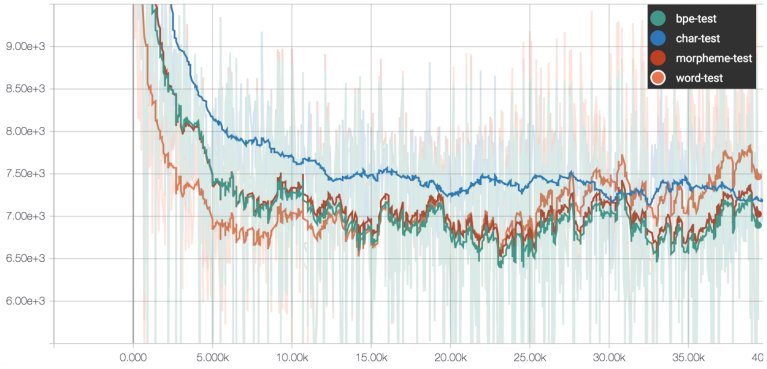


Figure 5.2: Negative of log-likelihood scores on testing data at different levels of representations.

5.5 Main Experiments

In this section, we first describe the setup of our experiments and then, explain the experimental results. To reduce the complexity of our problem, we only consider two segmentation-rules; 0 for no segment and 1 for segment. As a result, the morpheme induction model is simply segmenting the input sequence into morpheme-like segments. Therefore, we use the term *segmenter* for our morpheme induction model in this section. In the experiments, we first simplify our problem to word-level and sentence-level segment prediction, while using the outputs of off-the-shelf tool called Morfessor (Virpioja, 2013). In other words, we replace the language model with Morfessor in our proposed method (5.1) and use it as the evaluator to generate a score for the segmenter. Thus, we are able to train the segmenter in a supervised manner and make sure that our model is capable of learning the task. Finally, we experiment with our proposed method by training a Language Model as the evaluator.

5.5.1 Experimental Setup

Dataset

For these experiments, we utilize the raw version of WikiText2 (Merity et al., 2016). It contains more than 2 million tokens extracted from Wikipedia articles. Following Kawakami et al. (2017), we replaced the characters with a frequency of less than 25 with a special character. Moreover, we lower-cased all the dataset to reduce the complexity of our problem.

Segmenter (Morpheme Induction Model)

For the segmenter, we initially used a simple Transformer with 2 heads and 2 layers. In some experiments we changed the architecture to 4 heads and 4 layers. In addition, we find the Bi-

LSTM architecture proposed at (Cao and Rei, 2016) to be useful. Figure 5.3 shows how the two layers are concatenated to produce the hidden representations. In each time-step the word is split into two non-overlapping parts where they build a specific representation of the word.

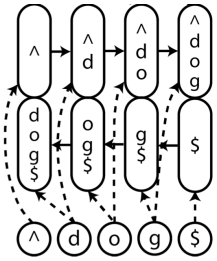


Figure 5.3: The architecture proposed in Cao and Rei (2016)

Language Model

For the language model, we used a 1 layer LSTM with 256 hidden units and an embedding size of 256.

Baseline Function Estimator

For the baseline of the REINFORCE algorithm which estimates the value of each state, we used a simple Feed Forward neural network with 2 layers and hidden units of size 100.

5.5.2 Word-level Segment Prediction with Morfessor as Evaluator

In these experiments we input a word into the morpheme induction model and train it using the outputs of Morfessor by defining different reward functions. We can obtain the correct decision at each position from the outputs of Morfessor.

Sequence-level Reward

We consider the reward function at sequence-level as

$$\mathscr{R}(s_1, s_2, \ldots, s_n) = \alpha \times \text{wrong splits} + (1 - \alpha) \times \text{wrong non-splits} \tag{5.6}$$

In this equation, we consider **wrong splits** as the number of wrong decisions where the model should output a split (1) but outputs non-split (0). Similarly, for **wrong non-splits** we mean the number of wrong decisions by the model where it should output 0 but outputs 1. The parameter α controls the ratio of these two terms. Since most of the positions are non-splits, this ratio is set to 0.75 to encourage the model to split at some positions. We utilize REINFORCE algorithm with a baseline, where the baseline score is obtained from a value function estimator.

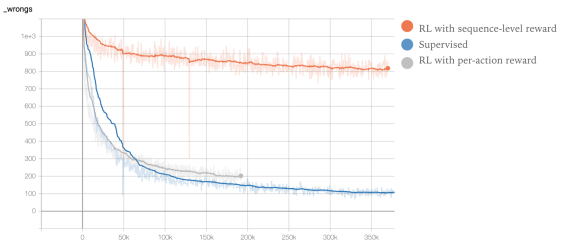


Figure 5.4: Performance (number of wrong decisions) of different reward strategies at word-level using the Morfessor as evaluator.

Per-action Reward

We consider the reward function as

$$\mathscr{R}(s_1, s_2, \ldots, s_n) = [r_1, r_2, \ldots, r_n] \tag{5.7}$$
$$r_i = +1 \text{ if } s_i \text{ is correct, else } -1$$

and we utilize simple REINFORCE algorithm to train our models. Since the rewards are already balanced around zero there is no need to have a baseline.

Results

Figure 5.4 shows the performance of different reward functions while using the Transformer Architecture. We observe that the per-action rewards help the model to nearly achieve the performance of a supervised model, while the sequence-level does not.

5.5.3 Sentence-level Segment Prediction with Morfessor as Evaluator

Similar to the word-level experiments, we try the same reward functions in at sentence-level (Equations (5.6) and (5.7)), where the whole sentence is fed to the model.

Results

Figure 5.5 shows the results of training a Transformer and Bi-LSTM with the given rewards. RNN converges faster in comparison to the Transformer, although at the end both of them have similar performance. Interestingly, sequence-level reward works as well as the per-action reward in the number of wrong decisions it makes and also better in the number of wrong splits. This is probably due to the ratio α we have for sequence-level reward for different types of mistakes the model makes.

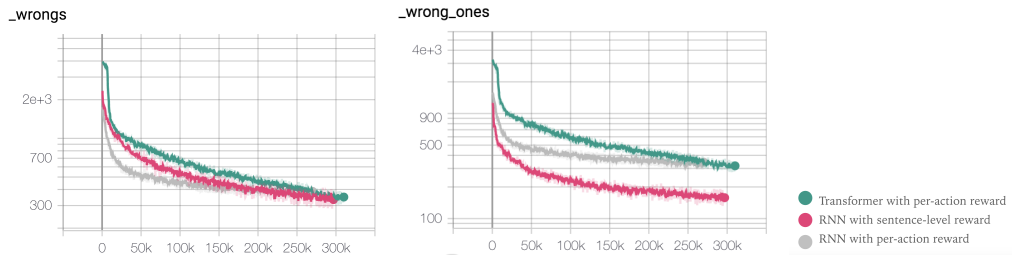


Figure 5.5: Performance of different reward strategies at sentence-level using the Morfessor as evaluator. The figure on the left shows number of wrong decisions the model made, and the figure on the right shown number of wrong splits.

5.5.4 Sentence-level Segment Prediction with a Language Model as Evaluator (Main Idea)

In these experiments, we feed in sentences of length less than 128 to our segmenter model. We let the language model reach the same state as the segmenter by training it for 1 epoch on the outputs of randomly initialized segmenter. This helps the language model to provide meaningful scores for the segmenter outputs and helps the two models progress in the same direction. As transformers are shown to be unstable when training with an RL objective (Parisotto et al., 2019), we decided to use RNN in these experiments. Moreover, the results of our previous experiment show that RNN converges faster, hence, it is a better choice for implementing our idea.

Sequence-level Rewards

We try different reward functions at sequence level as follows:

- Language model score with entropy regularization: we penalize the model with the output of the language model. We refer to this score as $\mathrm{LM}(S,X)$, and we also add an entropy regularization term with ratio γ to stop the model from converging very fast.

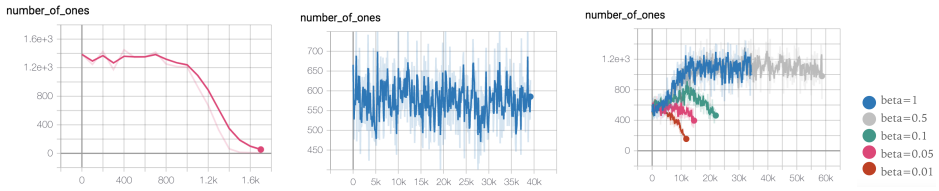
$$L(S,X) = -\log P(S,X) + \gamma \mathscr{H}(P(S|X)) = \mathrm{LM}(S,X) + \gamma \mathscr{H}(P(S|X)) \tag{5.8}$$

- Language model score with the inspiration that we have to penalize $P(X)$ not $P(S,X)$,

$$\begin{aligned} P(X) &= P(S,X)/P(S|X) \\ \log P(X) &= \log P(S,X) - \log P(S|X) \\ L(S,X) &= -\log P(X) = \mathrm{LM}(S,X) + \log P(S|X) \end{aligned} \tag{5.9}$$

  In this loss, we are encouraging the model to have high entropy while having a high language modeling score.

- Language model score and space boundary prediction penalty with entropy regularization:



(a) Language model score with entropy regularization

(b) $P(X)$ as score

(c) Language model score and space boundary prediction penalty with entropy regularization

Figure 5.6: Performance of different sequence-level rewards

In this reward we also add correct prediction of spaces (splitting at that position) to the reward function with a ratio of β.

$$L(S,X) = LM(S,X) + \beta \times \text{wrong-prediction-at-spaces} + \gamma \mathcal{H}(P(S|X)) \tag{5.10}$$

Results

Figure 5.6 shows the results of training our model with different reward functions. Using the language modeling loss (Equation 5.8) collapses to word-level segmentation, since no split decisions are made by the model. Moreover, using $P(X)$ as reward diverges. Finally, language model loss and space boundary prediction collapses to either character-level segmentation for high βs or word-level for low βs.

Per-action Rewards

In addition to sentence-level rewards, we also consider per-action rewards as follows:

- difference in the language model score when we flip the action and fix others,

$$r_i = \text{LM}([s_0,\ldots,s_i,\ldots,s_n],X) - \text{LM}([s_0,\ldots,\bar{s_i},\ldots,s_n],X) \tag{5.11}$$

- difference in the language model score when we *resample* the action. So if the action flips we consider the difference in the language model score as in Equation 5.11 and if it does not, the reward would be 0.

Results

Figure 5.7 shows the performance of different per-action rewards. As it is shown, none of them work in the desired way. The action flipping strategy collapses to character-level and the action resampling one does not converge.

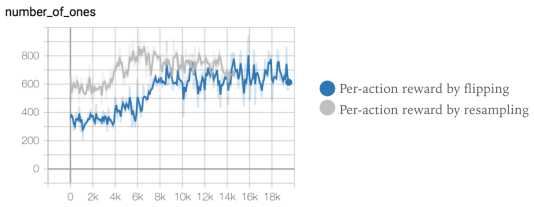


Figure 5.7: Performance of different per-action rewards

5.6 Conclusions and Discussions

In this chapter, we explored a task-driven approach towards inducing entities. Namely, to discover the units which would improve the downstream task's performance. For the downstream task, we consider the generic task of Language Modeling. For the unit discovery module, we predict the segmentation boundaries at every character position.

We first validated our idea that representing input at the level of subwords leads to better performance in the language model compared to characters and words. Then, we evaluated the capability of our unit discovery module when the evaluator is an off-the-shelf model, i.e., Morfessor. The results conveyed that the model is capable of learning the boundaries that Morfessor outputs both with a supervised signal and with an indirect Reinforcement Learning signal. Lastly, since our goal was to learn the boundaries in an unsupervised fashion, we trained the language model simultaneously with our unit discovery module. We tried different reward strategies to train the unit discovery module with RL, but none of them worked as expected. The models either diverged, or converged to trivial solutions such as outputting characters or words. This approach is a rather convoluted approach towards solving the task since the segmentation decisions are discrete and thus, RL is necessary for training the model. We think that using RL is complicated and does not work with such a small signal from the LM. We leave further investigations of this method to future work.

6 Conclusions and Future Work

In this chapter, we first conclude the findings of this thesis in Section 6.1. Afterward, we discuss the future directions that we envision for our work in Section 6.2.

6.1 Conclusions

In this thesis, we introduced and addressed the entity induction problem. That is, to find a more coarse-grained representation of the input without supervision where every entity represents a more abstract concept. We summarize our contributions in every chapter in the following.

In Chapter 2, we discovered entities at the level of morphemes from a sequence of characters. We learned these entities by adapting a method called Slot Attention (Locatello et al., 2020) for our task. Intuitively, each slot vector binds to a specific part of the input via an iterative top-down attention mechanism. We proposed to initialize the slots with a separate parameter while adding a fixed amount of noise to them. In order to dynamically decide the number of required entities per input sentence, we utilized an L_0 regularization method called L_0Drop (Zhang et al., 2021). We proposed to do bi-directional probing to evaluate if the induced entities are at the same level of abstraction as expected. In addition, we demonstrated the potential benefits of our model over a character-based and segmentation-based model in a downstream topic classification task.

In Chapter 3, we induced entities at the level of words from a sequence of characters. We integrated NVIB (Henderson and Fehr, 2023) layers into the last Transformer encoder layers and trained the model with a noisy reconstruction objective. We showed that a model with NVIB has better robustness over synthetic noise. And that it is more linguistically informed in terms of both a highly-semantic topic classification task and a sentence representation benchmark.

In Chapter 4, we induced entities at the level of phrases from a sequence of tokens in a multi-modal setup, i.e., images and their captions. The goal was to learn entities which are at the level of objects in the image. We applied a grouping algorithm that learns to group input tokens with top-down attention. Our results verified that inducing phrase-like entities leads to a better

understanding of vision and language in comparison to representing text with only one vector.

In Chapter 5, we explored discovering entities at the level of morphemes from a character sequence with a different approach than the previous chapters. Namely, we modeled the problem as deciding where the morpheme-segment boundaries should be. Then, given the output of the segmenter, we aimed to improve a language model simultaneously. We trained our model with RL. Eventually, we did not find this approach effective in inducing meaningful units. We hypothesize that small changes in the language model are not enough for the RL algorithm to work successfully.

In summary, our findings suggest that it is possible to induce entities through compression. In Chapters 2 to 4, we achieve compression by reducing the number of vectors (and reducing the dimension in some cases) which represent the input. In addition, in order to encourage segmentation, a reconstruction objective is helpful. We also propose and gather different ways of evaluating the resulting units. Namely, for intrinsic evaluation: visualizing the attention maps, greedy segmentation or bi-directional probing are beneficial. And for extrinsic evaluation, we can evaluate the robustness and how linguistically informed they are, as well as their potential benefits in downstream tasks.

All of our proposed methods are scalable to longer sequences by tuning the number of entities either directly as in Text GroupViT or indirectly by tuning the ratio of sparsity losses. While our methods introduce new parameters to the model, they do not increase the computational complexity of the resulting model. However, they might slightly increase the training time.

6.2 Future Work

In this section, we discuss future directions for our work.

6.2.1 Unsupervised Entity Discovery

We believe that the research on entity discovery in text is still in its early stages, presenting many avenues for further exploration. The first is to explore and develop more sophisticated models to perform the task. This includes exploring other object discovery modules from vision or developing specific NLP unit discovery models. The second is to investigate the effect of having a hierarchy of entity discovery modules on top of each other. It would be interesting to see how the resulting model behaves and if the modules are able to learn different levels of abstraction within the same module. The third is to address low-resourced languages and design or adapt entity discovery modules for those languages. Lastly, we can develop models where the entities are directly optimized for downstream tasks in order to have tokenization-free models.

6.2.2 Evaluation of Entity Discovery Modules in Text

We think that evaluating entity discovery modules in text is still a challenge. A direction could be to explore more downstream tasks and investigate if entity discovery is advantageous for them or not. It is also possible to develop task-specific datasets or benchmarks for this task. Moreover, we presume that integrating an entity discovery module could be a potential defense mechanism against adversarial attacks. In particular, since these units are obtained by creating a bottleneck, they can discard noises injected into the input from the adversary.

6.2.3 Discovering Aligned Objects in Different Modalities

Our work in Chapter 4 was a step towards learning aligned objects in two modalities, i.e., image and text. However, due to the limitations in computing resources, we were not able to perform end-to-end training for both modalities. We believe that learning objects for both modalities at the same time will help the model gain a better understanding of the world and in particular, the visual concepts. In addition, exploring other modalities such as audio and vision where the objects are expressed through speech is an interesting direction.

A Appendix for Chapter 2

A.1 Supplementary Results

A.1.1 Visualization of the Attention Maps

Attention of Slots over the Input

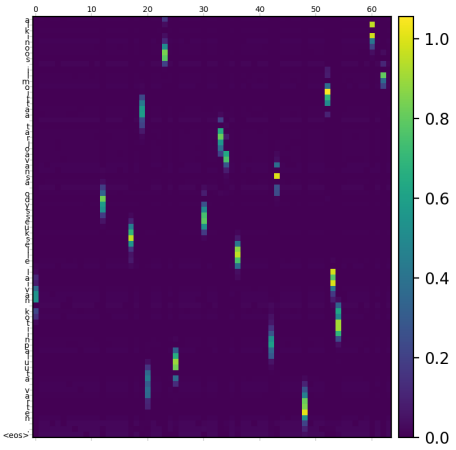
In Figures A.1 and A.2, we illustrate the attention of slots over the input as well as the attention of decoder over the slots for the same input. The two attention maps show similar patterns, but on the decoder side the attention weights are higher and therefore the patters are more visible. We believe that at each generation step, the decoder only attends to the slots which contain the information about that specific character and thus, the attention patterns are stronger at decoding time. For these reason, we used the attention of the decoder over the slots in section 2.4.4. Interestingly, in Figure A.2b, there are some slots which are only attending to the space boundaries (the horizontal bands).

Decoder Attention for the Stride-Based models

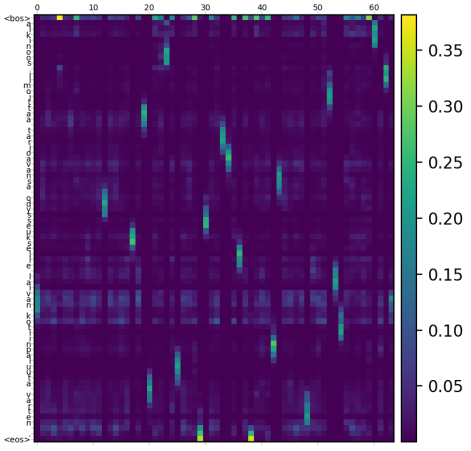
Figure A.3 visualizes the attention of decoder over the vectors of different stride-based models. As expected, the vertical bands are often of the same length and too much overlapping for larger strides. As a result, the bands do not correspond to meaningful units in the input in contrast to the slot attention based model (see Figure 2.2c).

A.1.2 Bidirectional Probing Results' Tables

Tables A.1 to A.11 show the detailed results of the performance of forward and reverse probing tasks visualized in Figure 2.4 (Reverse reconstruction vs F1). n_input denotes the input length which is 128 in our experiments.



(a) Attention of decoder (y-axis) over slots (x-axis).



(b) Attention of slots (x-axis) over the input (y-axis) before the L_0Drop.

Figure A.1: Illustration of the Attention of Slots over the input vs the Attention of decoder over the slots for Finnish language.

stride	stride=2	stride=3	stride=4	stride=6
F1	0.8647	0.7965	0.8065	0.7678
Loss(1e+5)	1.318	1.7084	2.2163	3.4569
P	0.9376	0.9125	0.9236	0.9514
R	0.8106	0.7335	0.732	0.6667
Acc	0.9114	0.8923	0.8811	0.8596
Reverse reconstruction	15.14	-30.44	-49.04	-105.1

Table A.1: Stride-based models for English with BPE targets

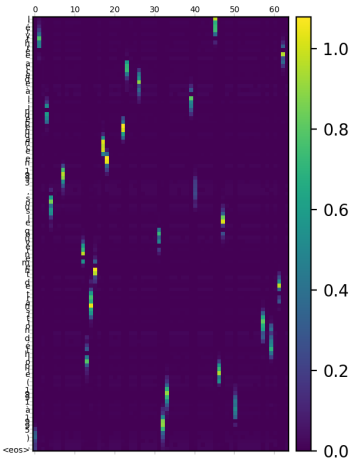
limit (r)	r=2	r=3	r=4	r=5	r=6	r=8
F1	0.8067	0.8636	0.8457	0.809	0.7878	0.7677
Loss(1e+5)	1.5364	1.2951	1.5757	2.0522	2.3737	2.8781
P	0.8634	0.924	0.9394	0.9555	0.9635	0.9763
R	0.7687	0.8213	0.7819	0.7196	0.6863	0.6517
Acc	0.8845	0.9139	0.9058	0.8934	0.8865	0.8798
Reverse reconstruction	-10.56	-58.98	-84.32	-129	-150.1	-204.4

Table A.2: Slot Attention based models for English with BPE targets

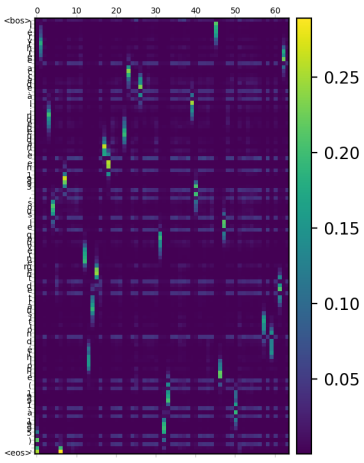
A.2 Settings

A.2.1 Data

As for the datasets, we lowercased the text and retained the characters which occur more than 25 times in the corpus, following Kawakami et al. (2017). We replace the low-frequent characters with an unknown placeholder. Table A.13 shows the licenses of each dataset that we used in our



(a) Attention of decoder (y-axis) over slots (x-axis).



(b) Attention of slots (x-axis) over the input (y-axis) before the L_0Drop.

Figure A.2: Illustration of the Attention of Slots over the input vs the Attention of decoder over the slots for French language.

stride	stride=2	stride=3	stride=4	stride=5	stride=6
F1	0.8108	0.8086	0.7835	0.792	0.7461
Loss(1e+5)	2.6374	2.5115	2.6292	2.7607	3.5556
P	0.8971	0.883	0.8812	0.9083	0.9155
R	0.7479	0.7567	0.7179	0.7179	0.6498
Acc	0.8973	0.8964	0.8833	0.8897	0.8688
Reverse reconstruction	21.26	-51.9	-60.54	-151	-116

Table A.3: Stride-based models for English with Morpheme targets

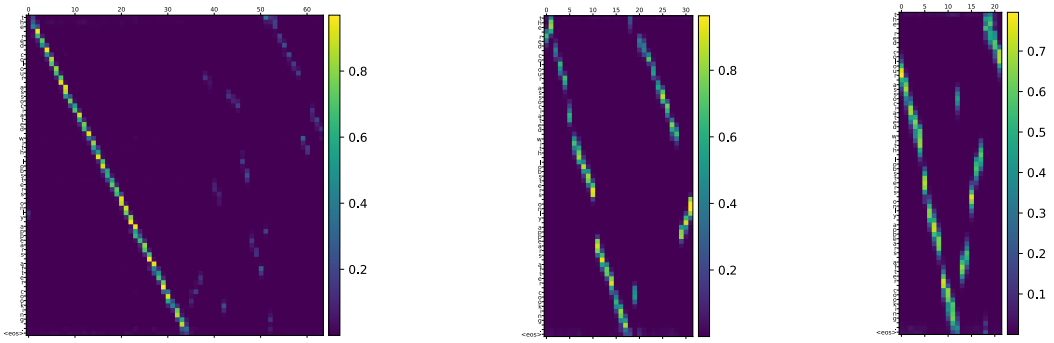
limit (r)	r=2	r=3	r=4	r=5	r=6	8
F1	0.7821	0.8394	0.8136	0.8022	0.7814	0.7705
Loss(1e+5)	2.0958	1.7118	2.1506	1.9883	2.3083	2.4825
P	0.8353	0.8844	0.8952	0.9083	0.9157	0.9331
R	0.7482	0.8101	0.7577	0.7383	0.7039	0.6789
Acc	0.884	0.9116	0.9017	0.9025	0.896	0.8949
Reverse reconstruction	5.975	-43.17	-84.65	-90.6	-123.7	-122.7

Table A.4: Slot attention based models for English with Morpheme targets

experiments. We used the same train/validation/test splits as provided in the mentioned datasets.

A.2.2 Main Model Settings

Table A.14 shows the remaining list of hyperparameters in training the main model. We scheduled the λ parameter in the training loss to start with a low value of 2×10^{-5} and exponentially increase



(a) stride=2 (b) stride=4 (c) stride=6

Figure A.3: Attention of decoder over the stride-based model vectors (x-axis) while generating every character (y-axis). The target sentence is "*the red colour associated with lobsters only appears after cooking.*".

stride	stride=2	stride=3	stride=4	stride=6
F1	0.8435	0.8126	0.8087	0.7622
Loss(1e+5)	2.7637	4.3381	2.6835	3.7541
P	0.9175	0.9012	0.9162	0.9458
R	0.7905	0.7529	0.7388	0.6606
Acc	0.9026	0.8872	0.8842	0.8595
Reverse reconstruction	15.28	-24.36	-48.84	-107.9

Table A.5: Stride-based models for English with Morfessor targets

limit (r)	r=2	r=3	r=4	r=5	r=6
F1	0.7662	0.8318	0.7892	0.7735	0.7451
Loss(1e+5)	2.7095	1.9951	2.7874	2.728	3.1524
P	0.8533	0.9089	0.9128	0.9323	0.9367
R	0.7091	0.7787	0.7087	0.6793	0.6383
Acc	0.8636	0.8975	0.8777	0.8763	0.8668
Reverse reconstruction	-8.193	-57.02	-89.83	-122.8	-140

Table A.6: Slot attention-based models for English with Morfessor targets

stride	stride=2	stride=3	stride=4	stride=5	stride=6
F1	0.6239	0.8365	0.8219	0.7911	0.754
Loss(1e+5)	5.3584	2.1264	2.669	3.6093	4.5231
P	0.8814	0.921	0.9356	0.9485	0.9584
R	0.4916	0.7756	0.7454	0.6939	0.6399
Acc	0.7767	0.885	0.8769	0.8603	0.8392
Reverse reconstruction	-10.51	-58.4	-93.31	-129	-160.7

Table A.7: Stride-based models for French with BPE targets

limit (r)	r=2	r=3	r=4	r=5	r=6
F1	0.7941	0.8422	0.8238	0.8082	0.7829
Loss(1e+5)	2.0313	1.8495	2.3189	3.0405	3.7684
P	0.8694	0.9247	0.934	0.964	0.9786
R	0.7403	0.7826	0.7488	0.7087	0.665
Acc	0.8594	0.8876	0.8789	0.8725	0.8593
Reverse reconstruction	-26.5	-68.98	-105.4	-164.2	-204.7

Table A.8: Slot attention-based models for French with BPE targets

stride	stride=2	stride=3	stride=4	stride=5	stride=6
F1	0.6409	0.7872	0.7922	0.7772	0.7184
Loss(1e+5)	5.5099	3.1225	3.0613	3.6183	4.8101
P	0.9014	0.8906	0.8936	0.914	0.9281
R	0.5051	0.714	0.722	0.6885	0.5989
Acc	0.8107	0.872	0.8746	0.8682	0.8438
Reverse reconstruction	-5.61	-85.57	-207.7	-204.3	-168.5

Table A.9: Stride-based models for French with Morpheme targets

limit (r)	r=2	r=3	r=4	r=5	r=6
F1	0.7277	0.7845	0.7976	0.8048	0.7896
Loss(1e+5)	3.5534	2.9845	2.835	2.565	3.0282
P	0.8403	0.876	0.8889	0.9195	94.05
R	65.26	0.7195	0.7329	0.7269	0.6921
Acc	0.8406	0.8679	0.8762	0.8829	0.8777
Reverse reconstruction	-8.35	-69.87	-148.4	-190.2	-181.9

Table A.10: Slot attention-based models for French with Morpheme targets

stride	stride=2	stride=3	stride=4	stride=5	stride=6
F1	0.6794	0.7919	0.7902	0.7632	0.704
Loss(1e+5)	5.5356	3.6372	3.7099	4.5012	5.8283
P	0.9041	0.9086	0.9189	0.9368	0.9454
R	0.5533	0.7117	0.7045	0.6578	0.5738
Acc	0.8078	0.8615	0.8619	0.8481	0.821
Reverse reconstruction	-10.33	-56.74	-90.33	-120.3	-150.1

Table A.11: Stride-based models for French with Morfessor targets

it every 10 epochs until it reaches a certain limit. In particular, we schedule the λ to exponentially increase with ratio 2 for English until reaching $6.4e-4$ and 1.5 for the rest of languages. More specifically, we stop the exponential increase after reaching $3e-4$ for German and Spanish and $5e-4$ for French, Finnish and Czech. We tried the stopping thresholds ranging from 3×10^{-4} to 6×10^{-4}. These scheduling values lead to having roughly as many slots as the average number BPE

limit (r)	r=2	r=3	r=4	r=5	r=6
F1	0.7383	0.7979	0.7877	0.7833	0.7621
Loss(1e+5)	4.2044	3.1744	3.4661	3.4707	4.229
P	0.867	0.9016	0.9117	0.9466	0.963
R	0.6474	0.7256	0.7044	0.681	0.6425
Acc	0.8302	0.8646	0.8602	0.8619	0.8521
Reverse reconstruction	-22.51	-66.06	-107.5	-160.8	-205

Table A.12: Slot attention-based models for French with Morfessor targets

dataset	license
WikiText2 (Merity et al., 2016)	Creative Commons Attribution-ShareAlike 3.0 Unported License (link to dataset)
Multilingual Wikipedia Corpus (MWC) (Kawakami et al., 2017)	https://aclanthology.org/P17-1137/
MorphoLex (Sánchez-Gutiérrez et al., 2018; Mailhot et al., 2020)	Creative Commons Attribution-NonCommercial-ShareAlike 4.0 License (CC BY-NC-SA 4.0) (https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-4629#)

Table A.13: Data licenses

units per sentence. We also tried our model with statistic λ in the $\{10^{-4}, [1,3,5,6,7,8] \times 10^{-5}\}$ which did not lead to stable results at training time. We tried 16, 32, and 64 slots in our experiments. We chose the number of slots to be half of the maximum sequence length (128) as this is a reasonable upperbound which also matches the maximum number of BPE or Morfessor units. We tried the transformer encoder with 4 and 6 layers but qualitatively did not find any improvements. We run the Slot Attention algorithm for T=1 iterations. We choose T=1 iterations for simplicity and efficiency, and because preliminary experiments showed no improvements with more iterations. We leave the investigation of how to get improvements from more iterations to future work.

A.2.3 Forward Probe Settings

We train a probing classifier for mapping a slot's vector to the target representation's vocabulary, namely, $f(m'_i) : \mathrm{R}^{D_{slots}} \rightarrow \mathrm{R}^S$, where S is the number of tokens of the target representation. We apply the classifier with shared parameters over each of our slots and obtain a *set* of predictions, i.e., $\{f(m'_1), f(m'_2), \ldots, f(m'_K)\}$. As we are dealing with a set, during training we need to find a one-to-one matching between the classifier's predictions and the target tokens. Therefore, we use the Hungarian matching algorithm (Kuhn, 1955) for finding the best match in terms of minimizing the classification loss. Consider the best matching as $i_j \rightarrow j$, which matches the i_jth

module	parameter	value
	batch size	16
	learning rate	$1e-4$
transformer	model dimension	256
transformer	feedforward layer dimension	4×256
transformer	dropout rate	0.1
L_0Drop	β	0.66
L_0Drop	ϵ	0.1
Slot Attention	Slot dimension (D_{slots})	128
Slot Attention	MLP hidden dimension	$2 \times D_{slots}$
Slot Attention	GRU hidden dimension	D_{slots}
Slot Attention	δ	$1e-8$
Slot Attention	T	1

Table A.14: Hyperparameters of the main model.

slot with the jth output (i.e., y_j). We then compute the loss as $\mathscr{L} = \sum_{j=1}^{K} l(f(m'_{i_j}), y_j)$, where l is the cross-entropy between the predictions and targets.

Our probing classifier consists of two fully connected layers with ReLU activation function in between the two layers. The hidden dimension of the classifier is the same as the slots' dimension, which is 128.

We use the same datasets as our main model for training and testing the probes. We train BPE with a vocabulary size of 5000 for all languages. For Morfessor, we use the pretrained model and consider the set of its outputs on the training data as our target representation. As for the morpheme targets, we take the morphemes for the words which were available in the linguistically annotated data (i.e., MorphoLex (Sánchez-Gutiérrez et al., 2018; Mailhot et al., 2020)) and for the rest of the words we take Morfessor outputs as an approximation of morphemes.

For training the probing classifiers we take a batch size of 4, since the Hungarian matching algorithm requires a huge amount of memory. We train our classifiers for 200 epochs with Adam optimizer with learning rate of 1×10^{-3}.

A.2.4 Reverse Probe Settings

We illustrate how forward and reverse probing are related to each other in Figure A.4. We first assign a one-hot vector to the tokens in the target set of units (i.e., R^S, where S is the target set size). Then, we learn an embedding layer to map every one-hot vector into a continuous vector with dimension 128. Afterward, we pass the embedded token into two fully connected layers with ReLU activations in between with the hidden dimension 128. We then predict the mean and the standard deviation of a Gaussian distribution. The dimension of the Gaussian distribution d is $D_{slots} = 128$. We use the same matching between the target units and slots as in the forward probing. Namely, we do not run the matching algorithm for this experiment. As for the training

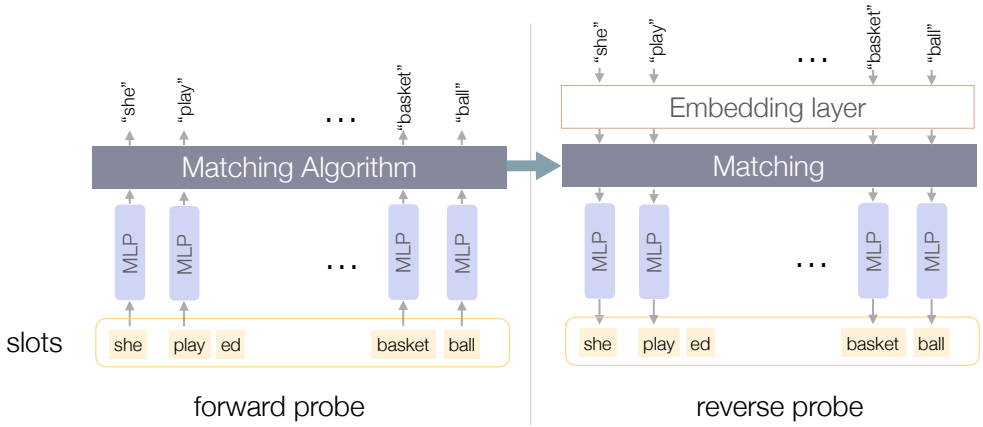


Figure A.4: An illustration of forward and reverse probing.

objective, we minimize the negative log-likelihood of the slot vector given this distribution, i.e., $-\log(p(m_i|\mu_{\text{predicted}}, \sigma_{\text{predicted}}))$ which is equivalent to minimizing $\sum \frac{1}{2\sigma^2_{\text{predicted}}}(m_i - \mu_{\text{predicted}})^2 - \log(\sigma_{\text{predicted}})$. We train this model with Adam optimizer and the learning rate 1×10^{-4} for 200 epochs. We report the best evaluation loss on test set.

A.2.5 Arxiv Classifier Settings

The hidden dimension of the MLP and also the query, key and value matrices are set to the same dimension as the slots, namely, 128. We train the classifier with a batch size of 32 for 200 epochs with Adam optimizer with a learning rate of $1e-4$. For the Arxiv-L dataset each subarea has 1000 samples which would be 20000 samples in total.

A.3 Infrastructure

We use PyTorch version 1.2.0 framework and Python version 3.6.9 for implementing our code. Table A.15 shows the rest of the libraries we use. We run our code on a single GPU with model GTX1080ti and the operating system Debian10 (Buster) 64-bit. We use the same compute for all of our experiments including training the models and probes. The training time for the main model is five hours and for the probings is around 2 days. For reporting each of the results we run our algorithm once, since it would be too computationally expensive.

package	version	use
NLTK	3.5	sentence and word tokenization
youtokentome	1.0.5	BPE implementaion
polyglot	16.7.4	morfessor implementation
matplotlib	3.3.2	attention maps visualization
scipy	1.2.2	hungarian algorithm implementation
numpy	1.19.1	

Table A.15: List of packages and their versions.

B Appendix for Chapter 3

B.1 Training Details

General Training

All models are trained, without pretraining, using the same encoder and decoder configuration for comparability. Our encoder size is defined by the base Transformer (Vaswani et al., 2017b) such that we have a six layer Transformer encoder. However, we use a two layer decoder to ensure the task is not learned in the decoder alone. We use a single attention head. The size for the word embedding vectors and model projections are 512 and feed forward dimensions 512, which leads to models of approximately 12-17 million trainable parameters. An English character level tokeniser is used for tokenisation with a vocabulary of approximately 100 characters. During training we use: a learning rate of $1e^{-3}$ with a cosine cool-down over all steps, RAdam optimiser (Liu et al., 2020) with mixed precision (FP16), a batch size of 512, gradient norm clipping 0.1 and trained for 55 epochs ($\approx 8K$ steps). The number of steps were selected considering model convergence and minimising computation time. We use a dropout rate of 0.1. The input is noised at each batch with a probability of character deletion of 0.1. Each model takes approximately 2.5hrs on a single NVIDIA GeForce RTX 3090.

NVIB Training

Training the models with the NVIB layers requires regularising the representations. The introduction of the exponential activation function (as opposed to ReLU) for the psuedo-count parameter $\boldsymbol{\alpha}$ requires a threshold at test time to be exactly zero. We use a threshold for this at 0.1. During training and testing we enforce a bottleneck between the encoder and decoder by masking the final encoder representations by the aforementioned threshold.

The NVIB hyperparameters λ_G, λ_D and α^{Δ} are selected through hyperparameter tuning. However, during training we only sample once from each component thus the approximation parameter is set to $\kappa^{\Delta} = 1$. We use a Kullback-Leibler annealing divergence strategy where the introduction

of the KL divergence loss is linearly introduced between 30% – 60% of the training steps. This allows the model to learn initial representations, slowly introduce the regularisation and finally learn through the compressed latent representation.

B.2 Hyperparameter Tuning

The models are trained on the Wikitext-2 training dataset using the loss from Equation 3.2. They are tuned on the validation dataset with the aim to be able to reconstruct the character sequence from a noisy input. Following from (Henderson and Fehr, 2023) we set the weights of the Gaussian and Dirichlet KL divergences to be independent of the sentence length n and dimensionality of vectors d:

$$\lambda_D = \frac{1}{n}\lambda'_D \;\; ; \quad \lambda_G = \frac{1}{d}\frac{1}{n}\lambda'_G$$

where λ'_D and λ'_G are fixed hyperparameters. All combinations of the following hyperparameters were considered in a grid search for the respective models:

- $lr = \{1e^{-4}, 1e^{-3}\}$
- $\lambda'_G = \{1e^{-5},\ 1e^{-4},\ 1e^{-3},\ 1e^{-2}\}$
- $\lambda'_D = \{1e^{-2},\ 1e^{-1},\ 1\}$
- $\alpha^{\Delta} = \{0\ , 0.05,\ ...,\ 0.45,\ 0.5\}$

where λ'_G and λ'_D are the weights on the Gaussian and Dirichlet KL divergences. The α^{Δ} represents the conditional prior parameter. The final models' hyperparameters are reported in Table B.1 where the validation cross-entropy (CE) is matched for NVIB and baseline Transformers.

	Transformer	NVIB
NVIB layers	-	3
λ_G	-	$1e^{-2}$
λ_D	-	1
α^{Δ}	-	0.25
Training Steps	$2.5K$	$8K$
Val. CE	0.19	0.19

Table B.1: Hyperparameters for final models evaluated.

The encoders 6 layers are inspired by the base model of Vaswani et al. (2017b). For the Transformer decoder we use only 2 layers such that the decoder is not able to overpower the embeddings from the encoder it sees through cross attention.

NVIB Configuration

For the NVIB layers during experimentation we considered: All layer including NVIB; the last 3 layers including NVIB; and only the final layer including NVIB. When all layers were included it was challenging to get both compression and performance as the regularisation was too strong. Only regularising the last layer managed to reduce the number of vectors but often converged to a single sentence vector with lower, non-comparable validation cross-entropy. Finally, we settled on only regularising the last 3 layers as it gave the model enough flexibility in the lower layers and progressive compression in the higher layers.

B.3 KL Divergence Loss

Henderson and Fehr (2023) define the Kullback-Leibler divergence for NVIB with two terms: the L_D for the Dirichlet distribution weights defined by $\boldsymbol{\alpha}$; and the L_G for the distribution of vectors $\boldsymbol{Z}$ generated by the Gaussian components. We set the approximation parameter $\kappa_0 = 1$. This gives us the following loss terms for the KL divergence, where Γ is the gamma function and ψ is the digamma function:

$$L_D = \log\Gamma(\alpha_0^q) - \log\Gamma(\alpha_0^{p'}) + (\alpha_0^q - \alpha_0^{p'})\left(-\psi(\alpha_0^q) + \psi(\alpha_0^q)\right) + \left(\log\Gamma(\alpha_0^{p'}) - \log\Gamma(\alpha_0^q)\right)$$

where, α_0^q is the sum of all $\boldsymbol{\alpha}$ parameters generated by the NVIB layer. The conditional prior $\alpha_0^{p'} = \alpha_0^p + n\alpha^\Delta$ is controlled by $\alpha_0^p = 1$ and extra pseudo-counts defined by the length n and a hyperparameter α^Δ. The KL divergence between two Gaussians (with diagonal covariance with values $\boldsymbol{\sigma}$ and weighted by the $\boldsymbol{\alpha}$ parameters) is:

$$L_G = \tfrac{1}{2}\sum_{i=1}^{n+1}\frac{\alpha_i^q}{\alpha_0^q}\sum_{h=1}^{d}\left(\frac{(\mu_{ih}^q - \mu_h^p)^2}{(\sigma_h^p)^2} - 1 + \frac{(\sigma_{ih}^q)^2}{(\sigma_h^p)^2} - \log\frac{(\sigma_{ih}^q)^2}{(\sigma_h^p)^2}\right)$$

B.4 Denoising attention function

Henderson and Fehr (2023) generalise the set of vectors input to an attention function to a probability distribution over vectors, and generalise attention to a function of these probability distributions called denoising attention.

Training function

During training, the set of sampled vectors $\boldsymbol{Z} \in \mathbb{R}^{n\times p}$ and their sampled log-probability weights $\log(\boldsymbol{\pi}) \in \mathbb{R}^{1\times n}$ are both output by the NVIB layer, thereby specifying the sampled mixture distribution F. A set of query vectors $\boldsymbol{u}' \in \mathbb{R}^{m\times p}$ is projected into key space by the grouped matrices $\boldsymbol{W}^Q, \boldsymbol{W}^K \in \mathbb{R}^{p\times d}$ to $\boldsymbol{u} = (\boldsymbol{u}'\boldsymbol{W}^Q(\boldsymbol{W}^K)^T)$. The keys' dimensionality d is used for scaling.

Denoising attention can then be computed as:

$$\text{softmax}\left(\frac{1}{\sqrt{d}}\boldsymbol{u}\boldsymbol{Z}^T + \log(\boldsymbol{\pi}) - \frac{1}{2\sqrt{d}}\|\boldsymbol{Z}\|^2\right)\boldsymbol{Z} \tag{B.1}$$

For self-attention, we define the original query vectors $\boldsymbol{u}'$ to be the set of vectors input to the NVIB layer, before projecting to DP parameters and sampling.

Testing function

During test time, we do not sample F, but instead use the mean of the posterior distribution. The test-time denoising attention can then be computed as:

$$\text{softmax}\left(\boldsymbol{u}\left(\frac{\boldsymbol{\mu}}{(\boldsymbol{\sigma}^r)^2}\right)^T + \log(\frac{\boldsymbol{\alpha}}{\alpha_0}) - \left(\frac{1}{2}\left\|\frac{\boldsymbol{\mu}}{\boldsymbol{\sigma}^r}\right\|^2\right)^T - \mathbf{1}_p\left(\log(\boldsymbol{\sigma}^r)\right)^T\right) \times \left(\frac{(\boldsymbol{\sigma})^2}{(\boldsymbol{\sigma}^r)^2} \odot (\mathbf{1}_n^T\boldsymbol{u}) + \frac{\sqrt{d}}{(\boldsymbol{\sigma}^r)^2} \odot \boldsymbol{\mu}\right) \tag{B.2}$$

where $\mathbf{1}_p$ is a row vector of p ones and $(\boldsymbol{\sigma}^r)^2 = (\sqrt{d} + (\boldsymbol{\sigma}^q)^2)$.

B.5 SentEval Tasks

B.5.1 Model

We employ the Aggregating probe for performing this task. We froze our models and trained the probes for 10 epochs with a batch-size of 128. The hidden dimension for the probe is set to 256. We trained the model with Adam optimizer with a learning rate of $1e-4$. We report the test set accuracy for the best-performing model in terms of validation accuracy.

B.6 Arxiv Classification Task

Our goal here is to compare the representations and not have ultimate performance in the task, thus we do not fine-tune the models. Hence, we only evaluated our models on the large division of the task, i.e., ArXiv-L which consist of 1000 samples for each sub-area leading to 20000 samples in total. We employ the Attention-based probe to perform this task as it is quite a challenging task which requires the information in the vectors to be better managed by the Attention mechanism and also more similar to the way the model itself would perform the task. The hidden dimension of the MLP is set to 256 and the query, key, and value matrices are set to the same dimension as the model dimension, namely, 512. We train the classifier with a batch size of 256 for 50 epochs with Adam optimizer with a learning rate of $1e-3$. Following Hofmann et al. (2022) we report the test F1 for the best-performing model in terms of validation F1.

B.7 Visualisations

In Figures B.1 to B.4 we include additional visualisations of the self-attention weights.

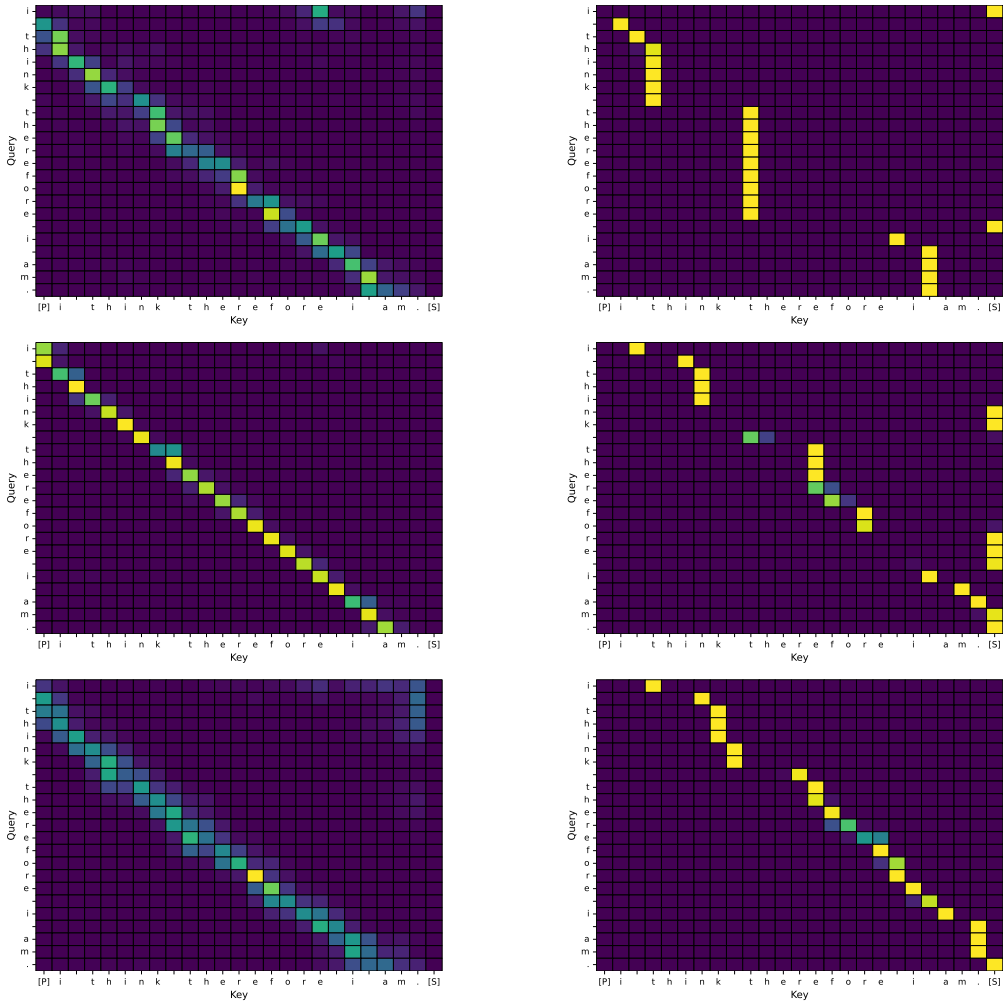


Figure B.1: Self-attention patterns of the last 3 layers of 6-layer Transformer encoders. **Left**: Standard self-attention. **Right**: With NVIB regularisation. **Sentence**: "I think therefore I am." Dark purple is 0 and light yellow is 1 for the attention values.

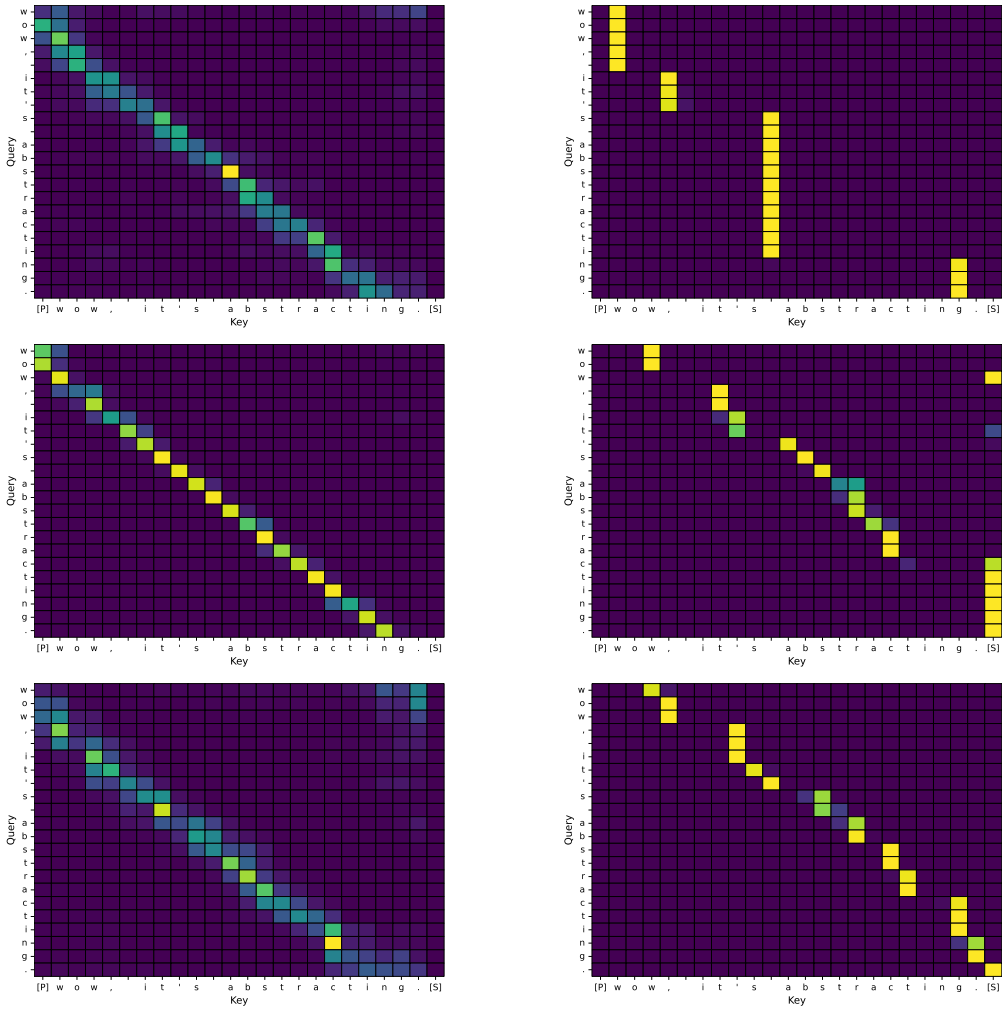


Figure B.2: Self-attention patterns of the last 3 layers of 6-layer Transformer encoders. **Left**: Standard self-attention. **Right**: With NVIB regularisation. **Sentence**: "Wow, it's abstracting." Dark purple is 0 and light yellow is 1 for the attention values.

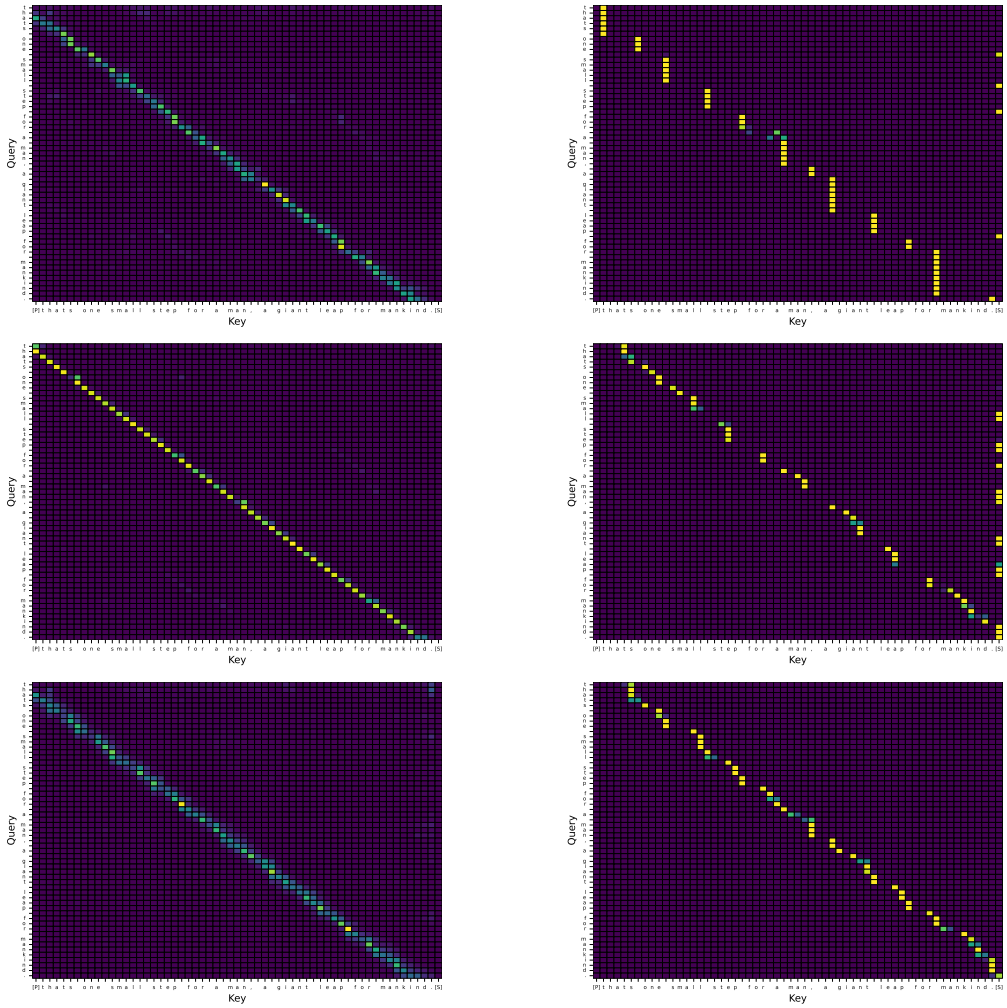


Figure B.3: Self-attention patterns of the last 3 layers of 6-layer Transformer encoders. **Left**: Standard self-attention. **Right**: With NVIB regularisation. **Sentence**: "Thats one small step for a man, a giant leap for mankind." Dark purple is 0 and light yellow is 1 for the attention values.

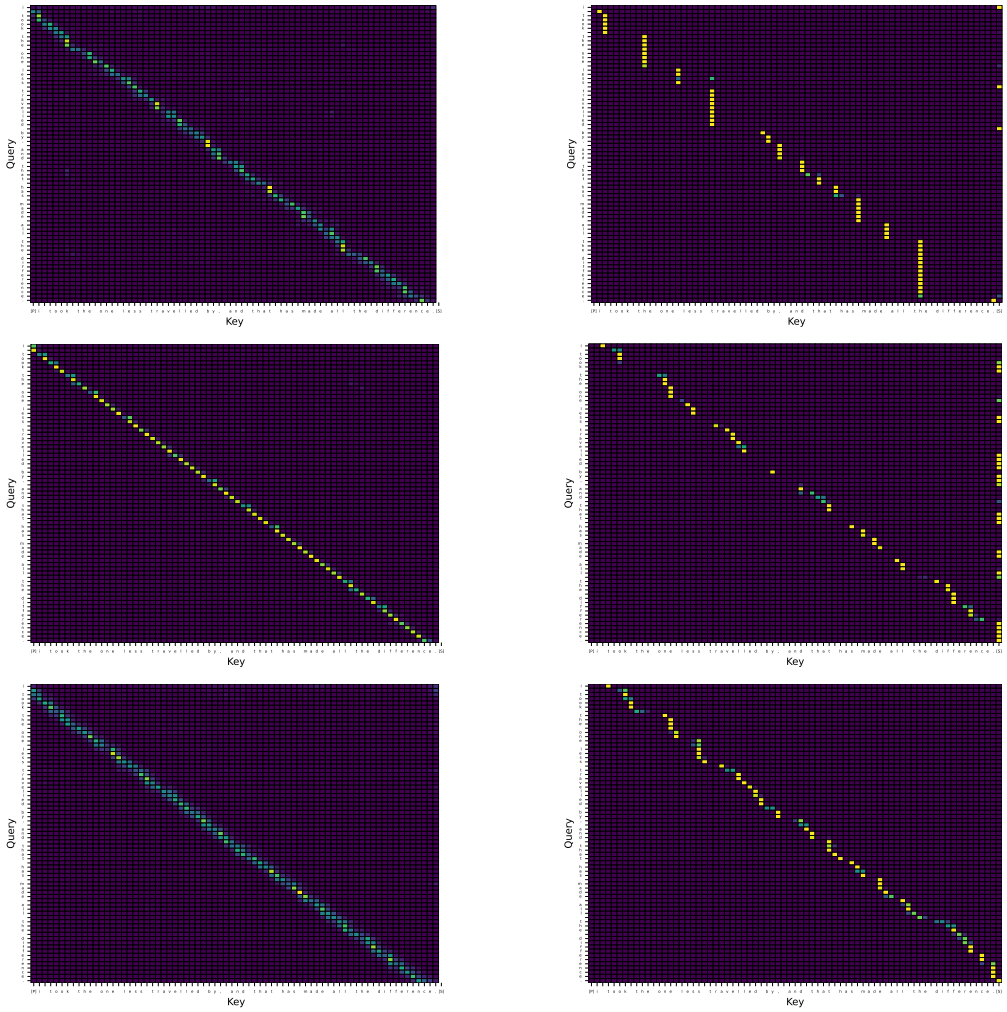


Figure B.4: Self-attention patterns of the last 3 layers of 6-layer Transformer encoders. **Left**: Standard self-attention. **Right**: With NVIB regularisation. **Sentence**: "I took the one less travelled by, and that made all the difference." Dark purple is 0 and light yellow is 1 for the attention values.

C Appendix for Chapter 4

C.1 Artifacts statements

The datasets used do not have personally identifying information or offensive content. We provide the list of datasets used and the corresponding licenses in Table C.2, which are all consistent with our academic use.

C.2 Descriptive Statistics

Our results are from single runs for all the models trained.

C.3 Packages

We provide a list of packages used in our code in Table C.1.

C.4 AI Assistants

We utilized AI assistants for minor text editing and code completion tasks during the development of the model.

Package	version
Python	3.7
PyTorch	1.8
webdataset	0.1.103
mmsegmentation	0.18.0
timm	0.4.12
nltk	3.8.1
ftfy	6.1.1
regex	2023.6.3

Table C.1: The packages used in our code development

.

Dataset	License
GCC3M	Google license (link)
SVO-Probes	Creative Commons Attribution 4.0 International Public License (CC BY 4.0)
FOIL-COCO	Creative Commons Attribution 4.0 License
Flikr	Creative Commons Attribution 0: Public Domain

Table C.2: Datasets and their licenses.

Bibliography

Al-Rfou, R., Choe, D., Constant, N., Guo, M., and Jones, L. (2019). Character-level language modeling with deeper self-attention. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3159–3166.

Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. (2017). Deep variational information bottleneck. In *International Conference on Learning Representations*.

Ataman, D., Aziz, W., and Birch, A. (2020a). A latent morphology model for open-vocabulary neural machine translation. In *International Conference on Learning Representations (ICLR)*.

Ataman, D., Aziz, W., and Birch, A. (2020b). A latent morphology model for open-vocabulary neural machine translation. In *International Conference on Learning Representations (ICLR)*.

Baevski, A., Zhou, Y., Mohamed, A., and Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460.

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., and Bengio, Y. (2017). An actor-critic algorithm for sequence prediction.

Baroni, M., Matiasek, J., and Trost, H. (2002). Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 48–57. Association for Computational Linguistics.

Behjati, M., Fehr, F., and Henderson, J. (2023). Learning to abstract with nonparametric variational information bottleneck. In Bouamor, H., Pino, J., and Bali, K., editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1576–1586, Singapore. Association for Computational Linguistics.

Behjati, M. and Henderson, J. (2023). Inducing meaningful units from character sequences with dynamic capacity slot attention. *Transactions on Machine Learning Research*.

Belinkov, Y. and Bisk, Y. (2017). Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*.

Belinkov, Y. and Glass, J. (2019). Analysis Methods in Neural Language Processing: A Survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Bergmanis, T. and Goldwater, S. (2017). From segmentation to analyses: a probabilistic model for unsupervised morphology induction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 337–346.

Bica, I., Ilić, A., Bauer, M., Erdogan, G., Bošnjak, M., Kaplanis, C., Gritsenko, A. A., Minderer, M., Blundell, C., Pascanu, R., et al. (2024). Improving fine-grained understanding in image-text pre-training.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. (2015). Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Bugliarello, E., Sartran, L., Agrawal, A., Hendricks, L. A., and Nematzadeh, A. (2023). Measuring progress in fine-grained vision-and-language understanding. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1559–1582, Toronto, Canada. Association for Computational Linguistics.

Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., and Lerchner, A. (2019). Monet: Unsupervised scene decomposition and representation.

Can, B. and Manandhar, S. (2014). Methods and algorithms for unsupervised learning of morphology. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 177–205. Springer.

Cao, K. (2023). What is the best recipe for character-level encoder-only modelling? In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5924–5938, Toronto, Canada. Association for Computational Linguistics.

Cao, K. and Rei, M. (2016). A joint model for word embedding and word morphology. *arXiv preprint arXiv:1606.02601*.

Cao, K. and Rimell, L. (2021). You should evaluate your language model on marginal likelihood over tokenisations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2104–2114, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer.

Changpinyo, S., Sharma, P., Ding, N., and Soricut, R. (2021). Conceptual 12M: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In *CVPR*.

Chen, J. and Jin, Q. (2020). Better captioning with sequence-level exploration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10890–10899.

Chen, K., Zhang, R., Mensah, S., and Mao, Y. (2022). Contrastive learning with expectation-maximization for weakly supervised phrase grounding. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8549–8559, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Cho, K., van Merrienboer, B., Çaglar Gülçehre, Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*.

Choe, D., Al-Rfou, R., Guo, M., Lee, H., and Constant, N. (2019). Bridging the gap for tokenizer-free language models. *arXiv preprint arXiv:1908.10322*.

Chung, J., Ahn, S., and Bengio, Y. (2017a). Hierarchical multiscale recurrent neural networks. In *International Conference on Learning Representations (ICLR)*.

Chung, J., Ahn, S., and Bengio, Y. (2017b). Hierarchical multiscale recurrent neural networks. In *International Conference on Learning Representations*.

Clark, J. H., Garrette, D., Turc, I., and Wieting, J. (2022). Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91.

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619.

Conneau, A. and Kiela, D. (2018). SentEval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Conneau, A., Kruszewski, G., Lample, G., Barrault, L., and Baroni, M. (2018). What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Cotterell, R. and Schütze, H. (2018). Joint semantic synthesis and morphological analysis of the derived word. *Transactions of the Association for Computational Linguistics*, 6:33–48.

Crawford, E. and Pineau, J. (2019). Spatially invariant unsupervised object detection with convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3412–3420.

Creutz, M. (2003). Unsupervised segmentation of words using prior distributions of morph length and frequency. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 280–287. Association for Computational Linguistics.

Creutz, M. and Lagus, K. (2002). Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6*, pages 21–30. Association for Computational Linguistics.

Creutz, M. and Lagus, K. (2005). *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0*. Helsinki University of Technology Helsinki.

Creutz, M. and Lagus, K. (2007). Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(1):1–34.

Dai, Z., Lai, G., Yang, Y., and Le, Q. (2020). Funnel-transformer: Filtering out sequential redundancy for efficient language processing. *Advances in neural information processing systems*, 33:4271–4282.

Datta, S., Sikka, K., Roy, A., Ahuja, K., Parikh, D., and Divakaran, A. (2019). Align2ground: Weakly supervised phrase grounding guided by image-caption alignment. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2601–2610.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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.

Durrani, N., Dalvi, F., Sajjad, H., Belinkov, Y., and Nakov, P. (2019). One size does not fit all: Comparing nmt representations of different granularities. 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 1504–1516.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Elsayed, G., Mahendran, A., van Steenkiste, S., Greff, K., Mozer, M. C., and Kipf, T. (2022). Savi++: Towards end-to-end object-centric learning from real-world videos. *Advances in Neural Information Processing Systems*, 35:28940–28954.

Emami, P., He, P., Ranka, S., and Rangarajan, A. (2021). Efficient iterative amortized inference for learning symmetric and disentangled multi-object representations. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2970–2981. PMLR.

Engelcke, M., Kosiorek, A. R., Jones, O. P., and Posner, I. (2020). Genesis: Generative scene inference and sampling with object-centric latent representations. In *International Conference on Learning Representations (ICLR)*.

Eskander, R., Callejas, F., Nichols, E., Klavans, J., and Muresan, S. (2020). MorphAGram, evaluation and framework for unsupervised morphological segmentation. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 7112–7122, Marseille, France. European Language Resources Association.

Eskander, R., Klavans, J., and Muresan, S. (2019). Unsupervised morphological segmentation for low-resource polysynthetic languages. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 189–195, Florence, Italy. Association for Computational Linguistics.

Eskander, R., Rambow, O., and Yang, T. (2016). Extending the use of Adaptor Grammars for unsupervised morphological segmentation of unseen languages. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 900–910, Osaka, Japan. The COLING 2016 Organizing Committee.

Eslami, S., 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*.

Gowda, T. and May, J. (2020). Finding the optimal vocabulary size for neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3955–3964, Online. Association for Computational Linguistics.

Greff, K., Kaufman, R. L., Kabra, R., Watters, N., Burgess, C., Zoran, D., Matthey, L., Botvinick, M., and Lerchner, A. (2019). Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*.

Greff, K., Rasmus, A., Berglund, M., Hao, T., Valpola, H., and Schmidhuber, J. (2016). Tagger: Deep unsupervised perceptual grouping. In *Advances in Neural Information Processing Systems*, volume 29, pages 4484–4492.

Greff, K., Van Steenkiste, S., and Schmidhuber, J. (2017). Neural expectation maximization. In *Advances in Neural Information Processing Systems*.

Gupta, T., Vahdat, A., Chechik, G., Yang, X., Kautz, J., and Hoiem, D. (2020). Contrastive learning for weakly supervised phrase grounding. In *European Conference on Computer Vision*, pages 752–768. Springer.

Henderson, J. and Fehr, F. J. (2023). A VAE for transformers with nonparametric variational information bottleneck. In *The Eleventh International Conference on Learning Representations*.

Hendricks, L. A. and Nematzadeh, A. (2021). Probing image-language transformers for verb understanding. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3635–3644, Online. Association for Computational Linguistics.

Hewitt, J., Ethayarajh, K., Liang, P., and Manning, C. (2021). Conditional probing: measuring usable information beyond a baseline. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1626–1639, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*.

Hofmann, V., Schuetze, H., and Pierrehumbert, J. (2022). An embarrassingly simple method to mitigate undesirable properties of pretrained language model tokenizers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 385–393, Dublin, Ireland. Association for Computational Linguistics.

Hsu, W.-N., Bolte, B., Tsai, Y.-H. H., Lakhotia, K., Salakhutdinov, R., and Mohamed, A. (2021). Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460.

Jang, E., Gu, S., and Poole, B. (2017). Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations (ICLR)*.

Jegou, H., Douze, M., and Schmid, C. (2010). Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128.

Kamath, A., Hessel, J., and Chang, K.-W. (2023). Text encoders bottleneck compositionality in contrastive vision-language models. In Bouamor, H., Pino, J., and Bali, K., editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4933–4944, Singapore. Association for Computational Linguistics.

Kamath, A., Singh, M., LeCun, Y., Synnaeve, G., Misra, I., and Carion, N. (2021). Mdetr-modulated detection for end-to-end multi-modal understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1780–1790.

Kann, K., Cotterell, R., and Schütze, H. (2016). Neural morphological analysis: Encoding-decoding canonical segments. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 961–967.

Kaushal, A. and Mahowald, K. (2022). What do tokens know about their characters and how do they know it? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2487–2507, Seattle, United States. Association for Computational Linguistics.

Kawakami, K., Dyer, C., and Blunsom, P. (2017). Learning to create and reuse words in open-vocabulary neural language modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1492–1502, Vancouver, Canada. Association for Computational Linguistics.

Kawakami, K., Dyer, C., and Blunsom, P. (2019). Learning to discover, ground and use words with segmental neural language models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6429–6441, Florence, Italy. Association for Computational Linguistics.

Kim, D., Kim, N., Lan, C., and Kwak, S. (2023). Shatter and gather: Learning referring image segmentation with text supervision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15547–15557.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y., editors, *2nd International Conference on Learning Representations, ICLR 2014, Conference Track Proceedings*, Banff, AB, Canada.

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 *International Conference on Learning Representations*.

Kudo, T. (2018). Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Kudo, T. and Richardson, J. (2018a). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing.

Kudo, T. and Richardson, J. (2018b). SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

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, J., He, X., Wei, L., Qian, L., Zhu, L., Xie, L., Zhuang, Y., Tian, Q., and Tang, S. (2022). Fine-grained semantically aligned vision-language pre-training. *Advances in neural information processing systems*, 35:7290–7303.

Lin, Z., Wu, Y.-F., 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 *International Conference on Learning Representations (ICLR)*.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. (2020). On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*.

Liu, S., Huang, S., Li, F., Zhang, H., Liang, Y., Su, H., Zhu, J., and Zhang, L. (2023). Dq-detr: Dual query detection transformer for phrase extraction and grounding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 1728–1736.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. (2020). Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 33:11525–11538.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. *International Conference on Learning Representations (ICLR)*.

Louizos, C., Welling, M., and Kingma, D. P. (2018). Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations (ICLR)*.

Luo, J., Narasimhan, K., and Barzilay, R. (2017). Unsupervised learning of morphological forests. *Transactions of the Association for Computational Linguistics*, 5:353–364.

Luyu Gao, Yunyi Zhang, J. H. J. C. (2021). Scaling deep contrastive learning batch size under memory limited setup. In *Proceedings of the 6th Workshop on Representation Learning for NLP*.

Maddison, C. J., Mnih, A., and Teh, Y. W. (2017). The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations (ICLR)*.

Mai, F. and Henderson, J. (2022). Bag-of-vectors autoencoders for unsupervised conditional text generation. 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*, pages 468–488.

Mailhot, H., Wilson, M. A., Macoir, J., Deacon, S. H., and Sánchez-Gutiérrez, C. H. (2020). MorphoLex-FR: A derivational morphological database for 38,840 French words. *Behavior Research Methods*, 52(3):1008–1025.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. (2016). Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. (2017). Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings*, Toulon, France. OpenReview.net.

Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Morris, J., Lifland, E., Yoo, J. Y., Grigsby, J., Jin, D., and Qi, Y. (2020). Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126.

Mukhoti, J., Lin, T.-Y., Poursaeed, O., Wang, R., Shah, A., Torr, P. H., and Lim, S.-N. (2023). Open vocabulary semantic segmentation with patch aligned contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19413–19423.

Narasimhan, K., Barzilay, R., and Jaakkola, T. (2015). An unsupervised method for uncovering morphological chains. *Transactions of the Association for Computational Linguistics*, 3:157–167.

Nawrot, P., Chorowski, J., Lancucki, A., and Ponti, E. M. (2023). Efficient transformers with dynamic token pooling. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6403–6417, Toronto, Canada. Association for Computational Linguistics.

Nawrot, P., Tworkowski, S., Tyrolski, M., Kaiser, L., Wu, Y., Szegedy, C., and Michalewski, H. (2022). Hierarchical transformers are more efficient language models. In Carpuat, M., de Marneffe, M.-C., and Meza Ruiz, I. V., editors, *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1559–1571, Seattle, United States. Association for Computational Linguistics.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018a). Representation learning with contrastive predictive coding. In *arXiv preprint arXiv:1807.03748*.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018b). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Parisotto, E., Song, H. F., Rae, J. W., Pascanu, R., Gulcehre, C., Jayakumar, S. M., Jaderberg, M., Kaufman, R. L., Clark, A., Noury, S., et al. (2019). Stabilizing transformers for reinforcement learning. *arXiv preprint arXiv:1910.06764*.

Patel, Y., Xie, Y., Zhu, Y., Appalaraju, S., and Manmatha, R. (2023). Simcon loss with multiple views for text supervised semantic segmentation. *arXiv preprint arXiv:2302.03432*.

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.

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.

Pimentel, T., Valvoda, J., Stoehr, N., and Cotterell, R. (2022). Attentional probe: Estimating a module's functional potential. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11459–11472, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Plummer, B. A., Wang, L., Cervantes, C. M., Caicedo, J. C., Hockenmaier, J., and Lazebnik, S. (2015). Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the IEEE international conference on computer vision*, pages 2641–2649.

Poon, H., Cherry, C., and Toutanova, K. (2009). Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 209–217, Boulder, Colorado. Association for Computational Linguistics.

Provilkov, I., Emelianenko, D., and Voita, E. (2020). BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online. Association for Computational Linguistics.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2016). Sequence level training with recurrent neural networks. In *International Conference on Learning Representations (ICLR)*.

Sajjadi, M. S., Duckworth, D., Mahendran, A., van Steenkiste, S., Pavetic, F., Lucic, M., Guibas, L. J., Greff, K., and Kipf, T. (2022). Object scene representation transformer. *Advances in Neural Information Processing Systems*, 35:9512–9524.

Schuster, M. and Nakajima, K. (2012). Japanese and korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5149–5152. IEEE.

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 *The Eleventh International Conference on Learning Representations*.

Sennrich, R., Haddow, B., and Birch, A. (2015). Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

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, P., Ding, N., Goodman, S., and Soricut, R. (2018). Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565.

Shekhar, R., Pezzelle, S., Klimovich, Y., Herbelot, A., Nabi, M., Sangineto, E., and Bernardi, R. (2017). FOIL it! find one mismatch between image and language caption. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 255–265, Vancouver, Canada. Association for Computational Linguistics.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905.

Singh, G., Deng, F., and Ahn, S. (2022). Illiterate dall-e learns to compose. In *International Conference on Learning Representations*.

Singh, G., Kim, Y., and Ahn, S. (2023a). Neural systematic binder. In *The Eleventh International Conference on Learning Representations*.

Singh, G., Kim, Y., and Ahn, S. (2023b). Neural systematic binder. In *The Eleventh International Conference on Learning Representations*.

Sirts, K. and Goldwater, S. (2013). Minimally-supervised morphological segmentation using Adaptor Grammars. *Transactions of the Association for Computational Linguistics*, 1:255–266.

Soricut, R. and Och, F. (2015a). Unsupervised morphology induction using word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1627–1637, Denver, Colorado. Association for Computational Linguistics.

Soricut, R. and Och, F. J. (2015b). Unsupervised morphology induction using word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1627–1637.

Sun, L., Luisier, F., Batmanghelich, K., Florencio, D., and Zhang, C. (2023). From characters to words: Hierarchical pre-trained language model for open-vocabulary language understanding. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3605–3620, Toronto, Canada. Association for Computational Linguistics.

Sun, Z. and Deng, Z.-H. (2018). Unsupervised neural word segmentation for Chinese via segmental language modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4915–4920, Brussels, Belgium. Association for Computational Linguistics.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Sánchez-Gutiérrez, C. H., Mailhot, H., Deacon, S. H., and Wilson, M. A. (2018). MorphoLex: A derivational morphological database for 70,000 English words. *Behavior Research Methods*, 50(4):1568–1580.

Tay, Y., Tran, V. Q., Ruder, S., Gupta, J., Chung, H. W., Bahri, D., Qin, Z., Baumgartner, S., Yu, C., and Metzler, D. (2022). Charformer: Fast character transformers via gradient-based subword tokenization. In *International Conference on Learning Representations*.

Thomee, B., Shamma, D. A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., and Li, L.-J. (2016). Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73.

Tjandra, A., Sakti, S., and Nakamura, S. (2020). Transformer vq-vae for unsupervised unit discovery and speech synthesis: Zerospeech 2020 challenge. *arXiv preprint arXiv:2005.11676*.

Üstün, A. and Can, B. (2016). Unsupervised morphological segmentation using neural word embeddings. In Král, P. and Martín-Vide, C., editors, *Statistical Language and Speech Processing*, pages 43–53, Cham. Springer International Publishing.

Van Den Oord, A., Vinyals, O., et al. (2017). Neural discrete representation learning. *Advances in neural information processing systems*, 30.

Van Steenkiste, S., Chang, M., Greff, K., and Schmidhuber, J. (2018). Relational neural expectation maximization: Unsupervised discovery of objects and their interactions.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017a). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017b). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Virpioja, Sami ; Smit, P. . G. S.-A. . K. M. (2013). Morfessor 2.0: Python implementation and extensions for morfessor baseline. In *Aalto University publication series*. Department of Signal Processing and Acoustics, Aalto University.

Wang, F., Zhou, Y., Wang, S., Vardhanabhuti, V., and Yu, L. (2022). Multi-granularity cross-modal alignment for generalized medical visual representation learning. *Advances in Neural Information Processing Systems*, 35:33536–33549.

Wang, Q., Tan, H., Shen, S., Mahoney, M., and Yao, Z. (2020). MAF: Multimodal alignment framework for weakly-supervised phrase grounding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2030–2038, Online. Association for Computational Linguistics.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Wu, Y.-F., Greff, K., Elsayed, G. F., Mozer, M. C., Kipf, T., and van Steenkiste, S. (2023). Inverted-attention transformers can learn object representations: Insights from slot attention. In *Causal Representation Learning Workshop at NeurIPS 2023*.

Wu, Y.-F., Lee, M., and Ahn, S. (2024). Structured world modeling via semantic vector quantization. *arXiv preprint arXiv:2402.01203*.

Xu, H., Kodner, J., Marcus, M., and Yang, C. (2020). Modeling morphological typology for unsupervised learning of language morphology. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6672–6681, Online. Association for Computational Linguistics.

Xu, H., Marcus, M., Yang, C., and Ungar, L. (2018a). Unsupervised morphology learning with statistical paradigms. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 44–54, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Xu, H., Marcus, M., Yang, C., and Ungar, L. (2018b). Unsupervised morphology learning with statistical paradigms. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 44–54.

Xu, J., De Mello, S., Liu, S., Byeon, W., Breuel, T., Kautz, J., and Wang, X. (2022). Groupvit: Semantic segmentation emerges from text supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18134–18144.

Xue, L., Barua, A., Constant, N., Al-Rfou, R., Narang, S., Kale, M., Roberts, A., and Raffel, C. (2022). ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

Yao, L., Huang, R., Hou, L., Lu, G., Niu, M., Xu, H., Liang, X., Li, Z., Jiang, X., and Xu, C. (2022). Filip: Fine-grained interactive language-image pre-training. In *The Eleventh International Conference on Learning Representations*.

Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Yuksekgonul, M., Bianchi, F., Kalluri, P., Jurafsky, D., and Zou, J. (2022). When and why vision-language models behave like bags-of-words, and what to do about it? In *The Eleventh International Conference on Learning Representations*.

Žabokrtský, Z., Bafna, N., Bodnár, J., Kyjánek, L., Svoboda, E., Ševčíková, M., Vidra, J., Angle, S., Ansari, E., Arkhangelskiy, T., Batsuren, K., Bella, G., Bertinetto, P. M., Bonami, O., Celata, C., Daniel, M., Fedorenko, A., Filko, M., Giunchiglia, F., Haghdoost, H., Hathout, N.,

Khomchenkova, I., Khurshudyan, V., Levonian, D., Litta, E., Medvedeva, M., Muralikrishna, S. N., Namer, F., Nikravesh, M., Padó, S., Passarotti, M., Plungian, V., Polyakov, A., Potapov, M., Pruthwik, M., Rao B, A., Rubakov, S., Samar, H., Sharma, D. M., Šnajder, J., Šojat, K., Štefanec, V., Talamo, L., Tribout, D., Vodolazsky, D., Vydrin, A., Zakirova, A., and Zeller, B. (2022). Universal segmentations 1.0 (UniSegments 1.0). LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Zeng, Y., Zhang, X., and Li, H. (2022a). Multi-grained vision language pre-training: Aligning texts with visual concepts. In *Proceedings of the 39th International Conference on Machine Learning*.

Zeng, Y., Zhang, X., and Li, H. (2022b). Multi-grained vision language pre-training: Aligning texts with visual concepts. *International Conference on Machine Learning*.

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.

Lausanne, Switzerland
+41 (78) 707 87 47
melikabehjati@gmail.com
Swiss Permit B

Melika Behjati

Experiences

2019 – 2024 **Research Assistant**, *Natural Language Understanding Group*, Idiap institute, Switzerland
Supervisor: Dr. James Henderson
Research on discovering meaningful units from text sequences in an unsupervised manner.
Hands-on experience with **Transformers**, **Autoencoders**, **Language Models**, **Vision Language Models**, and different **interpretability** methods.
Familiar with **Tokenization**, **Contrastive Learning**, **Generative Models**, **Multimodality**, **Reinforcement Learning**, **Unsupervised Object Discovery** and **Information Retrieval**.

2017 – 2019 **Graduate Researcher**, *Machine Learning Laboratory*, Sharif University of Technology, Iran
Supervisor: Dr. Mahdieh Soleymani Baghshah
Develop a method for a universal **adversarial attack** over **text classifiers** based on **word embeddings** and **language models.**

2018 **Research Intern**, *LTS4 Laboratory*, EPFL, Switzerland
Supervisor: Dr. Pascal Frossard
Research on adversarial attacks in the text domain.

2017 **BSc Project**, *Machine Learning Laboratory*, Sharif University of Technology, Iran
Supervisor: Dr. Mahdieh Soleymani Baghshah
Analysis of Persian poems based on **semantic embeddings**, **clustering** and **topic modeling techniques**.

2016 **Front-end Development Internship**, *Hasin co.*, Iran
Front-end development of a product website using **AngularJS**.

Education

2019 – 2024 **Ph.D. in Computer Science**, *Ecole Polytechnique Fédérale de Lausanne (EPFL)*, Lausanne, Switzerland
Supervisor: Dr. James Henderson, Dr. Alexandre Alahi
Thesis: Discovering Meaningful Units from Text Sequences

2017 – 2019 **M.Sc. in Computer Engineering**, *Sharif University of Technology*, Tehran, Iran
Major: Artificial Intelligence and Robotics
Supervisor: Dr. Mahdieh Soleymani Baghshah
Thesis: Adversarial Robustness of Deep Neural Networks in Text Domain

2013 – 2017 **BSc in Computer Engineering**, *Sharif University of Technology*, Tehran, Iran
Major: Software Engineering
Thesis: Persian Poems Analysis based on Semantic Embedding

Technical Skills

Technologies: Machine Learning, Artificial Intelligence, Deep Learning, Natural Language Processing, Computer Vision

Programming Languages Python, Java, C/C++, , Matlab, CUDA

ML	Pytorch, Tensorflow, Numpy, Transformers, Weights&Biases, NLTK, Scikit-learn, SQLite
Software Development	git, CI/CD, Agile Software Development (Scrum), Slurm
Web Development	Django, AngularJS, JavaScript, HTML, CSS

Awards and Honors

- Awarded Ph.D. Fellowship from the School of Computer and Communication Sciences, *Ecole Polytechnique Fédérale de Lausanne*, Switzerland, 2019
- Winner of National Elites Foundation Scholarship, Iran, 2015-2019
- Ranked 4 in terms of cumulative GPA among M.Sc. students of Artificial Intelligence among 2017 beginners, *Sharif University of Technology*, Iran, 2018
- Awarded MSc Fellowship for Exceptional Talents in Computer Engineering (Artificial Intelligence), *Sharif University of Technology*, Iran, 2017
- Ranked 5 in terms of cumulative GPA among students of Computer Engineering among 2013 beginners, *Sharif University of Technology*, Iran, 2017

Publications

2024 Abdollahi, Ali, Mahdi Ghaznavi, Mohammad Reza Karimi Nejad, Arash Mari Oriyad, Reza Abbasi, Ali Salesi, Melika Behjati, Mohammad Hossein Rohban, and Mahdieh Soleymani Baghshah (2024). "GABInsight: Exploring Gender-Activity Binding Bias in Vision-Language Models". In: *ECAI*.

2024 Behjati, Melika and James Henderson (2024). "Discovering Meaningful Units with Visually Grounded Semantics from Image Captions". In: *Under review*.

2023 Behjati, Melika, Fabio Fehr, and James Henderson (2023). "Learning to Abstract with Nonparametric Variational Information Bottleneck". In: *Findings of EMNLP*.

2023 Behjati, Melika and James Henderson (2023). "Inducing Meaningful Units from Character Sequences with Dynamic Capacity Slot Attention". In: *TMLR*.

2019 Behjati, Melika, Seyed-Mohsen Moosavi-Dezfooli, Mahdieh Soleymani Baghshah, and Pascal Frossard (2019). "Universal adversarial attacks on text classifiers". In: *ICASSP*.

Languages

Persian	Native
English	Fluent
French	Intermediate

Interests

Arts	Calligraphy, Painting
Sport	Hiking, Yoga