Philosophische Fakultät

CONTEXTUALISATION OF AUTOMATIC SPEECH RECOGNITION AND RELATED APPLICATIONS

Thesis
presented to the Faculty of Arts and Social Sciences
of the University of Zurich
for the degree of Doctor of Philosophy

by
Iuliia Thorbecke

Supervisory Committee:
PD Dr. Tanja Samardžić (Principal Supervisor)
Prof. Dr. Rico Sennrich
Prof. Dr. Petr Motlíček

Accepted in the fall semester 2025
on the recommendation of the Doctoral Committee composed of
PD Dr. Tanja Samardžić
Prof. Dr. Rico Sennrich
Prof. Dr. Petr Motlíček
PD Dr. Ralf Schlüter

Zurich, 2026

Abstract

Automatic Speech Recognition (ASR) systems have made remarkable progress in recent years, largely driven by advances in deep learning and the availability of large-scale training datasets. These improvements have led to significant reductions in Word Error Rate (WER), particularly in general-purpose, in-domain tasks. However, real-world applications often demand more than overall accuracy—they require precise recognition of specific, context-dependent information such as domain-specific terms, critical word sequences, or named entities (NEs). These are the very areas where high-performing ASR systems tend to underperform, especially in the absence of explicit contextual support.

This thesis addresses the problem of contextual ASR, with a specific focus on text-only contextual data and scenarios where the base ASR model cannot be retrained. We aim to improve both general ASR performance and the accurate recognition of key target entities, namely (1) n-grams that are more likely within a specific context and (2) NEs, through context-aware biasing strategies. These strategies are evaluated across different ASR paradigms, including hybrid and End-to-End models, and in varied configurations (e.g., offline/online ASR, semi-supervised learning, streaming, and domain adaptation).

A central contribution of this work is a comprehensive evaluation and analysis of contextual integration methods, offering practical insights into when and how different techniques can be effectively employed. For hybrid (pipeline) ASR, we investigate three context injection methods: lattice-level biasing, decoding graph modification, and grammar FST augmentation with context-specific entities. We demonstrate the effectiveness of these methods in improving recognition not only for frequent terms but also for rare and out-of-vocabulary (OOV) words. Furthermore, we introduce a GPU-efficient algorithm for real-time dynamic contextual biasing. This method adjusts decoding graph arc weights using input keywords and n-grams, without generating lattices, enabling rapid adaptation and closer integration of inference and decoding processes.

In the domain of End-to-End ASR, we propose an efficient approach for integrating word-level n-gram language models (LMs) using the Aho-Corasick (AC) algorithm. This enables the fusion of keyword biasing and n-gram LM adaptation into a single, unified context trie. Compared to standard shallow fusion (SF) with neural network LMs, our method achieves faster decoding with competitive WER and real-time factor (RTF) performance. Evaluations on four languages and datasets confirm improvements in both NE recognition and general ASR accuracy, including for OOV terms. We further show that SF remains effective even when biasing entities are generated automatically or when distractors are present, and we benchmark multiple

SF strategies across languages and data domains. In addition, we explore dynamic contextualisation via SF as a complement to static encoder conditioning, and present early results on domain adaptation using text-only and pseudo-audio data.

We extend the investigation to ASR-related tasks, including Named Entity Recognition (NER) from speech and semi-supervised learning (SSL). We develop a two-step biasing method combining ASR and NLP modules: first boosting NER via FST biasing, then correcting erroneous predictions with NLP post-processing. This approach is shown to work effectively in both cascaded ASR-NER pipelines and End-to-End multitask models. In SSL, we propose a method to incorporate contextual knowledge into iterative model training. Pseudo-labeled lattices from untranscribed data are rescored using context-aware techniques, improving key information recognition and outperforming traditional SSL methods in unseen domains.

To address the challenge of training streaming ASR models with minimal supervision, we introduce a framework that uses noisy pseudo-labels from foundational models. Transformer-Transducer architectures trained from scratch on this data show promising results, particularly when combined with shallow fusion using n-gram LMs and extracted NEs.

Finally, a significant portion of this research is conducted using ATC data, where radar surveillance information serves as context. Our contributions in this domain demonstrate the potential of ASR contextualisation to enhance communication reliability and efficiency in high-stakes environments. We consider our work to be a meaningful step toward the development of ASR-assisted ATC systems capable of supporting domain-critical tasks and mitigating the risks associated with miscommunication.

Zusammenfassung

Systeme zur automatischen Spracherkennung (ASR) haben in den letzten Jahren bemerkenswerte Fortschritte gemacht, die größtenteils auf Fortschritte im Deep Learning und die Verfügbarkeit umfangreicher Trainingsdatensätze zurückzuführen sind. Diese Verbesserungen haben zu deutlichen Reduktionen der Wortfehlerrate (Word Error Rate, WER) geführt, insbesondere bei allgemeinen, domänenspezifischen Aufgaben. In der Praxis erfordern Anwendungen jedoch oft mehr als nur allgemeine Genauigkeit – sie verlangen eine präzise Erkennung spezifischer, kontextabhängiger Informationen wie domänenspezifischer Begriffe, kritischer Wortsequenzen oder Eigennamen (NEs). Gerade in diesen Bereichen neigen leistungsfähige ASR-Systeme dazu, unzureichend abzuschneiden, insbesondere wenn keine explizite kontextuelle Unterstützung vorhanden ist.

Diese Arbeit beschäftigt sich mit dem Problem der kontextuellen Spracherkennung, wobei der Fokus speziell auf rein textbasierten Kontextdaten liegt und Szenarien betrachtet werden, in denen das zugrundeliegende ASR-Modell nicht neu trainiert werden kann. Das Ziel ist es, sowohl die allgemeine Leistung von ASR-Systemen als auch die genaue Erkennung bestimmter Zielentitäten zu verbessern, insbesondere (1) n-Gramme, die in einem bestimmten Kontext wahrscheinlicher sind, und (2) Eigennamen (NEs) durch kontextbewusste Biasing-Strategien. Diese Strategien werden in verschiedenen ASR-Paradigmen evaluiert, darunter hybride und End-to-End-Modelle, sowie in unterschiedlichen Konfigurationen (z.B. Offline-/Online-ASR, halbüberwachtes Lernen, Streaming und Domänenanpassung).

Die Arbeit präsentiert eine umfassende Evaluierung und Analyse von Methoden zur Integration von Kontext und liefert praktische Einblicke, wann und wie verschiedene Techniken effektiv eingesetzt werden können. Für hybride (Pipeline-)ASR-Systeme untersuchen wir drei Methoden zur Kontexteinbindung: Biasing auf Lattice-Ebene, Modifikation des Decodiergraphen und Erweiterung der Grammatik-FST mit kontextspezifische Entitäten. Wir zeigen die Wirksamkeit dieser Methoden bei der Verbesserung der Erkennung nicht nur häufiger Begriffe, sondern auch seltener und nicht im Vokabular enthaltener (Out-of-Vocabulary, OOV) Wörter. Darüber hinaus stellen wir einen GPU-effizienten Algorithmus für die Echtzeit-Adaption kontextueller Biasing vor. Diese Methode passt die Kantengewichte des Decodiergraphen anhand von Eingabe-Keywords und n-Grammen an, ohne Lattices zu generieren, was eine schnelle Adaption und eine engere Integration von Inferenz- und Decodierprozessen ermöglicht.

Im Bereich der End-to-End-ASR schlagen wir einen effizienten Ansatz zur Integration wortbasierter n-Gramm-Sprachmodelle (LMs) unter Verwendung des Aho-Corasick-(AC)-Algorithmus vor. Dies ermöglicht die Fusion von Keyword-Biasing

und n-Gramm-LM-Anpassung in einen einzigen, vereinheitlichten Kontext-Trie. Im Vergleich zur klassischen Shallow-Fusion (SF) mit neuronalen Netzwerk-LMs erreicht unsere Methode eine schnellere Decodierung bei konkurrenzfähiger WER und Echtzeitfaktor (RTF). Evaluierungen auf vier Sprachen und Datensätzen bestätigen Verbesserungen sowohl bei der Erkennung Eigennamen als auch bei der allgemeinen ASR-Genauigkeit, einschließlich OOV-Begriffen. Wir zeigen außerdem, dass SF auch dann effektiv bleibt, wenn die zu biasenden Entitäten automatisch generiert oder Störreize vorhanden sind, und stellen einen Vergleich mehrerer SF-Strategien über Sprachen und Datendomänen hinweg an. Zusätzlich untersuchen wir dynamische Kontextualisierung mittels SF als Ergänzung zur statischen Encoder-Bedingung und präsentieren erste Ergebnisse zur Domänenanpassung unter Verwendung rein textbasierter und Pseudo-Audiodaten.

Die Untersuchung wird auf ASR-nahe Aufgaben erweitert, darunter die Erkennung Eigennamen (NER) aus Sprache und halbüberwachtes Lernen (SSL). Wir entwickeln eine zweistufige Biasing-Methode, die ASR- und NLP-Module kombiniert: Zuerst wird die NER durch FST-Biasing verstärkt, anschließend werden fehlerhafte Vorhersagen durch NLP-Nachbearbeitung korrigiert. Diese Methode erweist sich als effektiv sowohl in kaskadierten ASR-NER-Pipelines als auch in End-to-End-Multitask-Modellen. Im Bereich des SSL schlagen wir eine Methode vor, kontextuelles Wissen in das iterative Modelltraining einzubinden. Pseudobeschriftete Lattices aus untranskribierten Daten werden mit kontextbewussten Techniken neu bewertet, wodurch die Erkennung wichtiger Informationen verbessert wird und die Leistung in unbekannten Domänen über traditionelle SSL-Methoden hinausgeht.

Um die Herausforderung des Trainings von Streaming-ASR-Modellen mit minimalem Supervisionsaufwand zu adressieren, stellen wir einen Rahmen vor, der verrauschte Pseudolabels grundlegender Modelle nutzt. Transformer-Transducer Architekturen, die von Grund auf mit diesen Daten trainiert werden, zeigen vielversprechende Ergebnisse, insbesondere in Kombination mit Shallow Fusion unter Verwendung von n-Gramm-LMs und extrahierten NEs.

Ein wesentlicher Teil dieser Forschung wird mit Daten aus dem Bereich der Flugsicherung (ATC) durchgeführt, bei denen Radarüberwachungsinformationen als Kontext dienen. Unsere Beiträge in diesem Bereich zeigen das Potenzial der kontextuellen ASR auf, die Zuverlässigkeit und Effizienz der Kommunikation in sicherheitskritischen Umgebungen zu verbessern. Wir betrachten unsere Arbeit als einen bedeutenden Schritt hin zu ASR-unterstützten ATC-Systemen, die domänenspezifische kritische Aufgaben unterstützen und die Risiken durch Fehlkommunikation verringern können.

Acknowledgements

This thesis would not have been possible without many colleagues and friends.

First and foremost, I would like to thank my UZH supervisors, Tanja Samardžić and Rico Sennrich. My love for speech recognition roots deeply in the past, and thanks to Tanja, I did a big step towards ASR research already during my master's thesis. Regular discussions with both Tanja and Rico helped me to find my way through the thorny journey of a PhD, and their belief in me did not let me give up. I thank them for their expertise, motivating guidance, detailed feedback, and support.

I would like to thank my IDIAP supervisor, Petr Motliček, and all the colleagues from the IDIAP Speech & Audio Processing group, with special thanks to Srikanth Madikeri, Pablo Zuluaga-Gomez, Rudolf Braun, Esaú Villatoro-Tello, Mrinmoy Bhattacharjee, Driss Khalil, Shashi Kumar, Amrutha Prasad, Seyyed Saeed Sarfjoo, and Sergio Burdisso. At the IDIAP Research Institute, I had the opportunity to participate in interesting and challenging projects that allowed me to work on real-world use cases and significantly shaped my PhD research. I would like to thank Hartmut Helmke and Matthias Kleinert from DLR, Hanno Wiese from Fraport AG, Karthik Pandia, Aravind Ganapathiraju, Alexei V. Ivanov, Kadri Hacioğlu, and Andreas Stolcke from Uniphore. I am grateful for the opportunity to collaborate with speech recognition experts, apply my knowledge, and further strengthen my expertise.

I would also like to thank my external reviewer, Ralf Schlüter, for his valuable feedback and comments, which helped to improve the final version of this thesis. I thank Volker Dellwo and Anastassia Shaitarova for being the Chair and Minute Taker at the Thesis Defence.

Finally, I thank my family: my parents who were always close despite the distance and all the circumstances, my in-laws who helped a lot babysitting my son, my brother who is my greatest support and inspiration ever, my husband who has been by my side since the very beginning and who did his utmost to help me maintain a healthy work-life balance, and my little son Erik who joined us at the end of the project but had the greatest impact on everything.

Contents

List of Figures

List of Tables

List of Acronyms

AC	Aho-Corasick
AM	Acoustic Model
AED	Attention-Based Encoder Decoder
ATC	Air Traffic Communication
ASR	Automatic Speech Recognition
BPE	Byte-Pair Encoding
CTC	Connectionist Temporal Classification
CNN	Convolutional Neural Network
Conformer	Convolution-Augmented Transformer
dB	Decibel
DNN	Deep Neural Networks
E2E	End-To-End
ELM	External Language Model
ENDP	Endpointing
FST	Finite State Transducer
ICAO	International Civil Aviation Organization
ILM	Internal Language Model
ILMA	Internal Language Model Adaptation
GMM	Gaussian Mixture Model
HMM	Hidden Markov model
KD	Knowledge Distillation
LF-MMI	Lattice-Free Maximum Mutual Information
LM	Language Model
ML	Machine Learning
MT	Machine Translation
MFCC	Mel-frequency Cepstral Coefficients
MHA	Multi-Head Attention
NE	Named Entity
NER	Named Entity Recognition
NLP	Natural Language Processing
NLU	Natural Language Understanding

OOV	Out-of-Vocabulary
PL	Pseudo-Label
RTX	Real-Time Factor
RNN	Recurrent Neural Network
RNN-T	RNN-Transducer
SF	Shallow Fusion
SLU	Spoken Language Understanding
SOTA	State-Of-The-Art
SSL	Self-Supervised Learning
SNR	Signal-To-Noise
TTS	Text-to-Speech
TT	Transformer-Transducer
TDNN	Time Delay Neural Network
TDNNF	Factorized TDNN
VHF	Very-High Frequency
WER	Word Error Rate
WFST	Weighted Finite State Transducer

To my son Erik

1. Introduction

Any human conversation occurs in a context, which can be linguistic, as well as extralinguistic. It can be the context of the situation, including previous and upcoming events, the topic or aim of the conversation, or the common background of the speakers. Such context information is a great assistance to reaching a mutual understanding — the goal of the conversation. Context plays an important role in the correct "reconstruction" of the intended message, especially in a situation of a noisy environment, phonetic reduction or ambiguity. For example, two phrases can sound acoustically identical but have very different meanings:

These /dʒiːnz/ *are very strong.*
These ***genes*** *are very strong.*
These ***jeans*** *are very strong.*

Depending on the situation, the word /dʒiːnz/ can be interpreted either as "sequence of nucleotides in DNA" or "fabric". Without knowing the context, it is impossible to be sure about the phrase's meaning. However, in the course of a conversation, ambiguity is usually easily resolved. Words related to the topic of conversation tend to become semantically more probable. In other words, exposure to a specific context can influence how an utterance is interpreted. In psychology and psycholinguistics, this phenomenon is closely related to *semantic priming*, where the perception or recognition of a stimulus is facilitated by prior exposure to a related stimulus (Meyer and Schvaneveldt, 1971; Flores d'Arcais et al., 1985; Shelton and Martin, 1992; McNamara, 2005).

Semantic priming means that thinking about one concept activates related memories and makes people more likely and faster to think about conceptually related words (see Fig. 1.1: the choice between *SOUP* and *SOAP* depends on the stimuli presented either from the "food" or from the "bathing" domain). Priming bias can be reached by stimuli of different types, i.e. linguistic (semantic, acoustic, etc.), visual (shape, colour, size, etc.), and cultural. In the above example, the preceding context, including such words as *biology, molecular, DNA* would activate the first interpretation of the phrase: *"These* ***genes*** *are very strong"*. At the same time,

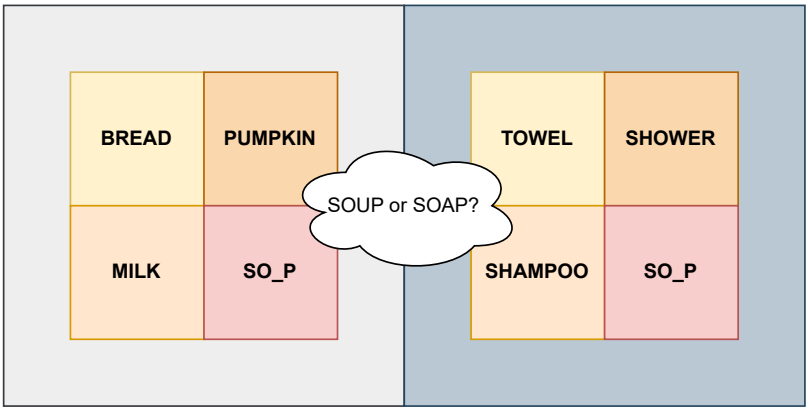

Figure 1.1.: An example of semantic priming for the orthographically and acoustically close words **SOUP** and **SOAP**.

preceding words like *fashion, clothes, pants* would make a listener choose first the second interpretation of the phrase: *"These **jeans** are very strong"*. Thus, priming effects and exposure to a particular stimulus can bias the listener's recognition preferences.

Evidence from human perception inspires the introduction of contextualisation to automatic speech recognition (ASR) as well. In a similar way, contextual information can pre-activate certain linguistic expectations, aiding comprehension and recognition in ASR systems. In a human conversation, the context typically goes far beyond just speech, and in ASR, it is by design limited to speech. The linguistic context includes *semantic context* that refers to the meaning of words in a sentence, and *syntactic context* that focuses on grammatical structure and rules governing the arrangement of those words. The size of the semantic context considered in ASR varies from model to model and depends on how the model is trained. When considering models without specialised contextual training, the syntactic context is modelled by the language model (LM) or the decoder component. It can be modelled on the level of utterance (by the attention mechanism) or on the level of an n-gram size (by an n-gram LM or by a transducer decoder). Additionally, it can be adjusted by an external n-gram or neural network LM by rescoring or fusion methods.

Despite the different natures of speech perception in people and in ASR, ASR also can benefit from the context, yet it is done through different mechanisms. To bias the model's predictions toward desired recognition, probabilities of the model's hypotheses should be adjusted to favour target words. The probabilities can be modified by further training the model on the target domain data or by introducing target words and entities to the model during inference. The modifications can be either intro-

duced "on-the-fly" during inference adjusting the scores of final predictions only, or before the inference when the context information is integrated stationary into the model's components (for example, modifying an LM). In the first case, the model remains unmodified, which makes it well-suited for flexible and dynamic adaptation with low computational cost, even in rapidly changing contexts. In the second case, the model (or some of its components) is modified, which usually does not allow fast integration because the model needs to be trained again for each new domain, which is slow and expensive. At the same time, model adaptation often leads to better context integration and, thus, better performance. Model contextualisation can be achieved either by incorporating dedicated context-aware components directly within the model architecture or by integrating external contextual modules that operate independently alongside the primary model.

In this thesis, we tackle the problem of ASR contextualisation, considering different ASR models, application challenges, and various real-life use cases.

1.1. Goal and Motivation

The **goal** of the thesis is to investigate and improve methods for contextual integration when only text context data is available and without retraining the base ASR model. More specifically, the **objective** of the thesis is to improve the overall ASR performance and the recognition of target key entities, i.e. (1) word sequences (n-gram) which are more probable in a given context, (2) named entities (NE).

Contextualisation with text data alone is addressed because text data is easier to collect compared to the parallel audio-text data and can even be available directly from the final users. Text data needs considerably less memory and can be used in streaming applications. We focus on the contextualisation methods that do not require retraining of the base ASR model. This principle introduces flexible contextualisation, which allows easy customisation of the system, fast inter-domain switching without the model change, and privacy preservation when biasing is applied locally on the user's side.

These flexible contextualisation strategies are particularly valuable given the remarkable progress ASR models have achieved. In recent years, ASR models have seen significant advancements in word error rate performance, primarily due to deep learning and large-scale data-driven approaches. Foundational speech models have shown robust performance, especially in well-defined benchmarks and databases. A crucial aspect of their effectiveness is their *generalisation ability* — the capacity

to perform well on unseen data that differs from the training distribution. Good generalisation is achieved when a model learns the general distribution of the training data well and without learning irrelevant data points' specific details that can cause overfitting. Several machine learning (ML) techniques, such as regularisation, dropout, and early stopping, help maximise generalisation during training.

As we aim for a model to perform well on the unseen test data, reaching a strong generalisation ability is typically the main goal when training a new model. Recent large ASR and language models, such as wav2vec 2.0 (Baevski et al., 2020), Whisper (Radford et al., 2023), BERT (Devlin et al., 2019), Llama (Touvron et al., 2023), GPT (Radford et al., 2019), are trained on unprecedentedly large amounts of data, which is the key to their impressive performance, because large data brings diversity necessary to achieve great generalisation ability. Despite the strength of large models and state-of-the-art (SOTA) results on most benchmarks (Rekesh et al., 2023; Radford et al., 2023; Abouelenin et al., 2025), their performance still has some limitations. The scope for improvement is particularly visible when evaluating domain-specific datasets, noisy data, streaming, low-resourced, and complicated applications, and rare and specific words. These challenging conditions are a primary motivation for ongoing ASR development. While models may perform well on general tasks, practical applications often demand accurate recognition of specific information and key entities — precisely the cases where models are more likely to fail despite strong overall performance. Models with strong generalisation capabilities can enhance their performance on such outliers by adapting to specific target domains or incorporating contextual knowledge.

When there is a need to improve the performance of a model in a specific domain or in recognising specific words and terms, the most technically straightforward method is to continue training the model for several more iterations on the target data in that domain. Further model training updates the model's weights, tuning them to the target domain and its lexicon. This method is called *domain fine-tuning* and usually works well when the adaptation data match the test data. However, this method has its limitations. First, the straightforward way to train an ASR model is to use an annotated corpus, which means that we must provide paired AUDIO-TEXT training data for the domain we are targeting. Transcribed data is not always available because it is very expensive to produce compared to text data only. Second, when the parameters of the base model are updated to the particular domain, it can overfit to the new data, and its overall performance on other data can degrade. This is not convenient in case we have several different domains in the data and do not want our model to loose its generalisation strength. Third, fine-tuning has to be performed as a separate training step before the tuned model can

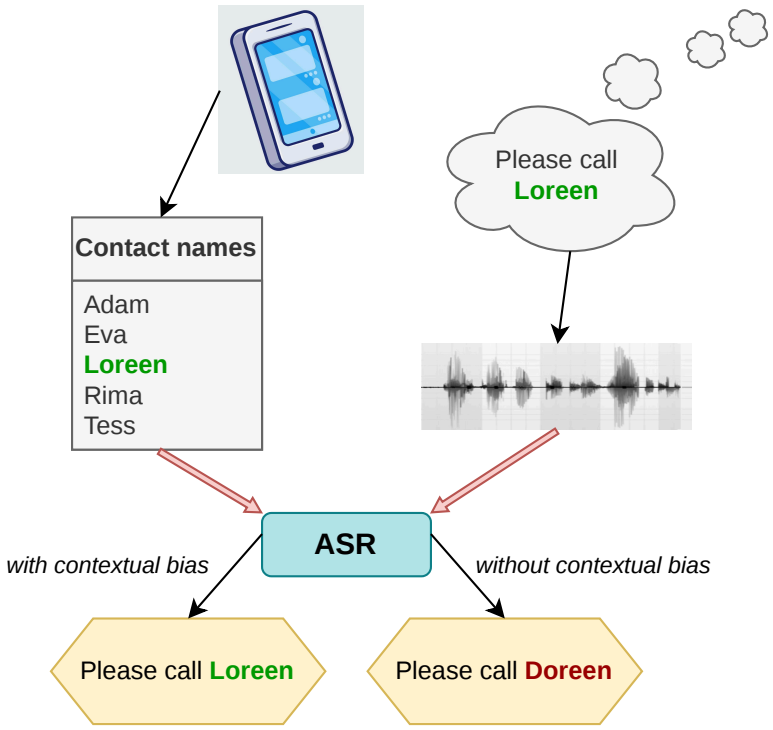

Figure 1.2.: An example of contextual biasing with the user's data for the acoustically close words **Loreen** and **Doreen**.

be used. Moreover, retraining can be problematic in situations where the domain or context of a conversation changes rapidly and unpredictably, and there is no time for new adaptation training. Finally, retraining the model includes the need for expert knowledge, annotated training data, and additional time and work load.

Taking into account the above limitations, a solution for ASR with dynamically incorporated text-only context data and without specialised training of the base model is useful for many practical scenarios. Customisation possible on the end-user side is especially valuable, as it allows contextualisation of the ASR system without expensive expert knowledge and workload, which makes it fast and flexible in use. ASR models that incorporate context provide more natural and relevant interactions, reducing the need for users to correct misrecognised words. This is particularly important for virtual assistants, dictation software, and customer service chatbots. Different industries and use cases require ASR models to adapt to specialised vocabularies and terminologies. Contextual adaptation allows ASR systems to understand domain-specific jargon, improving recognition accuracy in fields such as legal, medical, and technical domains. For example, contextualisation is important for call center conversations when the ASR system has to quickly adapt

to different topics or when clients' personal information can be used (see an example of contextual biasing with the user's data at Fig. 1.2).

Context in speech is often dynamic, influenced by the conversation's flow, speaker intent, and environmental factors. ASR systems must adapt in real-time to changing contexts without relying solely on predefined language models. By leveraging real-time contextual signals, such as speaker intent, location, and previous conversation history, ASR models can dynamically adjust their predictions. For example, the knowledge about the aircrafts in the air space collected by radar every few seconds can be injected into the ASR system as text context to enhance understanding in communication between pilots and controllers.

Another advantage of using only text data and keeping the base model unmodified is *privacy preservation*, which can be crucial for users' security. Personalisation of an ASR system for different users can improve its recognition performance when users' personal information (contact list, playlist, favourite locations, etc.) is available. ASR systems learn from decentralised user data and adapt to individual users' speech patterns while maintaining generalisation across broader datasets.

Crucially, text data provides compelling practical advantages: text data is considerably easier to collect compared to paired data, it requires only a little memory, and text data can be introduced to the model as a list of entities or as a graph.

1.2. Outline

We propose textual context-aware ASR biasing methods, i.e. techniques designed to bias ASR model predictions toward desired target entities, across various configurations, as outlined in Fig. 1.3. The methods are tested on conversational datasets from different domains, including air traffic communication, banking, healthcare, insurance, and earnings.

Hybrid vs. End-to-End ASR. We distinguish between contextualisation methods applied to (1) *pipeline* (or hybrid) ASR systems and (2) *End-to-End* ASR systems. In hybrid ASR models — which typically consist of separate acoustic, pronunciation, and language components — contextualisation is primarily achieved by adapting the language model. This can include incorporating domain-specific text corpora, biasing recognition toward relevant phrases, and integrating external knowledge sources such as user profiles or knowledge graphs. In contrast, End-to-End ASR models are built using deep learning architectures like transformers or transducers, where all components are trained jointly and optimised toward a single

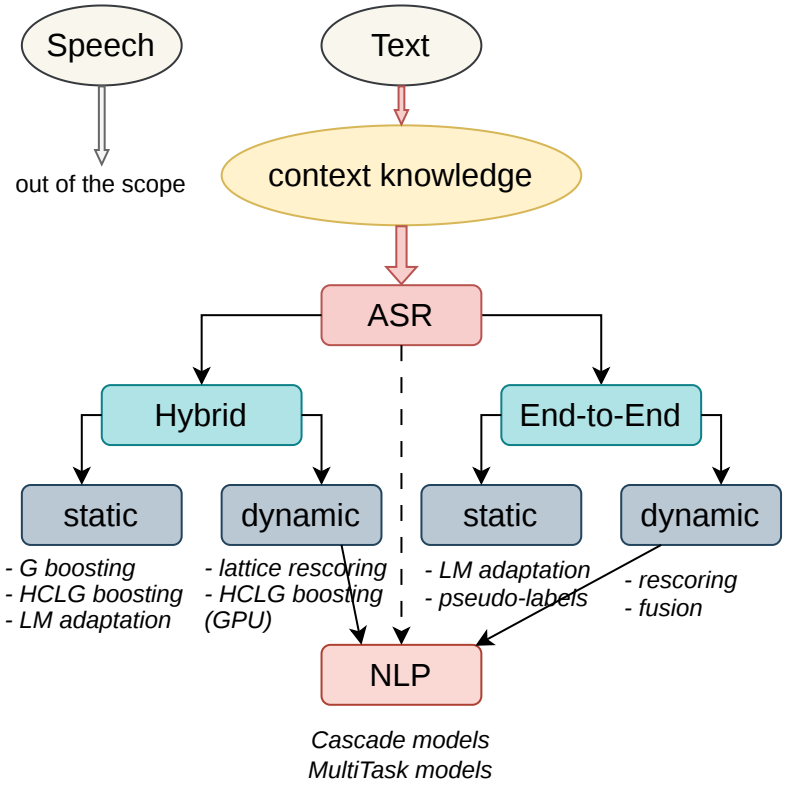

Figure 1.3.: An overview of the thesis.

objective function. Due to their unified structure, it is more challenging to adjust individual components in isolation. Consequently, contextualisation in End-to-End models relies on strategies such as large-scale pretraining on diverse datasets, use of attention mechanisms to attend to context words, contextual word embeddings, and external language model integration. These models can further benefit from fine-tuning on domain-specific data and employing context-aware decoding techniques, such as biasing output based on dialogue history or speaker intent.

Dynamic vs. static context integration. The second distinction we make is between *dynamic* and *static* approaches to context integration. A dynamic approach refers to contextualisation that occurs during inference without requiring additional preprocessing, pre-training, or specialised architectures. In contrast, a static approach involves incorporating context either prior to decoding or through dedicated model architectures designed to integrate contextual information during training or inference. Additionally, we investigate solutions for fast context integration including the ones suitable for streaming ASR: from fast and light offline contextualisation to "on-the-fly" online context integration for streaming applications.

Contextualisation for ASR and related applied tasks. Finally, we consider the role of contextualisation in supporting downstream tasks and enabling semi-supervised learning. For downstream tasks such as machine translation, sentiment analysis, and named entity recognition, incorporating contextual information from ASR outputs ensures that transcriptions are more accurate and meaningful. This improves the overall performance of subsequent models that rely on ASR-generated text. In semi-supervised learning, where models are trained using a combination of labelled and unlabelled data, contextualisation helps in refining pseudo-labelling strategies and improving representation learning. By leveraging contextual embeddings and pre-trained language models, semi-supervised ASR systems can better generalise to unseen data, reducing errors caused by ambiguous or out-of-vocabulary words. As a result, contextualisation enhances both the robustness and efficiency of models in low-resource settings, enabling better performance with limited labelled data.

The various configurations outlined above define how contextualisation is approached in each chapter of the thesis:

Chapter 2 This chapter establishes the fundamental theoretical and methodological framework of the thesis. It first provides an overview of key principles and architectures underlying ASR models, along with contextualisation techniques, and specify those adopted in this work. Then, it describes the types of textual data necessary for ASR contextualisation, details the datasets used in the experiments, and explain the evaluation metrics applied.

Chapter 3 This chapter focuses on the contextualisation techniques applied with the *pipeline (hybrid) ASR models*. It covers three different methods for *static* and *dynamic* boosting of target entities during inference and inside the language model. We test the methods on sequences of words with frequent words, as well as on rare and out-of-vocabulary words. In addition, we present an implementation of contextual biasing within the decoding graph, executed directly on GPUs to support real-time decoding. The work described in the chapter has been published in (Nigmatulina et al., 2023; Bhattacharjee et al., 2023, 2024; Zuluaga-Gomez et al., 2023a); a part of the work is available online in a pre-print (Nigmatulina et al., 2021).

Chapter 4 This chapter explores contextualisation techniques specifically designed for *End-to-End ASR models*. We consider two biasing methods for contextualising during inference. Both of them are *dynamic* approaches applied directly at the decoding stage. A part of the work described in the chapter has been published in (Thorbecke et al., 2025).

Chapter 5 The chapter presents preliminary findings that have not yet been published. If in Chapter 4, we use lists of entities as adaptation data, in Chapter 5, adaptation data is a domain-specific text corpus. We investigate two methods for domain adaptation of *End-to-End models*. The first method can be described as *static contextualisation*, when only the language model component is adapted. The second method differs from the others because, while it aligns with the primary principle of text-only contextualisation, it diverges from the secondary principle of leaving the base model unchanged. This approach involves fine-tuning the entire model using both text and synthetic audio-like features generated from the text. Additionally, we address the topic of language modelling within End-to-End ASR, highlighting the unique challenges it poses rather than treating it as a straightforward task.

Chapter 6 In this chapter, we explore how contextual information can be harnessed to enhance both downstream *Natural Language Processing (NLP)* tasks based on ASR outputs and the training of ASR models in low-resource settings. We present experimental evidence and analysis demonstrating that contextual biasing enhances performance on the named entity recognition (NER) task for both pipeline and End-to-End ASR systems. We also demonstrate the benefits of contextual biasing in scenarios with limited training data, specifically, in (1) *semi-supervised learning*, and (2) improving the accuracy of streaming ASR models trained from scratch using pseudo-labels. The work described in the chapter has been partly published in (Nigmatulina et al., 2022; Zuluaga-Gomez et al., 2021; Thorbecke et al., 2024).

Chapter 7 This chapter begins by summarising our key contributions and findings, followed by a discussion of potential directions for future research.

1.3. Publications

Publications included in the thesis.

Accepted as the first author:

1. **Nigmatulina, I.**, Zuluaga-Gomez, J., Prasad, A., Sarfjoo, S. S., Motlicek, P. (2022). A two-step approach to leverage contextual data: speech recognition in air-traffic communications. In ICASSP 2022-2022 IEEE international conference on acoustics, speech and signal processing (ICASSP) (pp. 6282-6286). IEEE. (Nigmatulina et al., 2022)

2. **Nigmatulina, I.**, Madikeri, S., Villatoro-Tello, E., Motlíček, P., Zuluaga-Gomez, J., Pandia, K., Ganapathiraju, A. (2023). Implementing contextual

biasing in GPU decoder for online ASR. INTERSPEECH. (Nigmatulina et al., 2023)

3. **Thorbecke, I.**, Zuluaga-Gomez J., Villatoro-Tello E., Kumar S., Rangappa P, Burdisso S., Motlicek P., Pandia K., Ganapathiraju A. (2024). Fast Streaming Transducer ASR Prototyping via Knowledge Distillation with Whisper. EMNLP Findings. (Thorbecke et al., 2024)

4. **Thorbecke, I.**, Villatoro-Tello, E., Zuluaga-Gomez, J., Kumar, S., Burdisso, S., Rangappa, P., Carofilis, A., Madikeri, S., Motlicek, P., Pandia, K., Hacioğlu, K., and Stolcke, A. (2025). Unifying Global and Near-Context Biasing in a Single Trie Pass. The 28th International Conference of Text, Speech and Dialogue (TSD2025). (Thorbecke et al., 2025)

Accepted as the second and third author:

1. Zuluaga-Gomez, J., **Nigmatulina, I.**, Prasad, A., Motlicek, P., Veselý, K., Kocour, M., Szöke, I. (2021). Contextual semi-supervised learning: An approach to leverage air-surveillance and untranscribed ATC data in ASR systems. INTERSPEECH. (Zuluaga-Gomez et al., 2021)

2. Zuluaga-Gomez, J., **Nigmatulina, I.**, Prasad, A., Motlicek, P., Khalil, D., Madikeri, S., ... Choukri, K. (2023). Lessons Learned in Transcribing 5000 h of Air Traffic Control Communications for Robust Automatic Speech Understanding. Aerospace, 10(10), 898. (Zuluaga-Gomez et al., 2023a)

3. Bhattacharjee, M., **Nigmatulina, I.**, Prasad, A., Rangappa, P., Madikeri, S., Motlicek, P., Helmke, H., and Kleinert, M. (2024). Contextual Biasing Methods for Improving Rare Word Detection in Automatic Speech Recognition. In ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 12652-12656. IEEE, 2024. (Bhattacharjee et al., 2024)

4. Bhattacharjee, M., Motlicek, P., **Nigmatulina, I.**, Helmke, H., Ohneiser, O., Kleinert, M., Ehr, H. (2023). Customization of Automatic Speech Recognition Engines for Rare Word Detection Without Costly Model Re-Training. 13th SESAR Innovation Days. (Bhattacharjee et al., 2023)

Non-peer-reviewed papers:

1. **Nigmatulina, I.**, Braun, R., Zuluaga-Gomez, J., Motlicek, P. (2021). Improving callsign recognition with air-surveillance data in air-traffic communication. arXiv preprint arXiv:2108.12156. (Nigmatulina et al., 2021)

Publications NOT included in the thesis.

1. Kumar, S., **Thorbecke, I.**, Burdisso, S., Villatoro-Tello, E., KE, M., Hacioğlu, K., Rangappa, P., Motlicek, P., Ganapathiraju, A., Stolcke, A. (2025). Performance evaluation of slam-asr: The good, the bad, the ugly, and the way forward. In 2025 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW) (pp. 1-5). IEEE. (Kumar et al., 2025b)

2. Kumar, S., Madikeri, S., Zuluaga-Gomez, J., Villatoro-Tello, E., **Thorbecke, I.**, Motlicek, P., Manjunath, KE. Ganapathiraju, A. (2025). XLSR-Transducer: Streaming ASR for Self-Supervised Pretrained Models. In ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 1-5). IEEE. (Kumar et al., 2025a)

3. Kumar, S., Madikeri, S., Zuluaga-Gomez, J., **Thorbecke, I.**, Villatoro-Tello, E., Burdisso, S., Motlicek, P., Pandia, K. and Ganapathiraju, A., (2024). TokenVerse: Towards Unifying Speech and NLP Tasks via Transducer-based ASR. In ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 1–5. IEEE.(Kumar et al., 2024b)

4. Kumar, S., Madikeri, S., **Nigmatulina, I.**, Villatoro-Tello, E., Motlicek, P., Pandia, K., Dubagunta, S.P. and Ganapathiraju, A. (2024). Multitask speech recognition and speaker change detection for unknown number of speakers. In ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 12592-12596). IEEE. (Kumar et al., 2024a)

5. Motlicek, P., Prasad, A., **Nigmatulina, I.**, Helmke, H., Ohneiser, O. and Kleinert, M., (2023). Automatic Speech Analysis Framework for ATC Communication in HAAWAII. 13th SESAR Innovation Days 2023, SIDS 2023. (Motlicek et al., 2023)

6. Zuluaga-Gomez, J., Prasad, A., **Nigmatulina, I.**, Motlicek, P. and Kleinert, M. (2023). A virtual simulation-pilot agent for training of air traffic controllers. Aerospace, 10(5), p.490. (Zuluaga-Gomez et al., 2023b)

7. Zuluaga-Gomez, J., Sarfjoo, S.S., Prasad, A., **Nigmatulina, I.**, Motlicek, P., Ondrej, K., Ohneiser, O. and Helmke, H. (2023). Bertraffic: Bert-based joint speaker role and speaker change detection for air traffic control communications. In 2022 IEEE Spoken Language Technology Workshop (SLT) (pp. 633-640). IEEE. (Zuluaga-Gomez et al., 2023d)

8. Zuluaga-Gomez, J., Prasad, A., **Nigmatulina, I.**, Sarfjoo, S.S., Motlicek, P., Kleinert, M., Helmke, H., Ohneiser, O. and Zhan, Q. (2023). How does pre-trained wav2vec 2.0 perform on domain-shifted asr? an extensive benchmark on air traffic control communications. In 2022 IEEE spoken language technology workshop (SLT) (pp. 205-212). IEEE. (Zuluaga-Gomez et al., 2023c)

1.4. Contribution to the projects

This thesis has been performed in the context of research and industrial projects, and focuses on the following application areas:

1.4.1. Air-Traffic communication (STARFISH project)

In the STARFISH[1] project with Deutsche Zentrum für Luft- und Raumfahrt (DLR) and Frankfurter Flughafen AG (Fraport), I worked on collecting and preprocessing audio data, training ASR models, "on-the-fly" integration of radar data to improve the ASR predictions, and software for automatic streaming transcription and recording pilots-controllers conversations.

1.4.2. Call center conversations (Uniphore project)

In the industrial Uniphore project, I worked on problems of contextualisation and personalisation of hybrid and End-to-End ASR models to improve the recognition of call center conversations in English from different and often changing domains. Additionally, I worked on the implementation of contextual biasing for streaming decoding.

[1] https://www.dlr.de/de/medien/publikationen/magazine/alle-magazine-webversion/dlrmagazin-172/wir-verstehen-uns

2. Background

This chapter lays down the basic theoretical and methodological aspects of the thesis. The first two sections overview of the most important principles and architectures of ASR models, as well as methods of contextualisation, and define those that are used in the present thesis. The last three sections outline the types of text data required for ASR contextualisation, the datasets used in the experiments, and the evaluation metrics employed.

2.1. Automatic Speech Recognition

Automatic Speech Recognition (ASR) is the technology that converts spoken language into written text, mapping an acoustic input sequence of length T to a corresponding text sequence of N words. Formally, ASR seeks to model the conditional probability $p(Y|X)$, where $Y = \{y_1, ..., y_n\}$ is a word sequence drawn from the vocabulary $\mathcal{V}$, and $X = \{x_1, ..., x_t\}$ represents the sequence of observed acoustic feature vectors. The goal is to find the most probable word sequence Y given the observed acoustic input X, effectively selecting the best candidate sentence from all possible word combinations in the language. This formulation follows the standard ASR framework described in (Jurafsky and Martin, 2009; Wang et al., 2019):

$$\hat{Y} = \underset{Y \in \mathcal{V}}{\operatorname{argmax}}\, p(Y|X) \tag{2.1}$$

Bayes's theorem is then applied to reformulate the Eq. 2.1:

$$\hat{Y} = \underset{Y \in \mathcal{V}}{\operatorname{argmax}}\, \frac{p(X|Y)p(Y)}{p(X)} = \underset{Y \in \mathcal{V}}{\operatorname{argmax}}\, p(X|Y)p(Y) \tag{2.2}$$

where $p(Y)$ is a language model and the probability of acoustic features given a word $p(X|Y)$ can be regarded as an acoustic model.

The first step in ASR training involves transforming the raw acoustic signal into a

numerical representation — a sequence of feature vectors. To capture the signal's temporal dynamics, the waveform is segmented into short, overlapping frames, typically 25 milliseconds long with a 10-millisecond stride. This sliding window provides a good balance between temporal resolution and phonetic detail. The chosen frame length is sufficient to represent individual phonemes while maintaining local stationarity, which reduces ambiguity when mapping acoustic features to phonetic units. Each frame is then encoded as a feature vector, resulting in a sequence that serves as input to the acoustic model. The most widely used acoustic features are Mel-Frequency Cepstral Coefficients (MFCC) (Davis and Mermelstein, 1980). Sometimes the cepstrum analysis step is omitted and, then, FBANK features, which are the output from a Mel-frequency filterbank, are used instead (Mohamed et al., 2011).

Conceptually, all ASR models can be categorized as either (1) traditional, i.e. hybrid, pipeline models, or (2) End-to-End models. Over the past few years, deep learning techniques have become integral to almost all ASR systems. Both hybrid HMM-deep neural network (HMM-DNN) and End-to-End speech recognition models use deep learning techniques and perform similarly.

2.1.1. Hybrid ASR models

The ASR pipeline typically involves training three probabilistic models, followed by a decoding stage in which their outputs are combined. Each of these models captures a different aspect of spoken language: the *acoustic model* (AM), the *pronunciation model* (PM), and the *language model* (LM).

Acoustic model. The AM maps the input acoustic signal — represented as a sequence of feature vectors — to phonetic units or sound "labels" known as phonemes. In this work, with hybrid ASR, we will refer to a model with a neural network AM that estimates posterior probabilities over HMM states, i.e. NN–HMM ASR system.

Lexicon. The PM, often referred to as the *lexicon* or *dictionary*, is a table that establishes the mapping between phonemes and graphemes, effectively linking sounds to their written representations. This model is either curated by linguists or, when data permits, learned automatically from aligned training data. For example, the lexicon is often created with the grapheme-to-phoneme (G2P) module that converts words into their transcriptions. The *vocabulary* defines the finite set of words that an ASR system can recognise, with each word linked to its pronunciation through entries in the lexicon.

Language model. The LM predicts the most probable sequences of words in a

language, using the outputs from both the AM and PM to guide the construction of coherent and linguistically valid sentences. LMs can be broadly categorised into two types: statistical and neural. Statistical language models (SLMs) estimate the probability distribution over sequences of words using n-gram statistics. In contrast, neural language models (NLMs) learn distributed word representations, enabling them to capture more complex dependencies, generalize better, and scale more effectively with large vocabularies (Bengio et al., 2003). While NLMs have become increasingly popular, SLMs remain widely used due to their lower computational requirements and superior performance on small datasets — particularly when pretrained models are unavailable, such as in certain hybrid ASR systems (Xiong et al., 2018).

Representation of knowledge sources. For decoding with a hybrid ASR system, the knowledge sources can be represented in different ways. For example, the Hidden Markov Model Toolkit (HTK) (Young and Young, 1993) uses tree lexicons and phone/state networks to construct the search space[2]. The search strategy with trees used for search space construction is known as the *history conditioned lexical tree (HCLT)* (Ney et al., 1992; Ney and Ortmanns, 2002b). The lexical prefix tree can be constructed with varying degrees of granularity and scale. In this structure, leaf nodes signify the completion of one or more word pronunciations; a single leaf may represent multiple lexical entries in the case of homophones or when words share an identical sequence of tied HMM states. Because the specific identity of a hypothesis is only resolved upon reaching these terminal nodes, LM scores are integrated exclusively at word-end states (Ortmanns and Ney, 2000). To satisfy the requirements of an n-gram LM, search hypotheses must be differentiated based on their unique $(n-1)$ word histories, ensuring that transitions are calculated with the correct linguistic context.

Another way to represent the knowledge sources is *weighted finite state transducers (WFSTs)* (Mohri, 1997; Mohri et al., 2002), e.g. used in the Kaldi toolkit (Povey et al., 2011). While comparative studies have shown that HCLT and WFST-based search strategies can achieve similar efficiency depending on the task and system configuration (Rybach et al., 2011, 2013), the efficiency is distributed in different ways (Ortmanns and Ney, 2000). The HCLT decoder is characterised by efficient intra-word state expansions within its prefix tree structure; however, it incurs significant computational overhead during the LM look-ahead score estimation and word-boundary processing. Conversely, the WFST-based decoder experiences increased costs for within-word transitions due to the complexity of real-time composition, but

[2] The *search space* is a set of all possible paths of a given length in the search network given a certain acoustic observation.

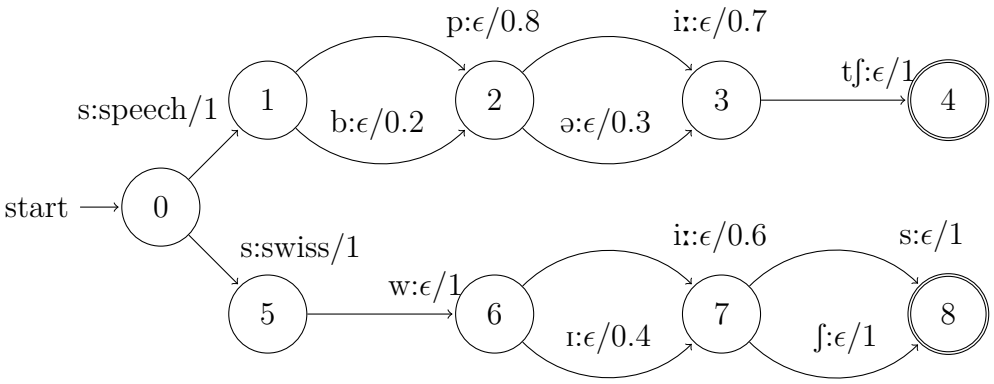

Figure 2.1.: Weighted finite-state transducer example illustrating a relationship between two levels of representation: pronunciation lexicon and language model $L \circ G$ (The input label i, the output label o, and weight w of a transition are marked on the corresponding directed arc by $i : o/w$).

it offers distinct advantages in the LM look-ahead scoring efficiency and hypothesis recombination. As in this thesis we work with the Kaldi toolkit, we will always use the WFST-based decoding.

A WFST function is a finite state machine comprising a set of states, including a designated start state and one or more final states, and transitions (arcs) between them, each associated with a weight or probability. These transitions are labelled with both input and output symbols, allowing the transducer to map an input symbol sequence to an output sequence as it follows a path through the states (Partee et al., 2012).

This property allows the transducer to combine different levels of representation, for example, phones and words, or the Hidden Markov Models (HMM) and context dependent (CD) phones (see Fig. 2.1). The combination is implemented with the operation of *composition* that maps sequences from input transducers in such a way that an output from one transducer should be an input to another (Mohri et al., 1998; Dolfing and Hetherington, 2001; Hori and Nakamura, 2005; Allauzen et al., 2009). In other words, with composition, one can integrate into a system information from additional knowledge sources.

In ASR, different WFST components are composed together in the final *decoding graph* $HCLG$ (Mohri et al., 2002; Povey et al., 2012):

$$HCLG = H \circ C \circ L \circ G \tag{2.3}$$

where H, C, L and G represent the HMMs (H.fst) structure that keeps transition probabilities, CD (C.fst) that maps states of triphones to phonemes, lexicon (L.fst)

that keeps pronunciation probabilities, and LM (or grammar, G.fst) that keeps n-gram probabilities. The HCLG graph is used to generate the best hypotheses during decoding.

Decoding. During decoding, the posterior probabilities estimated by the AM are converted into state likelihoods using the Bayes' rule, which requires division by the state prior probabilities. This procedure, commonly referred to as state prior correction, compensates for the imbalance in state frequencies observed during training and removes an implicit linguistic bias learned by the neural network. As such, state prior correction can be viewed as a simple form of internal language model compensation, ensuring a more proper separation between acoustic and language-model contributions during decoding.

Once the acoustic model produces observation likelihoods (or emission probabilities) for each frame, the decoding step aims to identify the most probable word sequence. This involves computing the joint probability of each state and its corresponding observation at each time step. These values are then combined with the prior probabilities provided by the language model (see Eq. 2.1) to determine the most likely word sequences.

Rather than producing only the single best hypothesis, the decoder typically generates multiple high-scoring candidates. A compact and efficient way to represent the decoding hypotheses is through a word *lattice*, or a search graph (or WFST) of an utterance of T frames that is used to find the best decoding path. To optimise the search through the lattice, Viterbi search algorithm with beam pruning is typically employed (Forney, 1973).

Hybrid systems continue to be among effective and versatile approaches for developing ASR engines, especially when only limited amounts of transcribed audio data are available. However, the rise in computational resources and the abundance of unlabeled data have led to the emergence of new possibilities for ASR training. For example, End-to-End ASR models need neither HMM nor complex pronunciation lexicons compared to hybrid models.

2.1.2. End-to-end ASR models

End-to-End models are typically sequence-to-sequence architectures that directly convert audio inputs represented as sequences of acoustic feature vectors into corresponding word or grapheme sequences. Unlike traditional ASR systems, E2E models eliminate the need for intermediate components by learning a direct mapping be-

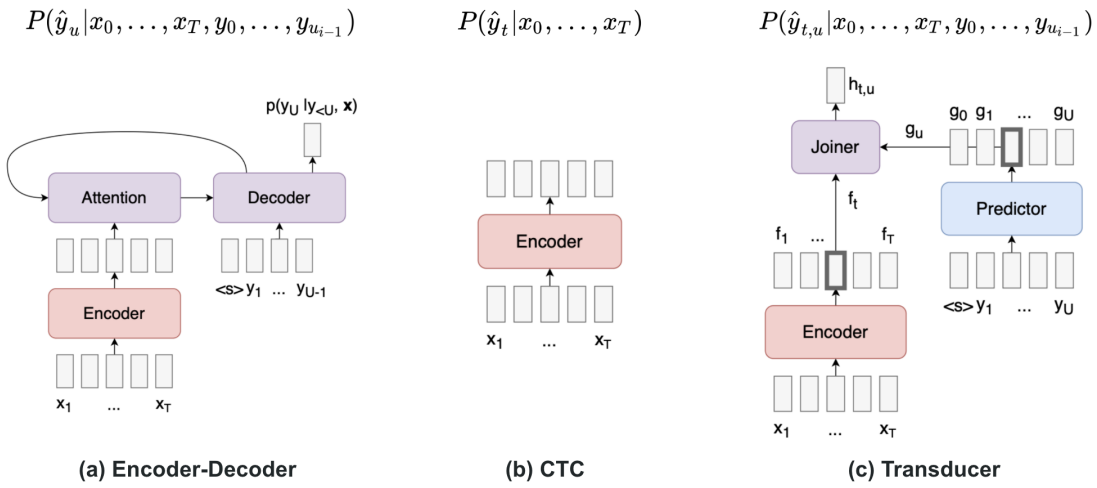

Figure 2.2.: ASR End-to-End architectures: (a) encoder-decoder, (b) CTC, and (c) transducer (Lugosch, 2020).

tween input and output labels. Through joint training, these models optimise a global objective function aligned closely with the final evaluation metrics, enabling more effective and holistic learning (Graves and Jaitly, 2014; Wang et al., 2019).

Although End-to-End models follow a unified framework, their architecture is typically composed of three main components: the **encoder**, which transforms the input speech signal into a sequence of features, effectively serving as the acoustic model; the **decoder**, which generates the final transcription and functions similarly to a language model; and the **aligner**, which establishes the correspondence between the audio features and the textual output, aligning the input and output sequences. Since neural networks in speech recognition are typically trained as frame-level classifiers, each frame requires a corresponding training target. Each phoneme may span a variable number of frames in the audio signal, making the acoustic frame sequence significantly longer than the corresponding text label sequence. To associate frames with their target labels, an alignment between the audio and transcription must be established that accounts for the temporal variability of speech. In traditional hybrid models, this alignment, referred to as "forced" alignment, is provided by the HMM and is progressively refined through iterative retraining and realignment. End-to-End models employ a "soft" alignment approach, where each audio frame is associated with a probability distribution over all possible output states, eliminating the need for an explicit or forced alignment. Depending on the implementation of soft alignment, there are three types of End-to-End model architecture: 1) Attention-based Encoder-Decoder, 2) Connectionist Temporal Classification (CTC), and 3) Transducer (see Fig. 2.2).

Attention encoder-decoder. The encoder-decoder (AED) architecture consists of three main components: *encoder*, *decoder*, and *attention mechanism* (aligner) (Chorowski et al., 2015; Bahdanau et al., 2016; Chan et al., 2015). Thus, the soft alignment between the input data and the output labels is calculated directly with the attention mechanism (see Fig. 2.2-a). Attention encoder-decoder is a powerful architecture for many tasks. However, in the context of ASR, it has a few drawbacks. First, the attention mechanism becomes computationally intensive with long input sequences, as it requires attending to the entire input for each output step. This results in a complexity of $O(TU)$, which can be particularly costly in speech tasks where both T (input length) and U (output length) are large. Second, the default attention models cannot operate in real-time (online) because they require access to the entire input sequence before the decoder can perform attention. Finally, the default attention models do not take advantage of the monotonic alignment between speech inputs and text outputs when the order of segments in the input sequence coincides with the order in the output sequence (opposite from the translation task).

Connectionist Temporal Classification. The Connectionist Temporal Classification (CTC) loss function was proposed for labelling sequence data without pre-segmented training data, when input and output sequences are monotonically aligned and can be of different lengths (Graves et al., 2006). The core idea of the CTC method is to treat the outputs as a probability distribution over all possible label sequences, conditioned on the input sequence. This allows the objective function to directly maximise the probability of the correct label sequence, thereby addressing the challenge of hard alignment during loss computation.

To encode spelling duplicate characters, the CTC method adds the pseudo-character *'blank'* label ('-') to the set of output labels. The *'blank'* label serves to mark the border of two characters in the output transcription: for example, *–hhee-lll-ll-ooo* $\rightarrow$ *hello* instead of *hhheeellllllooo* $\rightarrow$ *helo*. Therefore, given an input sequence $X = \{x_1, ..., x_t\}$ of length T, a CTC alignment a is a length T sequence of blank and label indices. The probability $p(a|x)$ of a is the product of the emission probabilities at every time-step (Graves and Jaitly, 2014):

$$p(a|x) = \prod_{t=1}^{T} p(a_t, t|x) \tag{2.4}$$

When the probabilities of paths (output sequences) are defined, the second step called many-to-one mapping is to remove all blanks and repeated labels in the path: $a^T \rightarrow a^{\leq T}$. For example, two different paths *"c-aa-t-"* and *"c-a-tt-"* are aggregated to give the same result: *"c-a-t-"* (Wang et al., 2019).

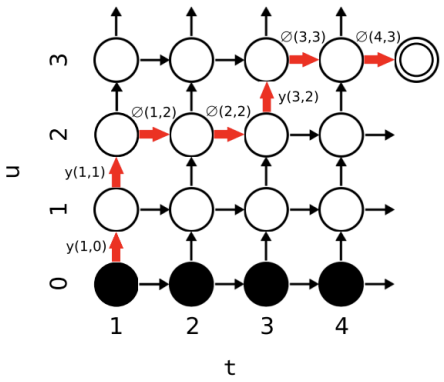

Figure 2.3.: Output probability lattice defined by $P(k|t,u)$ (Graves, 2012).

Limitations of the CTC approach are that (1) it can only map input sequences to output sequences that are shorter than it, and (2) it assumes that output elements are independent of each other and, therefore, cannot explicitly model inter-dependencies and learn the language model[3]. Thus, the CTC model has an encoder with alignment but no decoder, which makes it an acoustic model (see (Fig. 2.2-b)).

Neural transducer. The transducer architecture consists of three subnetworks: *transcription network* (encoder), *prediction network* (predictor, decoder) and *joint network* (aligner) (Fig. 2.2-c) (Graves, 2012; Graves et al., 2013). The encoder provides an 'acoustic' distribution $P(k|t)$, where t is the input timestep. The predictor network models the inter-dependencies within the output label sequence and provides the 'linguistic' distribution $P(k|u)$, where u is the output timestep. The predictor operates in an autoregressive manner, meaning it uses only past observations to predict the value at the next time step. Then, the hidden activations of both encoder and predictor networks are fed into a separate feedforward joint network, which combines and sends them to the last softmax layer to normalise it and output the final label distribution. As the true alignment is not available, the Transducer defines $P(Y|X)$ as the sum of the probabilities of all possible alignments between x and y, similar to the CTC.

The $T * U$ outputs from the joint network can be viewed as a grid (Figure 2.3). Horizontal transitions define proceeding through the acoustic frames and vertical transitions mean consuming labels. Moving from time frame t to $t+1$ is comparable to the empty state in CTC. Non-blank symbols are produced when transitioning vertically, or moving from one output label to the next (e.g. from output label u to $u+1$). It is important to note that in this process, multiple time frames can be used

[3] Contrary to the traditional view of CTC as a purely acoustic mapper, recent research reveals that these models develop a latent internal language model (Yang et al., 2025). Leveraging ILM estimation techniques allows for more effective integration with external LMs, leading to significant performance gains.

without producing a non-blank symbol (similar to CTC), but it is also possible to output multiple labels without moving forward in time. Overall, given a sequence of speech features $X = \{x_1, ..., x_T\}$ where $x_t \in \mathbb{R}^d$, the model is trained to minimise the sum of the negative log posteriors of the reference token sequences $Y = \{y_1, ..., y_U\}$ over the training corpus $\mathcal{D}_T$ where $y_u \in \mathcal{V}$ and $\mathcal{V}$ is the set of non-blank output tokens, e.g., word pieces:

$$\mathcal{L}(\theta) = -\sum_{(X,Y)\in\mathcal{D}_T} logP(Y|X;\theta) \tag{2.5}$$

where θ are the parameters of a Transducer model.

With the predictor fulfilling the language model function and the jointer in addition to the transcription network, the transducer solves two problems of CTC models: 1) it allows multiple outputs for each input; 2) it introduces a dependency between the outputs by adding the predictor and joiner networks. Moreover, the transducer architecture allows decoding in an online/streaming fashion, where each x_t is processed as soon as it arrives. In the transducer framework, the transformer architecture is often used for the encoder network, resulting in the Transformer-Transducer (TT) architecture (Zhang et al., 2020a). This architecture combines the best of the transducer and the transformer and leads to a robust streaming ASR (Li et al., 2020a; Noroozi et al., 2024).

2.1.3. Summary

Hybrid models have a modular structure with separately prepared acoustic and language models and lexicon. The modularity has some advantages, as it makes a model, first, more explainable, and, second, more flexible when different data can be used for training different parts and when the language model can be easily replaced by another one. However, hybrid models (i.e. HMM-based) are limited by various unfavourable factors such as data forced segmentation alignment, independent hypotheses, multi-module individual training inherited from the HMM, and incoherence in optimisation, as modules are optimised separately with different objectives.

The End-to-End systems have a simplified model, joint training, direct output and no need to force data alignment. Such models provide easier “all-inclusive” training and, as they have evolved rapidly in recent times, give more space for research and promise more potential for further improvement. At the same time, End-to-End models are less flexible compared to the hybrid ones, and it can be difficult to modify

a single aspect without updating the whole model. With the End-to-End approach, we miss the fine-grained control and interpretability across different components. For instance, hand-crafted pronunciation lexicons and statistical language models can be tailored to specific applications, enhancing performance in specialised domains like aviation or medical transcription.

The choice between hybrid and End-to-End ASR systems often depends on the specific application context. While End-to-End models generally achieve superior performance in data-rich scenarios, hybrid systems often outperform End-to-End models when pretraining resources are scarce, as they are less reliant on large-scale annotated datasets. The ability of hybrid models to leverage proven techniques such as forced alignment and lattice-based decoding also contributes to their continued use in industry and research. In this thesis, we explore the integration of contextual knowledge in both ASR paradigms. For hybrid systems, we adopt the CNN-TDNNF architecture (Peddinti et al., 2015), while for End-to-End systems, we use the transformer-transducer model.

2.2. Contextual knowledge in ASR

While ASR models perform well in general speech recognition tasks, they often misrecognise uncommon or context-specific words, such as named entities (NEs), technical jargon, or brand names. Integrating contextual knowledge enhances ASR models in several directions. (1) Improving recognition of NEs: contextual biasing ensures that domain-specific entities (e.g., company names, product names, medical terms) are correctly transcribed. (2) Enhancing user experience in voice assistants: by prioritising contextually relevant words, ASR systems provide more accurate responses to user queries. (3) Boosting accuracy in noisy environments: when background noise affects speech recognition, contextual biasing helps the ASR model focus on relevant terms. (4) Optimising performance for domain-specific applications: call centers, legal transcription services, and healthcare applications benefit from context-aware speech recognition. The previous Section 2.1 has shown that hybrid and End-to-End models are trained in a very different way; thus, the integration of context is also realised differently. In this section, we provide a brief overview and discuss methods of contextualisation for both types of ASR models.

2.2.1. Contextualisation with hybrid models

In hybrid ASR systems, which combine separate acoustic, pronunciation, and language models, several methods can be employed to introduce contextual information effectively.

Language model adaptation. One of the most direct ways to incorporate contextual knowledge is by LM adaptation. The most widely used technique for LM adaptation is log-linear interpolation (Klakow et al., 1998). The existing general-purpose LM can be interpolated with a domain-specific model trained on contextual corpora. This enables the system to predict more accurately context-relevant terms while maintaining overall language fluency.

Shallow fusion. Another method shallow fusion, or a late integration technique, where an external LM, trained on context-rich data, is combined with the ASR decoder during inference. Instead of modifying the main LM, the decoder scores from the base LM and the external contextual LM are interpolated on-the-fly. This allows the system to boost the probabilities of context-specific hypotheses dynamically and adapt to varying scenarios, such as different users or domains, with minimal overhead (Hori et al., 2003).

WFST weights adjustment. As mentioned above (see Sec. 2.1.1), in a hybrid ASR system, different knowledge sources, i.e. phone context dependencies, lexicon, and language model, are represented as WFSTs and composed before decoding into a large decoding graph. In this framework, contextual biasing can be applied as a WFST weights adjustment by modifying the weights (or probabilities) of transitions associated with context-relevant terms. Lowering the cost (or increasing the weight) of certain arcs in the WFST makes those words more likely to be chosen during decoding. Contextual biasing can be used in any WFST of the pipeline, e.g. the **grammar** (G.fst) as more permanent integration, or a **lattice** as a flexible on-the-fly biasing. Thus, the method leverages the modular and composable nature of WFSTs, enabling seamless integration of contextual knowledge into the decoding pipeline.

Biasing towards target named entities. Contextual biasing has been applied in different scenarios, for example, improving recognition of NEs such as contacts, locations, films, OOV words etc., to adapt the recognition of voice-driven assistants to users (Hall et al., 2015; Aleksic et al., 2015; McGraw et al., 2016; Velikovich et al., 2018; Serrino et al., 2019). Aleksic et al. (2015) use class-based LM and bias towards the whole class in a particular context. Serrino et al. (2019) firstly detect potential NEs and tag them with a class, then, the input word lattice is composed with 'tagging' FSTs generated for each utterance with a tagged entity. At the final

step, contextually relevant NEs are suggested for given phoneme hypotheses inside the time interval corresponding to a tagged path. Approaches in both papers aim to boost a certain pre-defined semantic class and then to take the best recognition candidate within this class.

2.2.2. Contextualisation with End-to-End models

One limitation of End-to-End models is that they only learn on paired audio and text. While an LM for a hybrid system can be trained on large amounts of unpaired text data, an End-to-End "internal" language model (ILM) is tied to the transcriptions of annotated audio data. The emergence of an implicit ILM in standard End-to-End training is fundamentally rooted in the relationship between the posterior and the prior. According to Bayes' identity, the posterior $P(Y|X)$ and the prior $P(Y)$ are linked via marginalisation over the acoustic distribution $P(X)$, such that $P(Y) = \sum_X P(Y|X)P(X)$, as empirically represented by the training data (Meng et al., 2021c; Zhou et al., 2022). Consequently, maximising the posterior $P(Y|X;\theta_{E2E})$ implicitly optimises the ILM $P(Y;\theta_{E2E})$. Thus, components such as the prediction and joint networks of an RNN-T, or the decoder of an AED, function as acoustically-conditioned LMs, using both token and acoustic embeddings to predict the conditional probability of the next token. Moreover, recent studies demonstrate that an ILM is also developed during the training of CTC models (Yang et al., 2025).

Because the optimisation is performed jointly to minimise the overall End-to-End loss, the resulting ILM is often suboptimal and biased toward the specific domain of the transcribed speech. To overcome this inconvenience, an external language model (ELM) can be used to improve the performance of an End-to-End system. An ELM can be of different sizes and types: from a big LM trained on in-domain data to a small context trie created separately with words and/or word sequences we want to bias in the outputs. However, the same architectural reason mentioned above makes the adaptation of LM to new domains or contexts less straightforward (McGraw et al., 2016; Pundak et al., 2018). Flexible integration of an external contextual information into an End-to-End ASR system can be achieved by methods of two possible directions: 1) *deep context* when the model itself is trained to use contextual information with an attention module, and 1) *shallow fusion* (SF) when model predictions are adjusted at each step of the beam search.

Deep context. The "deep context" class of methods assumes that a model learns how to use extra contextual data from unpaired text (i.e. text without the corre-

sponding audio) during training. Training an ELM here is replaced with an attention module over a list of target-biasing word sequences. Thus, a model learns to focus on a short list of highly probable context phrases, which makes greater use of available context knowledge. Deep context can be used in models with different architectures: for example, transformers (contextual LAS, or CLAS[4] (Jain et al., 2020)) and transducers (Jain et al., 2020; Yang et al., 2024b).

The CLAS transformer approach is modelling $P(y|x, z)$, where z is a list of provided bias phrases (inkl. contact names, song lists, etc.) (Pundak et al., 2018; Chen et al., 2019; Bruguier et al., 2019). The difference between the CLAS and the traditional LAS model is the presence of an additional bias-encoder with a corresponding attention mechanism (Fig. 2.4-a). The bias-encoder produces the embeddings for given contextual phrases and then learns to attend to them. Bias word sequences are not necessarily relevant[5], and one of the model's objectives is to learn to determine which phrases are relevant for the current input and should be used to modify the target distribution, and which should be ignored. In order for the model to learn about the relevance of the phrase, an additional learnable vector corresponding to the no-bias condition is included into the training. Huang et al. (2020) propose the LAS-MOCHA (Monotonic Chunkwise Attention) model (Chiu and Raffel, 2017) following the basic ideas from (Williams et al., 2018; Chen et al., 2019). The authors identify the biasing phrases in the training data and model the transitions between regular words and biasing words. During inference, the user-specific bias phrases are inserted as WFST graphs at the relevant place in the beam search. In addition, the authors propose a method for word mapping, which transforms rare words into common words through pronunciation (manual or G2P).

In the Transducer, to enable contextualisation, a biasing block with an attention model (similar to CLAS) is added to the original Transducer architecture (Graves, 2012; Jain et al., 2020; Chang et al., 2021). The block consists of three components: an Embedding Extractor (EE), an Attention module (AttModule), and a Biasing Module (Fig. 2.4-b). The EE outputs the embeddings of contextual words, which the Attention module uses to compute the attention for each word. Similarly to the CLAS bias-conditioning, the Biasing Module can be added to find an active subset of the context words that have the same prefix as the last unfinished word in the text history Y_{u-1}. For example, if the decoded form of Y_{u-1} is *Africa An* and the list of the context word is *Android, Antenna* and *Pytorch* then the active context

[4] LAS here is for *Listen-Attend-Spell*, a model for ASR with transformer architecture (Chan et al., 2015).

[5] A contextual list can be a mix phrases, some of which are relevant for one utterance but irrelevant for another utterance and, thus, become distractors.

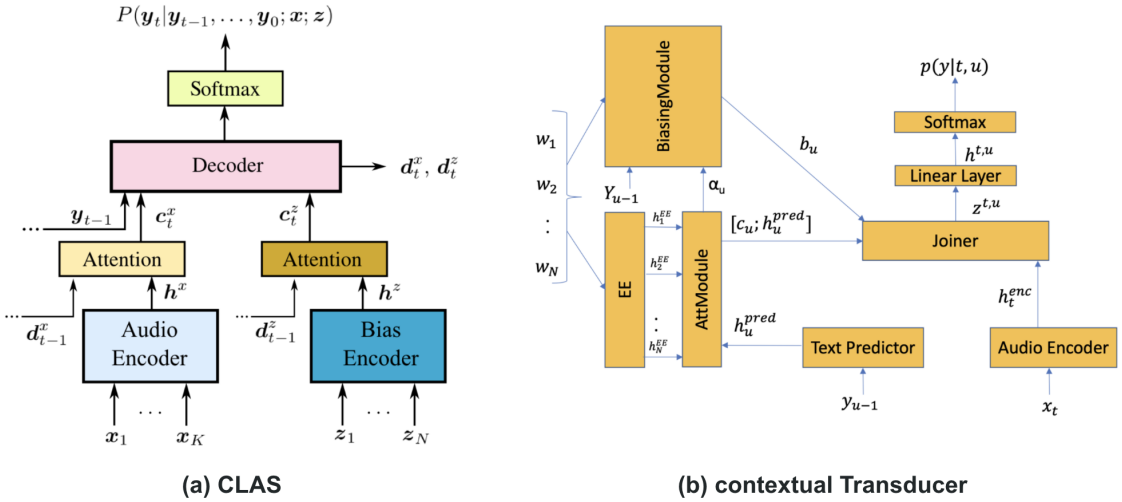

Figure 2.4.: CLAS (Pundak et al., 2018) and contextual Transducer (Jain et al., 2020) architectures.

words are *Android* and *Antenna*.

Overall, deep context methods achieve good results with limited biasing lists, while they tend to fail to scale to large and highly confusable biasing lists. Another inconvenience with deep context models is that they need to be specifically trained to deal with bias phrases that is computationally more costly and can be not always possible. In this case, SF methods allow rescoring on-the-fly, without retraining the model.

Shallow fusion. Although End-to-End systems have less adjustability compared to traditional ones, on-the-fly rescoring can be introduced in the beam search. The SF approach of introducing contextual information is close to the on-the-fly lattice rescoring adapted in the hybrid ASR systems, where an ELM and ASR model remain separate, and only their scores are combined during decoding. The contextual information is integrated with SF by performing log-linear interpolation between the scores of the End-to-End model and the ELM (see Fig. 2.5) (Bahdanau et al., 2016; Chorowski and Jaitly, 2016; Sriram et al., 2017; Kannan et al., 2018; Zhao et al., 2019).

As the End-to-End model is decoded at the grapheme or subword level, the biasing should be either done on the subword unit level (Zhao et al., 2019), or words in the word-level context WFST should be converted into the sequences of subwords (Bahdanau et al., 2016; Williams et al., 2018). For a partial hypothesis (prefix) $y_{1:u}$ given input speech x, the accumulated decoding score is defined as:

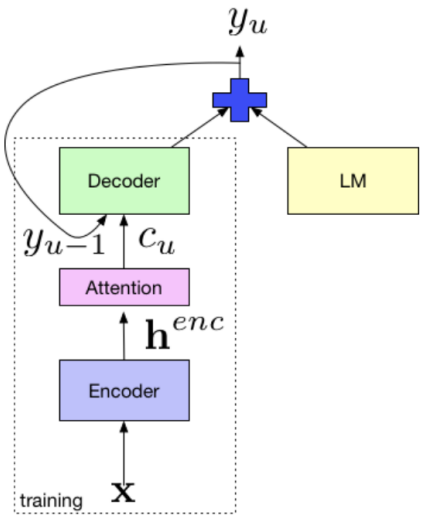

Figure 2.5.: The dotted line box illustrates the LAS architecture; with shallow fusion, an external LM is incorporated via log-linear interpolation (Kannan et al., 2018).

$$Score(y_{1:u}) = \sum_{i=1}^{u} \log P(y_i|y_{<i}, x) + \lambda \sum_{i=1}^{u} \log P_{ELM}(y_i|y_{<i}) \tag{2.6}$$

$$\hat{y} = \text{argmax}\, Score(y) \tag{2.7}$$

where $P_{ELM}(y)$ is an ELM and λ is a tunable hyperparameter controlling how much the contextual LM influences the overall model score during beam search.

A set of biasing phrases can be represented either as an n-gram *WFST graph* (Kannan et al., 2018; Williams et al., 2018) or as a *prefix trie*[6] (Knuth et al., 1977; Ney and Ortmanns, 2002a; Wang et al., 2010; Si et al., 2013; Le et al., 2021b,a). During beam search, the context search space is traversed along with the model's outputs, and contextual rescoring is performed when a match with any n-gram in the set of contextual n-grams is found. The bias function is defined over token hypotheses y_i: for word-level contextual models (e.g., WFST n-grams), the bias is activated only when the emitted token completes a word, whereas for subword or character models (e.g., trie-based prefix matching), the bias may be applied incrementally as soon as the hypothesis prefix matches a contextual entry:

[6] The lexical prefix tree (trie) was originally adopted in ASR to organise the search space for HMM-based Large Vocabulary Continuous Speech Recognition (LVCSR), enabling efficient phoneme sharing across word prefixes (Ravishankar, 1996; Ortmanns et al., 1997).

$$Bias\_score(y_i, y_{<i}) = \begin{cases} b(y_i, y_{<i}), & \text{if contextual match occurs} \\ 0, & \text{otherwise} \end{cases} \tag{2.8}$$

$$Score(y_{1:u}) = \sum_{i=1}^{u} \log P(y_i|y_{<i}, x) + \lambda \sum_{i=1}^{u} \log P_{ELM}(y_i|y_{<i}) + \gamma \sum_{i=1}^{u} Bias\_score(y_i, y_{<i}) \tag{2.9}$$

where γ is a tunable hyperparameter controlling the impact of bias. In particular cases, when target words are expected in a very precise context, incorporating common prefixes (like "call", "text") into SF can further improve the performance (similar to conventional models).

Since only a small number of hypotheses are usually kept in the beam search, limited by the size of the beam, the correct predictions can be pruned before the beam list is formed and thus would not be promoted by the SF. To avoid this, contextual FST can be integrated before the pruning, allowing biasing on earlier stages. To distinguish the two approaches, we call biasing before pruning shallow fusion, and biasing after pruning ***rescoring***:

$$\hat{y} = \text{argmax} \log P(y|x) + \lambda \log P_{ELM}(y) \tag{2.10}$$

To avoid an ILM conflicting with an ELM, some methods for ELM integration during inference aim to subtract the ILM, e.g. the *density ratio method* (DR) (McDermott et al., 2019) or *low-rank density ratio* (LODR) (Zheng et al., 2022). The DR approach uses Bayes' Theorem to mathematically "subtract" the model's internal bias before adding the new, external one:

$$\hat{y} = \text{argmax} \log P(y|x) - \lambda \log P_{ILM}(y) + \beta \log P_{ELM}(y) \tag{2.11}$$

The LODR method is an extension of the DR method, and it works by replacing the estimation of an ILM with a low-order weak language model, i.e. a bi-gram BPE language model, and subtracting its score during SF with an ELM.

Meng et al. (2021c) introduced an Internal Language Model Estimation (ILME) fusion strategy designed to alleviate the bias of the End-to-End model's internal prior. During inference, the ILM score is estimated by suppressing the encoder's acoustic representations – typically through the input of a null or zero vector. This estimated score is then subtracted from the log-linear combination of the End-to-

End and ELM probabilities, isolating the acoustic evidence and allowing for more robust integration of the ELM. Similar to the described approach, Zeineldeen et al. (2021) investigate several advanced strategies for ILM estimation in AED models. Rather than relying on a simplistic zero-vector, these methods employ an estimated bias vector to represent the encoder's contribution. Specifically, the bias vector can be computed by averaging encoder hidden states or context vectors, or by utilising a lightweight LSTM to predict the context vector based solely on the decoder's label history. These estimation techniques were later extended and evaluated for RNN-T architectures by Zhou et al. (2022).

In addition to SF, another method for integrating ELMs into End-to-End ASR systems is known as ***cold fusion***. Unlike shallow fusion, which combines the ELM output with the ASR model during beam search inference through a simple linear interpolation, cold fusion incorporates the ELM directly into the training process, applying a special gating mechanism (Sriram et al., 2017). Cold fusion embeds the LM into the ASR model and trains both simultaneously, allowing the ASR network to learn to rely on the ELM earlier in the process. The contribution of the ELM is regulated by the gating mechanisms. Cold fusion provides tighter integration and better performance by allowing the model to learn how and when to use external language knowledge, albeit at the cost of increased complexity and reduced flexibility, when the whole model should be specifically trained with an ELM.

2.2.3. Summary

Because of the difference between the traditional and End-to-End ASR training, contextual knowledge is introduced differently in these two frameworks. While in hybrid models, text data is integrated directly into an LM, in End-to-End models, the LM is a part of the whole ASR model, and its modification is less straightforward. *Deep context* and *shallow fusion* methods aim to overcome the problem of End-to-End ASR contextual adaptation. Shallow fusion allows flexible adaptation without special training of the whole ASR system. At the same time, the main limitation of shallow fusion is that biasing is performed already on top of decoder outputs. If outputs do not contain correct predictions, biasing will not add them. The deep context, or the attention-based, methods experience more difficulties scaling to large biasing lists.

Since this thesis focuses on the integration of text context without model retraining, most of our experiments use the shallow fusion and FST composition methods.

2.3. Use of text data for contextualisation

Our focus in this thesis is on how to use additional textual resources for ASR contextualisation when no audio data is available. Depending on availability and contextualisation methods, different types of text data can be adopted to enhance ASR contextualisation. They can be *domain-specific text corpora* or specific lists of *target entities*.

Domain-specific text corpora. When ASR models must adapt to different industries, such as healthcare, finance, and legal sectors, where specialized terminology is prevalent, domain-specific text corpora provide relevant linguistic patterns. For example, medical ASR models benefit from datasets containing clinical notes, electronic health records, and medical journals. Recordings of earnings calls, financial conferences, and podcasts from financial analysts can be used to improve the recognition accuracy in the market domain. Depending on the size and quality of the text corpus, the data can be used either to train domain-specific LMs or to fine-tune existing general LMs. Domain-specific language models are then adapted for ASR rescoring or integrated into the ASR model.

Lists of target entities. Dynamic data can be injected directly into the ASR system without the need to train an LM. This data is a list of specific single words, utterances, or named entities (NEs) that are especially important to recognise correctly. Often, it is an utterance-level contextualisation and it is closely related to a certain situation, user, or time moment. For example, in a personalised ASR system, it can be a list of user contacts, a user playlist, favourite locations, and other personalised settings. Adapting the ASR model to a particular conversation, one can introduce proper names, organisation names, and specific terminology.

2.3.1. Use cases

In this thesis, we evaluate the contextualisation of ASR across five specific domains: (1) healthcare, (2) banking, (3) insurance, (4) earnings calls, and (5) air-traffic communication. Examples for each domain are given in Table 2.1 with the target entities highlighted in bold. These domains are derived from the two use cases that I worked on during my PhD time: (i) ASR for call center conversations and (ii) ASR assistance for air-traffic communication (ATC). As the air-traffic use case is of a more specific domain compared to the other four, the following section provides a more detailed overview on ATC.

Table 2.1.: Examples from the five domains used for evaluation

Domain	Example
Healthcare	alright so the **nexium** would be a type of medication that you could actually take daily for extended treatment and that one would help too you know kind of prevent the symptoms and then if if you happen to notice any of them play the symptoms clearing up again that you would take the **nyquil** just to calm it down
Banking	thanks for calling **bank of america** how may i help you
Insurance	okay my name is **devin townsend** and my policy number is **four five seven nine two three one six four nine**
Earnings	good morning ladies and gentlemen, and welcome to the **monro inc** earnings conference call for the third quarter fiscal twenty twenty
Air-Traffic Communication	**sky travel three five juliett** turn right heading one eight zero vectoring for spacing

2.3.2. ATC use case: ASR assisted air-traffic communication

Communication between pilots and Air-Traffic Controllers (ATCo) is mainly based on voice to reduce possible distractions for pilots during flight and to accelerate information exchange. However, the voice-based approach makes communication potentially more error-prone because of the amount and speed of information transmission, environmental noise, and variability of accents due to the international setting. These factors make the task extremely stressful; pilots and ATCos must always be highly concentrated to guarantee a high level of accuracy to provide safety.

Technical support of pilot-controller communication can increase the level of confidence in correct command perception and reduce the workload. Recent progress in ASR, including the online recognition of continuous speech, allows new perspectives to improve communication methods between pilots and ATCos (Geacăr, 2010).

A possible direction to improve the recognition quality is using contextual information, e.g. *surveillance data*. One of the key parts of ATCo commands is a *callsign* — a unique identifier for the aircraft, of which the first part is an abbreviation of the airline name and the last part is a flight number that contains a digit combination and may also incorporate an additional character combination, e.g. **TVS84J**. The callsign data comes from the radar in compressed form, i.e., standardized phraseology format of International Civil Aviation Organization (ICAO) (ica, 2011) (see

Table 2.2.: Examples of callsigns

Callsign	**Extended callsign**
SWR2689	swiss two six eight nine
RYR1RK	ryanair one romeo kilo
RYR1SG	ryanair one sierra golf

Fig. 2.6 and the first column in Table 2.2). To introduce contextual knowledge into the ASR system, all callsigns need to be expanded from the ICAO format to word sequences (second column in Table 2.2). The compressed form often allows for more than one possible realisation in the ATCos' speech: For example, **DLH5KX** can be expanded as *'hansa five kilo x-ray'* or *'lufthansa five kilo x-ray'*. Since we can not say which particular expansion is true for an utterance callsign, it is important to take all expansion variants into account: see the process of retrieving the list of verbalised callsigns (contextual data) in Fig. 2.7.

At a certain time point, only a few aircrafts are usually in the radar zone, which means that only a limited number of callsigns can be referred to in ATCo communications (see Fig. 2.6). If a recognised callsign does not match any callsign registered by radar at the same time point, it means that there is no corresponding aircraft in the air space and the recognised command is invalid. Thus, surveillance information introduced to an ASR system helps to increase the probability of recognising those callsigns that are present in the air space at the moment of utterance. Previous research has shown that radar data can be integrated during semi-supervised learning, i.e., contextual semi-supervised learning (Zuluaga-Gomez et al., 2021).

Besides callsigns, other ATC entities that are challenging for ASR are *waypoints*. Waypoints are specialized terms in air traffic control (ATC) that correspond to specific coordinates. Air traffic controllers (ATCos) and pilots use these terms to navigate and adjust flight paths. These waypoints are typically region-specific. For instance, in the context of en-route navigation in Germany, the waypoints used are unique to that area. However, if the same system were applied to en-route navigation in Austria, the waypoints would likely differ, making them more challenging for the system to recognize accurately. Waypoints are names given to a latitude-longitude pair representing a geographic location. A waypoint name, such as *"WYK"* (pronounced *"wipper"*), is typically an abbreviation consisting of letters or numbers, like *"DL455"*. In ATCo-pilot communication, these waypoints may be referenced in different ways, such as *"wipper"*, *"whisky yankee kilo"*, or *"delta lima four five five"*. When waypoints are spoken using their individual letters, ASR systems can easily recognize them since the ICAO phonetic alphabet is commonly included in English

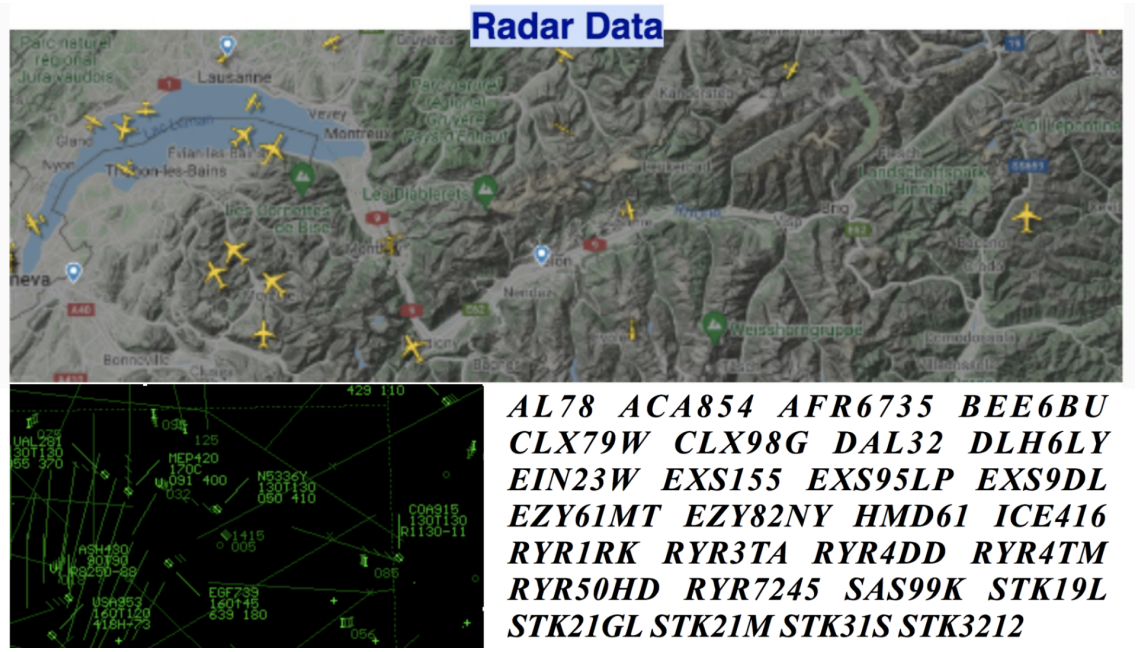

Figure 2.6.: Aircrafts registered in air space and callsigns in ICAO format received from radar.

language training data. However, challenges arise when controllers or pilots refer to waypoints using their artificial names, such as *"wipper"*, which may be newly introduced or rarely encountered during model training.

Contextual information for ATCo ASR has already been used in some previous studies (Shore et al., 2012; Oualil et al., 2015). In (Oualil et al., 2015), to overcome the problem of variability of ATCO commands, the weighted Levenshtein distance is applied as a post-processing to find the closest match between an ASR hypothesis and generated context word sequences. In (Shore et al., 2012), a grammar is used with all the semantic concepts of ATC embedded in XML annotation tags. After decoding, the lattice hypotheses are rescored by adding an additional knowledge source component to the cost function. The knowledge-based rescoring penalises hypotheses which are invalid in the context, e.g. callsigns not registered in the air space. Although this approach helps considerably increase the recognition accuracy, its limitation is that it deals only with concepts and callsigns that are annotated and included in the grammar. Those n-grams that do not appear in the grammar can not be extracted and evaluated.

In addition to ASR performance, contextual information for ATC has also been used to improve concept extraction (Schmidt et al., 2014; Shore et al., 2012; Oualil et al., 2015, 2017). Schmidt et al. (Schmidt et al., 2014) applied a Context-Free Grammar (CFG)-based LM limiting the search space according to the contextual

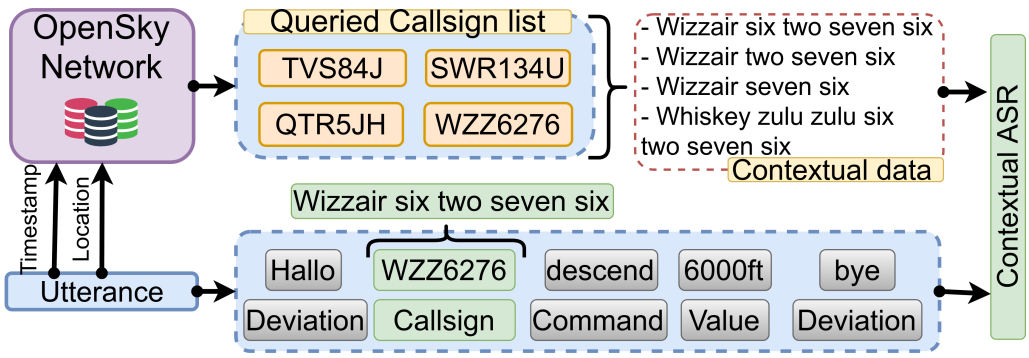

Figure 2.7.: The process of retrieving a list of callsigns (contextual data) from radar. The contextual data is the compendium of all possible verbalised versions of each callsign.

data. Shore et al. (Shore et al., 2012) and Oualil et al. (Oualil et al., 2015, 2017) build a CFG-based concept extractor with all semantic concepts of ATC embedded in XML annotation tags. Oualil et al. (Oualil et al., 2017) combines methods from (Shore et al., 2012; Oualil et al., 2015), adding more contextual constraints from the data with temporal information. Although these methods help considerably increase recognition accuracy, their limitation is that they deal only with annotated concepts and callsigns included in the grammar. Those n-grams that do not appear in the grammar can not be extracted and evaluated. Finally, Helmke et al. (Helmke et al., 2020) recently proposed a machine learning algorithm for command extraction from the ASR hypothesized output using keywords.

Therefore, ASR assistance in speech communication between pilots and ATCos can significantly reduce the complexity of the task and increase the reliability of transmitted information. For this thesis and the topic of ASR contextualisation, the ATC domain is an interesting and challenging use case. (1) Radar data provides lists of highly probable target entities, i.e., callsigns, necessary for contextualization task; (2) the context is always changing depending on the situation of air traffic, which demands the ASR system to be flexible enough to quickly adapt to constantly changing target entities; (3) in practice, online ASR recognition is often needed to help on time.

2.4. Speech Datasets

The selection of datasets for ASR contextualisation experiments largely depends on whether they include a predefined list of named entities that need enhancement. When employing flexible contextualisation without retraining the model, a list of target entities must be available in the test set to enable proper evaluation. For the

test sets we use in this thesis, all ATC test sets are supplied with the corresponding callsign data collected from radar and available per utterance. Two of the three other test sets cover the healthcare, banking, insurance, and earnings domains and have manually verified annotations for NEs. The last test set is the Common Voice dataset of a general topic. As it does not have NEs annotations, we extracted NEs automatically (for more details, see 2.4.2.4). Thus, all test sets have NEs annotations generated manually or automatically or provided from the situation. As not all test sets have the corresponding training sets or only small ones, additional data is used as training data for some of the experiments. In this thesis, we restrict our experiments to English-language data only.

2.4.1. ATC datasets

The ATCo communication is a specific domain for which only limited data is available. Collecting ATC data is a challenging task starting from the recording, which is often done in a noisy environment and can allow only medium quality data with frequencies filtered over 4kHz. The pilot speech is typically more noisy than ATCo speech due to the difference in the environment: pilot cabin VS control tower at the airport. Additionally, data labelling usually requires highly trained transcribers, mainly with ATCo experience. ATC communications require eight to ten man hours effort (Cordero et al., 2012) to annotate one hour of raw controller-pilot dialogues. Recently, a lot of work has been done in the direction of collecting and annotating ATC data (Delpech et al., 2018; Srinivasamurthy et al., 2017; Zuluaga-Gomez et al., 2022). For our callsign experiments, we use six test sets (see details in Table 2.4) and ATC training data created within different European ATC projects.

2.4.1.1. ATC training data

Training data includes ATC speech data of different quality and English accents taken from various public and private datasets: LDCATCC (Godfrey, 1994), N4 NATO (Pigeon et al., 2007), HIWIRE (Segura et al., 2007), ATCOSIM (Hofbauer et al., 2008), AIRBUS (Delpech et al., 2018), MALORCA (Srinivasamurthy et al., 2017; Kleinert et al., 2018), UWBATCC (Šmídl et al., 2019), NATS (Helmke et al., 2023), ISAVIA (Helmke et al., 2023), SOL-96, SOL-97, STARFISH (Kleinert et al., 2023), and ATCO2 (Zuluaga-Gomez et al., 2022). All the data is manually annotated. The datasets are listed in Table 2.3 with their durations and accents.

Additionally, for some of the experiments, we use the data from the **ATCO2 cor-**

Table 2.3.: ATC training data. †The duration is measured after removing silences.

Dataset	Duration†, h	Accents
Manually annotated:		
AIRBUS	39	French
ATCOSIM	8	German, Swiss German, French
HIWIRE	28.3	French, Greek, Italian, Spanish
ISAVIA	19	Icelandic
LDCATCC	23	American
MALORCA Vienna	8	Austrian German
MALORCA Prague	2	Czech
N4 NATO	10.7	Canadian, German, Dutch, British
NATS	24	British
SOL-96	5.5	Austrian German
SOL-97	10.3	Lithuanian
STARFISH	31.5	German
UWBATCC	10.3	Czech
Automatically annotated:		
ATCO2	5,281	-

pus of automatically annotated data. Recently, the European CleanSky EC-H2020 ATCO2 project[7] has developed an ASR-based platform for the collection and automatic pre-processing of ATC speech, radar, and surveillance data (Zuluaga-Gomez et al., 2020; Zuluaga-Gomez et al., 2020, 2022, 2023a). The outcome of the project is the ATCO2 corpus that consists of English voice data coming from several airports worldwide (e.g., Brno, Prague, Bratisliva, Sion, Zurich, Bern, Sydney).[8] The corpus is divided into three subsets: a training set and two test sets. The training set comprises 5,281 hours of unlabeled ATC data, supplemented with automatic transcripts generated by an in-domain speech recognizer. Additionally, it includes contextual information (a list of relevant n-gram sequences per utterance), speaker turn details, signal-to-noise ratio estimates, and an English language detection score for each sample.

2.4.1.2. ATC test data

LiveATC test set The first test set is from LiveATC[9] public data recorded from publicly accessible VHF radio channels, which includes both pilots and ATCo speech and, therefore, is of rather low quality (i.e. low SNR often below 10dB) (Zuluaga-Gomez et al., 2020).

[7] AuTomatic COllection and processing of voice data from Air-Traffic COmmunications, website: https://www.atco2.org/.

[8] The ATCO2 corpus is available for purchase through ELDA in http://catalog.elra.info/en-us/repository/browse/ELRA-S0484/

[9] LiveATC.net is primarily a streaming audio network consisting of local receivers tuned to aircraft communications around the world: https://www.liveatc.net/

Table 2.4.: ATC test sets with surveillance information. † *"Callsigns per utterance"* — median of callsigns per utterance in the surveillance data. ‡ DLR test set does not have information about the callsigns because it is used for the experiments on command recognition.

Test set	Num of utterances with a callsign	Num of utterances w/o a callsign	Callsigns per utterance†	Minutes	All callsigns
DLR‡	655	18	-	52	-
LiveATC	581	29	28	40	280K
Malorca Prague	784	88	5	82	17K
Malorca Vienna	877	38	19	65	59K
NATS	794	73	50	50	168K
ATCO2-test-set-4h	2,834	701	214	240	4'113
ATCO2-test-set-1h	756	85	140	60	3'230

MALORCA test sets The other two non-public test sets are prepared with good quality (i.e. telephone quality speech with SNR usually above 20dB) data from the MALORCA project[10] which includes only ATCo speech. The data was collected from the Prague and Vienna airports and, thus, forms two separate sets with correspondingly German and Czech accented English. From the 'standard' MALORCA test sets (Srinivasamurthy et al., 2017) only utterances with the available surveillance information are selected. These utterances are both with and without callsigns.

NATS test set A non-public test set collected under the HAAWAII project[11] with the data coming from the London approach (airport). The amount of manually transcribed data available for testing is around 1h. This data is relatively high quality, similar to MALORCA.

DLR test set A non-public test set rich with waypoints data and collected by DLR[12] specifically for evaluation on rare and OOV entities. The data was collected through proof-of-concept exercises involving ATC utterances from ATCos to pilots. The test set has 52 minutes of audio data and 673 utterances containing 1157 commands[13] like *CONTACT, CONTACT_FREQUENCY, DIRECT_TO*. The data set includes 84 unique waypoints (="rare words") with their total occurrence of 443 times.[14]

[10] The Horizon 2020 SESAR project MALORCA (MAchine Learning Of speech Recognition models for Controller Assistance) is partly funded by SESAR Joint Undertaking (Grant Number 698824): https://www.malorca-project.de/wp/.

[11] Highly Automated Air traffic controller Workstations with Artificial Intelligence Integration, website: https://www.haawaii.de/wp/

[12] DLR – Deutsche Zentrum für Luft- und Raumfahrt.

[13] Command is a high-level concept that represents an ATC instruction (Chen et al., 2023).

[14] We do not include DLR test set to Table 2.4, because it mainly focuses on the presentation of

ATCO2 test sets The first test set (ATCO2-test-set-4h) contains 4 hours of ATC speech with manual transcripts and a subset with gold annotations for named-entity recognition (NER) (callsign, command, value). The second test set is a small publicly available test set (ATCO2-test-set-1h) is a one-hour subset from the original test set corpus.[15]

Each utterance in the ATCO2 test sets is provided with a list of callsigns to bias and with the ground truth callsign or NO_CALLSIGN if an utterance does not contain any. All biasing lists include about 10% of OOV words. All used sets are English data; an overview including the number of biasing entities is given in Table 2.4.

2.4.2. Datasets in other domains

2.4.2.1. Earnings datasets

Earnings21 as a test set. The public corpus Earnings21 (Rio et al., 2021) is a 39-hour speech dataset of earnings calls that features entity-rich speech from nine distinct financial sectors. The corpus is designed to benchmark ASR systems, with a particular focus on NER. It is provided with two biasing lists based on the NER: the *oracle* and the *distractor* lists[16] (Drexler Fox and Delworth, 2022). For our experiments, we use only the oracle list. The Earnings21 biasing lists contain both unigrams and word sequences, and we keep them together. The corpus has 44 recordings of the length ranging from less than 17 minutes to 1 hour and 34 minutes, with the average recording being about 54 minutes in length. For decoding, we split the audios into 3-minute long segments, following the approach from (Drexler Fox and Delworth, 2022).

Earnings22 as an adaptation data. As the Earnings21 dataset is a test set, for the adaptation data, we use the public Earnings22 dataset[17] from the same financial domain (Rio et al., 2022). The corpus consists of 119 hours of speech and 125 recordings. The main difference between the Earnings21 and Earnings22 data is that the Earnings22 corpus contains earnings calls from global companies sourced from 27 unique countries and covering several accents of English. The authors distinguish 7 linguistic regions: Spanish/Portuguese, Asian, English, Other

waypoints and not callsings as the other test sets.

[15] The *ATCO2-test-set-1h* subset is offered for free in the following website: https://www.atco2.org/data.

[16] Earnings21 biasing lists: https://github.com/revdotcom/speech-datasets/tree/main/earnings21/bias_lists

[17] Earnings22 website: https://github.com/revdotcom/speech-datasets/tree/main/earnings22

Table 2.5.: Train and adaptation sets in other domains (CV† — CommonVoice)

Dataset	Duration (hours)	Description
ATC	195	For dictionary and for LM training additional text data from public resources such as airlines names, airports, ICAO alphabet and way-points in Europe was used.
Earnings22	119	Contains 125 calls recordings and accents from 7 linguistic regions.
DefinedAI	472	299 hours in insurance and 173 hours in banking domains.
GigaSpeech	10,000	Subsets Giga-S of 250 hours and Giga-XL of 10K hours are used.
CV-EN	1,000	-
CV-DE	600	-
CV-FR	600	-
CV-ES	317	-
CV-CA	1,200	-
CV-IT	200	-

Romance, Germanic, Slavic, African.[18] The accent variability makes the Earnings22 data acoustically very diverse, which can be an issue when using the paired data for domain adaptation.

2.4.2.2. DefinedAI dataset

The DefinedAI dataset[19] is a non-public data collected from call center conversations in healthcare, banking, and insurance domains and manually annotated with NEs. The dataset is split into training, test, and development subsets of correspondingly 48, 6, and 3 hours and with equally distributed domain data. In addition to the mentioned subsets, 472 extra hours of DefinedAI are available in the insurance (299 hours) and banking (173 hours) domains. However, this data has different accents of English, Indian and US, and we use it only as text data.

2.4.2.3. GigaSpeech dataset

As Earnings21 and DefinedAI data have no or little training data and belong to specific domains, it was decided to use GigaSpeech corpus Chen et al. (2021) as training data. GigaSpeech is a multi-domain English speech recognition dataset containing

[18] Ordering of the linguistic regions is due to their proportions in the corpus: from higher to lower.
[19] DefinedAI website: https://www.defined.ai

Table 2.6.: Test sets in other domains with statistics on context information († *"N of utt."* — number of utterances with at least one named entity)

Test set	Total num of utterances	Duration (hours)	Biasing entities		
			unique	N of utt.†	OOV
Earnings21	18K	39	1'013	-	-
DefinedAI	2K	6	367	486	16
CV-EN	16K	27	6'921	6'442	169
CV-DE	16K	27	6'949	8'491	304
CV-FR	16K	26	6'035	7'486	225
CV-ES	15.5K	26	4'776	6'528	15
CV-CA	16.3K	28	2'108	2'607	-
CV-IT	15K	26	5'838	5'938	-

10,000 hours of high quality labelled audio collected from YouTube, Audiobooks, and podcasts.[20] We choose GigaSpeech data because (1) it is conversational speech that acoustically matches Earnings and DefinedAI data, (2) and it is a general purpose dataset that can be suboptimal for recognising speech from specific domains, thus it is a good baseline for further adaptation experiments. Our choice of training data aligns with prior work in Earnings21, where the authors also use the Gigaspeech dataset for training (Drexler Fox and Delworth, 2022).

GigaSpeech corpus can be used in subsets of different sizes: XS (10 hours), S (250 hours), M (1,000 hours), L (2,500 hours), and XL (10,000 hours). In our experiments, we use either an ASR model pretrained on GigaSpeech-XL, or GigaSpeech-S for training a small model from scratch.

2.4.2.4. CommonVoice dataset

CommonVoice is a diverse crowdsourced public speech dataset (Ardila et al., 2020). The CommonVoice dataset comprises several thousand hours of audio in more than 100 languages. For experimentation, we select six languages from CommonVoice-v11[21] which have sufficient data for training ASR and language models: Catalan (CA), English (EN), German (DE), French (FR), Spanish (ES), and Italian (IT).

We use the official train sets and report WERs on the official test sets (see Table 2.6 for further statistics). CommonVoice lacks gold NEs, requiring us to generate our own bias lists. Biasing lists for target CommonVoice subsets are automatically cre-

[20] GigaSpeech website for more details about the data: https://github.com/SpeechColab/GigaSpeech.

[21] CommonVoice-v11: cv-corpus-11.0-2022-09-21

ated to perform contextual biasing experiments. For this purpose, we use BERT models from HuggingFace (Wolf et al., 2020) fine-tuned on the named-entity recognition (NER) task for each language individually.[22] The following steps are included: (1) automatic text labelling with BERT, (2) NEs extraction from the BERT labels, (3) NEs lists filtering. In Table 2.6, one can see the statistics of NE lists per language where the size of lists with unique NEs varies from 2,108 to 6,949 which is rather long for contextual biasing.[23] The middle column in "Biasing entities" section of Table 2.6 shows the number of utterances per test set that contain at least one NE. This information gives an estimation of the proportion of NEs in the test sets: DE and FR sets have almost half of the utterances with NEs, at the same time, the CA set has only 17% of those.

2.5. Metrics

Evaluating ASR system performance, especially for contextualised models, demands a nuanced set of metrics. While WER remains a foundational measure, this thesis places particular emphasis on assessing the precise recognition of NEs, which are critical for practical applications. This section details the primary evaluation metrics employed, encompassing overall ASR accuracy, dedicated NE-specific performance indicators, and the computational efficiency of our proposed methods.

2.5.1. Word error rate

The standard evaluation method in continuous speech recognition is word error rate (WER). WER calculates the minimum number of mistakes produced by a system at the word level, relative to the number of words in the reference transcription:

$$WER = \frac{S + D + I}{N} \cdot 100\% = \frac{S + D + I}{S + D + C} \cdot 100\% \tag{2.12}$$

where S, D and I is the minimum number of substitution, deletion, and insertion operations required to transform the predictions of the system into the reference text, where N is the number of words in the reference text and C is the number of words correctly recognised by the system. Thus, WER is proportional to the

[22] EN: dslim/bert-base-NER-uncased; DE, ES, FR: Babelscape/wikineural-multilingual-ner (Armengol-Estapé et al., 2021); CA: projecte-aina/roberta-base-ca-cased-ner (Tedeschi et al., 2021).

[23] The ideal size of the biasing FST is significantly influenced by the data; according to Chen et al. (2019), performance started to decline when the number of contextual entities surpassed 1000.

correction cost. WER value can be greater than 100%, which can happen due to the large number of insertion errors.

Two variations of WER are character error rate (CER) and sentence error rate (SER). Their formula are the same as for the WER (Eq. 2.12) but CER measures error rate at the character level and SER measures error rate at the sentence (or utterance) level. CER is more flexible to catch the difference in recognition within a word. A minor error in the ending would be less punished than a completely different word; both cases would be equal errors for the WER measure. At the same time, CER is less successful in evaluating syntax and semantics errors that are more noticeable at the word level. SER show how many sentences are recognised incorrectly.

There is one more score derived from WER and defining word accuracy (the same formula can be applied to SER to get the accuracy estimation per utterance):

$$WAcc = 100\% - WER = \frac{C - I}{N} \cdot 100\% \tag{2.13}$$

To describe the performance of the ASR model, the value of WER is used. To compare the evaluated system with the baseline system or between each other, the absolute and relative improvement in the quality of recognition, i.e. the absolute and relative decrease in WER, can be determined by two formulas correspondingly:

$$\Delta WER = WER_1 - WER_2 \tag{2.14}$$

$$WERR = \frac{WER_1 - WER_2}{WER_1} \cdot 100\% \tag{2.15}$$

where WER_1 and WER_2 are WER scores of two compared systems.

For WER evaluation on Earnings21, we use the *fstalign tool*[24] provided by the authors of Rio et al. (2021) as the dataset references are in a special NLP-format.

2.5.2. Metrics for named entities (NEs)

In addition to the WER metric, we evaluate the performance of the model only on NEs. To calculate NE metrics, only strings containing NEs are taken into account.

[24] fstalign website: https://github.com/revdotcom/fstalign

NE WER (*NE-WER*) To calculate the WER for only NEs, an additional annotation file is needed for each test set that contains information on the ground truth NEs appearing in each utterance, e.g. in the format *"utterance-ID NEs"*. With this information, the WER is calculated only on the expected ground truth entities when reference and hypothesis utterances are aligned. The NE-WER metric works in the same way as the standard WER and includes false positive misrecognitions on the word level within each NE.

Unbiased WER (*U-WER*) Unbiased WER is the opposite of the NE-WER metric used to calculate WER on those words that are not biased Similar to NE-WER, an additional annotation file with the ground truth NEs per utterance is needed. When WER is calculated, the biased NE are ignored.

NE accuracy (*NE-A*) NE accuracy is calculated in a binary manner: "yes" – when the NE is completely recognised correctly, "no" – when at least one error occurs within the NE. In this case, however, false positive misrecognitions are not taken into account.

NE precision and recall As accuracy does not show the noise effect that can potentially be caused by a contextualization method, precision and recall evaluation can fill this gap.

For NE evaluation on Earnings21, we used only "PERSON" and "ORG" categories that have 5189 occurrences in the dataset.

2.5.3. Evaluation of decoding speed

To estimate the difference in the decoding time with VS without applying contextualisation, relative decoding time is measured with the *inverse real-time factor (RTFX)*, which is the ratio between the length of the processed audio and the decoding time:

$$RTFX = \frac{audio\_inferred(seconds)}{compute\_time(seconds)} \tag{2.16}$$

it is the inverse of the RTF (Real Time Factor) metric, such as $RTFX = 1/RTF$. The RTFX is measured with 1 GPU NVIDIA GeForce RTX 3090, with 2K clients.

3. Contextual knowledge for hybrid ASR

A hybrid ASR model consists of three independently prepared components: an acoustic model (AM), a pronunciation model (PM), and a language model (LM) (see more in Sec. 2.1.1). This enables all adjustments based on additional textual knowledge to be directly incorporated into the language model (LM). In a hybrid ASR model, the LM is typically represented as a finite-state transducer (FST) graph (Fig. 2.1), meaning most textual modifications are applied within this graph structure. We propose and examine three different methods for boosting contextual entities by dynamically modifying their weights and adding them, if necessary, to the weighted finite-state transducer (WFST). Besides the results achieved by applying these methods, we propose an implementation of contextual graph biasing directly on GPUs for the "on-the-fly" decoding.

3.1. Methods for contextual biasing with FST (for dynamic and static context)

We address the question of how to boost context entities by investigating three approaches: (1) adjusting entity weights in the decoding lattice, (2) LM modification with boosting entities in grammar FST (G.fst) and dynamic decoding graph composition, and (3) boosting target entities in the HCLG decoding graph (Eq. 2.3). Talking about the difference between the three approaches, we define them either as *dynamic* or *static* contextualisation. In the first method, modifications are applied to the ASR output during decoding and without model modification; thus, we consider it dynamic contextualisation. The last two methods involve the adaptation of the decoding graph itself and are considered static contextualisation. At the same time, in the second method, the integration of modified G.fst can be done in either offline or online (with "lookahead" composition) ways. We compare the methods with each other, as well as with their combination.

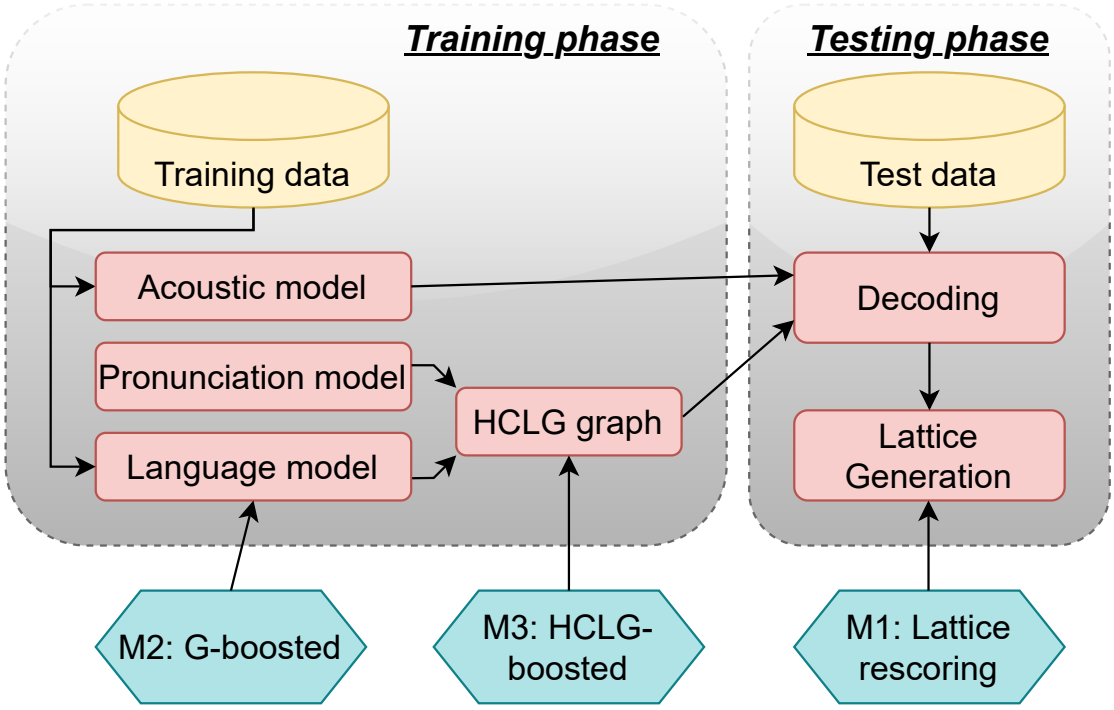

Figure 3.1.: Block diagram of the word-boosting ASR system.

The rationale for exploring these three approaches is threefold. First, these techniques do not require expert knowledge to implement, and the adaptation process can be automated on the end-user's side. Second, these approaches enhance existing ASR systems without requiring costly retraining and can be seamlessly integrated into the standard ASR pipeline, as illustrated in Fig. 3.1, making them well-suited for distribution as black-box utilities to clients who may lack expertise in the field. Third, the algorithms are lightweight, allowing for a relatively straightforward balance between word enhancement and overall system performance.

3.1.1. Biasing FST

In a hybrid ASR system, the different knowledge sources are represented as WFSTs combined in the decoding graph $HCLG$ by the operation of composition (see Eq. 2.3). However, the complete $HCLG$ graph is very large and complicated, and its modification, ideally in real time, is difficult. Thus, modifying lattices, i.e. FST graphs for the k-top hypotheses produced by decoding or modifying $G.fst$, i.e. LM graph, is easier and more flexible.

Contextual knowledge, i.e. a list of biasing words and word sequences, is used to create a small *biasing FST* graph with fixed bias values on the arcs that can be

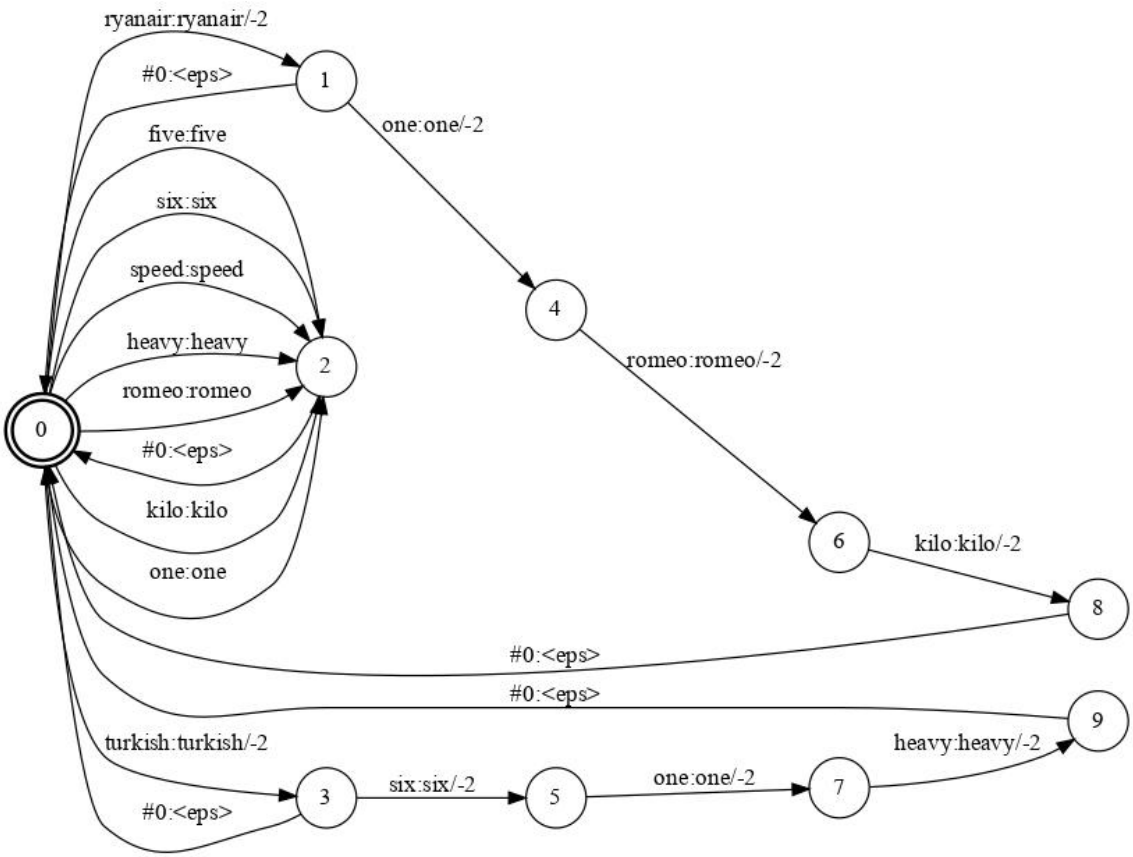

Figure 3.2.: A toy-example of biasing FST with callsigns **'ryanair one romeo kilo'** and **'turkish six one heavy'** and bias score "-2".

further integrated into another graph by the operation of composition. A 'toy' example of such *biasing* FST is shown in Fig. 3.2). For all operations on WFSTs we use the wrapper released as part of (Braun et al., 2021) in GitHub[25].

3.1.2. Method 1: 'lattice biasing'

The lattice biasing ("rescoring") is a method of dynamic contextualisation and a type of two-pass decoding. In the first pass, lower order (baseline) LM generates preliminary predictions in the lattice form; in the second pass, called 'rescoring', higher order LM adapts initial hypotheses according to context information. In our case, the higher order LM is a biasing FST that includes all target word entities with a discount factor. Word entities get boosted in the decoding lattices when their weights are adjusted in the composition, and thus, they become more probable

[25] https://github.com/idiap/icassp-oov-recognition

to appear in the final predictions:

$$Lattices' = Lattices \circ biasing\_FST \tag{3.1}$$

Contextual knowledge is added per each utterance as soon as the lattice is generated, converted to FST and composed per utterance with corresponding biasing callsigns. Therefore, if the context constantly changes, it can synchronise every single utterance. The only important condition for such an online contextualisation is that there should be enough time to generate a new biasing FST with the new context.

The main advantage of this method is its flexibility and low computational cost, as both lattice and biasing FST are usually very small and the operation of composition is computationally inexpensive. The main limitation is that only those n-grams that are present in the lattices can be boosted. As lattices are small graphs with k-top hypotheses, they do not include all possible combinations, but only those chosen by the model as the most probable. If the correct word sequence is pruned before the lattice is formed, lattice rescoring will have no impact on it.

3.1.3. Method 2: LM biasing with HCL ° G_boosted, i.e. 'G-boosting'

This rescoring method biases entities in an LM before decoding and, thus, addresses the limitation of the lattice rescoring approach that only n-grams present in the lattices can be boosted. Moreover, this method is able to add new word sequences if they are not in an LM. The method assumes that the words to be boosted are known in advance and included in the dictionary.

Target words or word sequences are enhanced by modifying the n-gram LM ($G.fst$) constructed from the training data before decoding. Firstly, the $G.fst$ is modified by iterating through the arcs in the baseline $G.fst$. Two objectives are: (1) adjusting the weights of existing arcs that match the target words by applying a constant discount of $-\log p$, and (2) creating arcs for new entities that are not originally included in the LM but need to be boosted with minimal weights. Secondly, a modified $G\_boosted.fst$, which assigns greater prominence to target words, is composed with HCL, allowing all necessary information from other ASR levels to be applied to all LM n-grams including newly added word sequences (Hori et al., 2007; Novak et al., 2012).

Let $\mathcal{B}_i$ be a word in a list of N rare words that need to be boosted in $G.fst$, where $i = 1, \ldots, N$. Also, let $\mathcal{A}^j_{\mathcal{B}_i}$ indicate the j^{th} arc in $G.fst$ that has the input label as $\mathcal{B}_i$, output label as $\mathcal{B}_i$, and the arc weight as $W\left(\mathcal{A}^j_{\mathcal{B}_i}\right)_{old}$. The boosting operation can be represented by Eq. 3.2.

$$W\left(\mathcal{A}^j_{\mathcal{B}_i}\right)_{new} = W\left(\mathcal{A}^j_{\mathcal{B}_i}\right)_{old} - \log(p) \tag{3.2}$$

The boosting operation is performed for all arcs representing the list of words to be boosted. When boosting a sequence of words, a new arc is added with a preset weight in the $G.fst$, if the arc for a particular word does not exist in the given context. The operation performed using Eq. 3.2 enhances the probability of the word (or sequence of words) being selected in the top hypothesis while decoding.

G-boosting can be performed either *offline* when the $HCL \circ G\_boosted.fst$ composition is done before the decoding, or *online* when HCL and $G\_boosted.fst$ are composed 'on-the-fly' with the Kaldi **lookahead** decoding[26].

Although this approach seems to us to be the most optimal when speaking about $HCLG$ pre-decoding rescoring, to use it, one should have access to HCL FST, which is not always possible.

3.1.4. Method 3: LM biasing with HCLG ° biasing_FST, i.e. 'HCLG-boosting'

Kocour et al. (2021) employs a rescoring approach that does not rely on lattices, instead focusing on enhancing the HCLG decoding graph before decoding. This method integrates the HCLG graph with a biasing FST by the operation of composition, allowing for the direct adjustment of target word weights within the decoding graph. Kocour et al. (2021) adjust weights for only single words (unigrams) in the HCLG graph because word-string boosting would considerably slow down the process. When using this method to boost word sequences, a key limitation remains: only the weights of existing sequences in the $HCLG$ graph can be adjusted, while new, previously unseen sequences cannot be added. For instance, if a 3-gram language model is used, only unigrams, bigrams, and trigrams already present in the model can be boosted, whereas longer n-grams cannot be incorporated.

This method is similar to the *G-boosting* method; both of them aim for biasing the LM but each with key differences. We describe both approaches; however, in our experiments, we will use either one or another method, choosing the more optimal

[26] Reference to the Kaldi lookahead decoding: https://kaldi-asr.org/doc/online2-wav-nnet3-latgen-faster_8cc.html (20.05.2025).

one depending on the experimental conditions. We categorise both of these methods as *static* contextualisation as it involves a change in the decoding $HCLG$ graph that makes them slower compared to the lattice biasing method. However, when the biasing of the G component of the graph is precomputed, contextualisation can be done "on-the-fly".

3.1.5. Online decoding on CPUs vs GPUs

Decoding on CPUs. Online decoding on CPUs (for example, Kaldi's *online2-tcp-nnet3-decode-faster*) is done in a similar way as offline decoding. Token and link structures are translated into OpenFst structures (Allauzen et al., 2007) that present an exact lattice (Povey et al., 2012). An exact lattice contains paths, which correspond to the candidates for ASR predictions, and stores precise costs and state-level alignments. The lattice structure enables flexible post-processing with different operations possible on lattices, such as acoustic scaling, computing the best, n-best, or oracle hypotheses, LM rescoring, lattice composition, etc. Although the lattice structure is convenient for operating with ASR output candidates before choosing the best ones, its implementation on GPUs would not be trivial. In the Kaldi GPU decoder, lattices are still created when an endpoint is reached to allow rich post-processing on CPUs (Chen et al., 2018).

Decoding on GPUs. There are many implementations of GPU decoders (Kim and Lane, 2014; Ivanov et al., 2016; Braun et al., 2020) and post-processing of their outputs with the CPU (Kim and Lane, 2014; Chen et al., 2018; Li et al., 2021b). For this study, we choose to work with the standard Kaldi GPU WFST decoder (Chen et al., 2018; Braun et al., 2020), as it is (1) open-source and (2) mostly built with Kaldi basic functions. The decoder[27] yields up to a 240x speedup over single-core CPU decoding (Braun et al., 2020). The addressed GPU decoding mechanism operates on two disparate asynchronous CUDA streams: one stream is responsible for running compute kernels, and the other one enables non-blocking device-to-host (D2H) memory copy of lattice tokens. This approach has fully parallelised decoding up to the outputs, yet its current implementation does not allow for any flexible rescoring. We proposed the rescoring approach inside the Kaldi GPU decoder, which is fully integrated into the parallelised decoding process, with no need for lattices. It allows asynchronous output of intermediate results during online decoding without interrupting the computational process. The decoder pipeline first transfers the models (i.e. acoustic model, and $HCLG$ graph) to the GPU, as depicted in Fig. 3.4.

[27] https://github.com/kaldi-asr/kaldi/tree/master/src/cudadecoder

The *HCLG* graph is represented by a special structure (*cuda-fst*) on the GPU. The FST structure is represented as a set of *compressed sparse rows* (CSRs) and additional metadata, which can be efficiently traversed with direct indexing (Braun et al., 2020).

3.1.6. Biasing lists

In our experiments on contextual biasing with hybrid models, most of the experiments are on the ATC datasets where biasing lists are callsigns registered by radar for every particular time spot or utterance (see Table 2.4). A non-ATC dataset is Earnings21 with a list of named entities (NE) as a biasing list (see Table 2.6).

The size of the biasing FST depends on the number of entities to boost. At the same time, with an increasing number of contextual entities, the biasing effect usually goes down. The optimal size of the biasing FST greatly depends on the data. In (Chen et al., 2019), the performance began to degrade when the number of contextual entities exceeded 1000. In our experiments, the largest FST has 1013 entities (Table 2.6).

3.2. Experiments on lattice rescoring and G-boosting for named entities

The section is partly based on the results described in Idiap research report publication (Nigmatulina et al., 2021) and paper (Zuluaga-Gomez et al., 2023a).

The section covers experimental setup and results when contextualisation is applied to bias target named entities in hybrid ASR models. The section describes experiments on contextual biasing with the *lattice biasing* (Sec. 3.1.2) and *G-boosting* (Sec. 3.1.3) methods on the ATC datasets.

3.2.1. Experimental setup

For the experiments, we used the Kaldi framework (Povey et al., 2011) to train the baseline acoustic model and to run decoding and rescoring experiments. The G-boosting experiments are done with the Kaldi *lookahead* decoding. The baseline results are obtained without applying any contextual biasing. We trained three

different models on the ATC data.

Model 1 CNN-TDNNF-ATC-1. For the first ATC model, the system follows the standard Kaldi recipe using MFCC and i-vectors features. Standard chain training is based on Lattice-Free Maximum Mutual Information (LF-MMI (Povey et al., 2016), which includes 3-fold speed perturbation and one third frame sub-sampling. The acoustic model (AM) is a CNN-TDNNF, which comprises a convolutional network and a factorised-TDNN.

For the training data, the following datasets are used: AIRBUS, ATCOSIM, HIWIRE, LDCATCC, MALORCA, N4 NATO, UWBATCC (see datasets description in Sec. 2.4.1.1 and Tab. 2.3) resulting into the total of 129 hours of transcribed speech data. First, the training databases are augmented by adding noises that match the LiveATC (see Sec. 2.4.1.2) audio channel. Afterwards, speed perturbation is applied, obtaining almost 1200 hours for training the AM.

The model dictionary consists of 28,410 words coming from diverse sources. The lexicon is composed of a word-list assembled from the transcripts of all available annotated train databases and from additional public resources including: (i) a list of airline designators for callsigns taken from Wikipedia: https://en.wikipedia.org/wiki/List_of_airline_codes; (ii) all five-letter waypoint names in Europe retrieved from the Traffic project, see https://pypi.org/project/traffic/; (iii) additional words, such as countries, cities, airports names, airplane models and brands, some ATC acronyms, ICAO alphabet, etc. The pronunciation of new words is obtained with Phonetisaurus G2P (Novak et al., 2016).

The model is further improved with 700 hours of semi-supervised data collected in LiveATC across various European airports. We applied a standard semi-supervised training approach in Kaldi, following the method outlined by Khonglah et al. (2020). This process involves training an initial seed model solely on supervised data, which is then used to generate lattices and best-path predictions for the unsupervised data (i.e., the 700-hour LiveATC dataset). The supervised and unsupervised lattices are subsequently combined to form new training archives. The LM is 3-gram trained on the same data as the acoustic model, supplemented with additional text data from publicly available sources, including airline names, airports, ICAO alphabet and way-points in Europe.

Model 2 CNN-TDNNF-ATC-2. This is the same model as the Model 1 described above but trained on more supervised data. In addition to the speech data used for Model 1, NATS, ISAVIA, SLO-96, SOL-97, and STARFISH datasets are also used resulting into the total of 190 hours of transcribed speech data (see datasets

Table 3.1.: Examples of improved callsign recognition (red — wrong; blue — correct)

System	Callsign
Baseline	hello sovar one nine lima
Biased	stobart two one nine lima
Baseline	ryanair four bye bye
Biased	ryanair four tango mike
Baseline	one six zero three five
Biased	airfrans six seven three five

description in Sec. 2.4.1.1 and Tab. 2.3). No noise augmentation is used. Speed perturbation is applied, obtaining around 570 hours of data for training. The model dictionary consists of 30,832 words.

Model 3 CNN-TDNNF-ATCO2. For the third ATC model, we used the same CNN-TDNNF training from the Kaldi framework with a convolutional network and a factorised-TDNN. The only difference is the training data, which in this case is 2,500 hours of automatically labelled data from ATCO2 corpus (see 2.4.1.1). The same ATCO2 data is used to train an LM.

3.2.2. Results

Table 3.1 gives some examples of improvement in which airline names and callsign word sequences are detected correctly compared to baseline predictions. In Table 3.2, the results of the experiments with the *lattice biasing* and *G-boosting* methods are presented in the LiveATC, Malorca Prague, and Malorca Vienna datasets. Results on LiveATC data are achieved with the CNN-TDNNF-ATC-1 model and are reported by Nigmatulina et al. (2021). The results on Malorca Prague and Vienna datasets are achieved with the CNN-TDNNF-ATC-2 model. The performance is shown with utterance WER, entity WER and accuracy of entity recognition. All entities in these experiments are callsigns registered by radar, i.e. *surveillance data*. A combination of both methods achieves the best callsign recognition results for all test sets. G-boosting and lattice rescoring together help to gain up to 4.6% of absolute (i.e. from 28.0 to 23.4%), or 16.5% of relative improvement of utterance WER, as well as up to 28.4% of absolute improvement in the accuracy of callsigns recognition (see Table 3.2). To better see the effect of biasing, in addition to evaluation on the LiveATC dataset, we separately calculated the performance only on those utterances from LiveATC that contain a callsign, in Table 3.2 it is marked as LiveATC†.

Table 3.2.: Results of the biasing experiments on LiveATC, Malorca Prague, and Malorca Vienna test sets. WER — word error rate; NE-WER — named entity WER; NE-A — accuracy of entities recognition; LiveATC† — LiveATC dataset but only the utterances where the target named entity is present. The top results per block are **highlighted in bold**; the oracle results are *highlighted in italic*.

Model	WER	NE-WER	NE-A
LiveATC			
baseline	30.7	29.2	50.5
lattice biasing (surveillance data)	29.5	23.9	60.8
lattice biasing (ground truth)	*27.8*	*17.8*	*72.8*
G-boosting lookahead (k=4)	28.1	19.5	66.2
Combined:			
G+lattice biasing (surveillance)	**27.2**	**16.0**	**71.3**
G+lattice biasing (ground truth)	*26.3*	*12.2*	*79.8*
LiveATC†			
baseline	28.0	28.5	41.3
lattice biasing (surveillance data)	24.9	19.3	63.4
lattice biasing (ground truth)	*24.7*	*18.0*	*67.6*
G-boosting lookahead (k=2)	24.8	18.5	63.4
Combined:			
G+lattice biasing (surveillance)	**23.4**	**14.3**	**69.7**
G+lattice biasing (ground truth)	*22.8*	*12.7*	*74.6*
Malorca Prague			
baseline	3.6	5.4	87.2
lattice biasing (surveillance data)	**3.2**	**3.9**	91.4
lattice biasing (ground truth)	*3.1*	*3.5*	*92.2*
G-boosting lookahead (k=4)	2.8	2.4	94.2
Combined:			
G+lattice biasing (surveillance)	2.7	**1.9**	**95.4**
G+lattice biasing (ground truth)	*2.6*	*1.8*	*96.1*
Malorca Vienna			
baseline	5.0	5.3	83.4
lattice biasing (surveillance data)	4.5	3.9	88.4
lattice biasing (ground truth)	*4.1*	*3.0*	*92.0*
G-boosting lookahead (k=4)	4.6	4.0	87.5
Combined:			
G+lattice biasing (surveillance)	**4.5**	**3.2**	**90.2**
G+lattice biasing (ground truth)	*3.9*	*2.0*	*94.5*

Number of distractors. The number of callsigns extracted from the radar and used to enhance the ASR system for each utterance can affect the effectiveness of the methods, as more biasing callsigns mean more distractors. Increasing the number of boosted n-grams raises the likelihood of the system selecting an incorrect one

during recognition. We investigated this effect by including test sets with different numbers of boosted n-grams, from 5 to 29 (see Table 2.4). Even with many boosted callsigns, the recognition accuracy goes up considerably compared to the baseline.

Noisy vs not-noisy data. Our methods also demonstrate consistent results for data of different qualities. The levels of noise in the recordings of LiveATC and Malorca test sets are very different, as well as the WERs achieved by their baseline systems. Nevertheless, we see considerable improvement for all test sets, and the general tendency remains the same.

Biasing unigrams VS n-grams. As a dynamic contextualisation method, lattice biasing is the most flexible and rapidly integrable approach among the three described methods. Additional experiments are conducted on ATCO2 data to compare lattice biasing only of unigrams (single words) VS lattice biasing of n-grams (sequences of words). Unigram lists are formed from the lists of biasing n-grams (named entities) and include all the words occurring in the entities. The main difference is that when biasing single words, these words are boosted not only inside of target entities (i.e. callsigns) but also in the context of other words. Thus, it can introduce noise, or "overboosting" and be less efficient than lattice biasing of word sequences. The results in Table 3.3 demonstrate that biasing n-grams consistently outperforms biasing unigrams: on the in-domain data, with a relative NE-WER improvement of 15.0% (n-grams) and 12.8% VS 13.8% and 9.1% (unigrams) for ATCO2-test-set-1h and ATCO2-test-set-4h, correspondingly; in the out-of-domain data, with a relative NE-WER improvement of 11.5% (n-grams) and 8.2% VS 5.2% and 4.6% (unigrams) for ATCO2-test-set-1h and ATCO2-test-set-4, correspondingly.[28]

Oracle results (ground truth). As mentioned above, surveillance data typically provides information on a limited number of callsigns registered in the airspace at a given time stamp. To compare our results with a setup that uses only the *ground truth* callsign for each utterance, we performed experiments using FSTs biased toward all registered callsigns as well as those biased to a single *ground truth* callsign (Table 3.2 and Table 3.3). Since real-world scenarios do not provide access to the correct callsign in advance, this serves as an oracle evaluation, representing an upper bound on performance. Although ground truth scores consistently outperform other setups, systems biased toward all surveillance data callsigns achieve results that remain relatively comparable.

[28] The results in Table 3.3 are from the publication (Zuluaga-Gomez et al., 2023a), where one of my contributions was describing and executing contextual biasing experiments and obtaining these results.

Table 3.3.: Results for *lattice biasing* experiment on ATCO2 corpora. Results are listed for the CNN-TDNNF model trained with either CNN-TDNNF-ATC-2 (supervised ATC data) or CNN-TDNNF-ATCO2 (ATCO2 data) model. The top results per block are **highlighted in bold**; the oracle results are *highlighted in italic*. WER — word error rate; NE-WER — named entity WER; NE-A — accuracy of entity recognition.

Lattice	**ATCO2-test-set-1h**			**ATCO2-test-set-4h**		
biasing	WER	NE-WER	NE-A	WER	NE-WER	NE-A
CNN-TDNNF-ATC-2 (supervised ATC data)						
Baseline	24.5	26.9	61.3	32.5	36.7	42.4
Unigrams	24.4	25.5	63.2	33.1	35.0	45.8
N-grams	**23.8**	**23.8**	**66.4**	**31.3**	**33.7**	**47.9**
Ground truth	*22.9*	*19.1*	*75.2*	*29.7*	*29.1*	*58.5*
CNN-TDNNF-ATCO2 (ATCO2 data)						
Baseline	17.9	16.7	70.5	24.9	24.2	62.0
Unigrams	18.3	14.4	73.8	25.6	22.0	65.9
N-grams	**17.3**	**14.2**	**74.3**	**24.3**	**21.1**	**66.5**
Ground truth	*15.9*	*6.5*	*89.4*	*22.2*	*12.5*	*83.9*

3.2.3. Summary

N-gram adaptation language modelling typically involves the interpolation of in-domain and out-of-domain LMs. The strong side of the proposed methods is their flexibility. They adapt current contextual information to improve the recognition of target n-grams without any additional in-domain LMs. This allows us to use new contextual data for each particular utterance.

We investigated two methods of integrating contextual data to improve the recognition of entities per utterance. In the first approach, n-grams are boosted by composing the decoded lattices (lattices generated in the first pass decoding and converted to WFSTs) and biasing WFSTs built with the dynamically changing contextual entities. The second approach involves modifying the weights for static contextual entities in the grammar component of the decoding graph (G) followed by dynamic lookahead decoding.

The *G-boosting* method often outperforms the *lattice biasing* method because (1) it adjusts entities' weights before the decoding and increases the probability that correct words (and entities) occur in the decoding lattices, and (2) it allows adding new words to $G.fst$. This method is convenient because it does not require an additional step of rescoring, and the generated hypotheses already include biased n-grams. The

main inconvenience of the G-boosting approach is increasing memory consumption. One has to create a new modified G.fst whenever new information is available. At the same time, the old G.fst is usually kept for the next updates. That doubles the memory usage. The lattice biasing method allows more flexibility, and its required additional memory equals the size of a biasing WFST.

The best results are achieved with the combination of both methods. The same trend of improvement is observed in all test sets and recordings of both lower and higher qualities, with the relative improvement of callsign recognition varying from 8.2% to 74.2% depending on the test set.

The introduction of contextual information considerably improves the recognition of callsigns and, therefore, the recognition of ATCo commands. As a noisy environment leading to lower recognition accuracy is often a reality in pilot-ATCo communication, the proposed methods and their combination will improve the recognition of key information in ATCo speech.

3.3. Experiments on lattice rescoring and G-boosting for rare entities, i.e. entities underrepresented in the training data

The section is based on the publications (Bhattacharjee et al., 2023, 2024). My contributions include: (1) describing the methods, and (2) participating in the design and execution of the biasing experiments.

Although the previous section demonstrated the overall effectiveness of our adaptation approaches, it is still uncertain whether this success extends to words that are particularly difficult to predict, either because they are never encountered during training (in both AM and LM) or appear only rarely. In this section, we present experiments specifically designed to address this question, i.e. to adapt deployed ASR systems to improve the recognition of words that are extremely rare or entirely absent from the training data. The experiments are carried out on ATC data. As rare words, special *airline designators* and *waypoint names* such as “balad” or “mabod”, which are waypoints for aircraft navigation in Austrian airspace, are used. Airport designators and waypoints are rare in the training data because they only appear in a specific region (see more in Sec. 2.3.2). At the same time, when used, they are highly informative and require high recognition accuracies. We examine

two approaches introduced above in Sec. 3.1: (1) lattice biasing and (2) G-boosting.

3.3.1. Experimental setup

To train the initial acoustic model and conduct decoding and rescoring experiments, as in the models above, we used the Kaldi framework. Two types of acoustic models are analysed.

Model 1 CNN-TDNNF-ATC. The first smaller hybrid-based CNN-TDNNF model was trained from scratch on ATC labelled data and described in 3.2.1 as Model 2 CNN-TDNNF-ATC-2.

Model 2 XLSR-LFMMI. The second model has the XLSR AM model (Conneau et al., 2020) pre-trained with a dataset as large as 56k hours of speech data and fine-tuned using the same data as in the CNN-TDNNF-ATC model and applying the approach described in (Vyas et al., 2021). Vyas et al. (2021) propose using the LF-MMI criterion (similar to hybrid-based ASR) for the supervised adaptation of the self-supervised pre-trained XLSR model (Conneau et al., 2020). This approach has been shown that it can outperform the CNN-TDNNF models trained with only the supervised data. A 3-gram language model trained on the same data as the acoustic model and supplemented with textual data from additional public resources such as airline names, airport information, ICAO alphabet, and European waypoints.

Typically, waypoints consist of a combination of two or three vowels along with a small number of consonants, i.e. “dexon” or “burok”. It is important to note that for the contextualisation methods investigated here, a critical assumption is that the newly introduced words do not contain any new phonemes.

As test sets, we use the ATCO2-test-set-1h (Sec. 2.4.1.2) and the DLR test set (Sec. 2.4.1.2). To evaluate the boosting performance on rare words, (1) 12 poorly recognised waypoints that appeared 191 times in the ATCO2-test-set-1h test set and (1) 84 poorly recognised waypoints that appeared 443 times in the DLR test set are selected.

3.3.2. Results

The performance of the lattice biasing and G-boosting methods on rare words (waypoints) is reported in Table 3.4. When target words are poorly presented in the training data, the two methods perform differently, and the results illustrate the difference in their nature. As rare words a priori have a low probability of being

chosen by the model, there is a low chance they appear in the decoding lattices, i.e. top k list for recognition. Thus, when only lattice biasing is applied to the baseline lattices, the rare words are not present there and cannot be boosted. We see smaller improvement with the lattice biasing compared to the baseline: the relative improvement in $F1$ when lattice biasing is applied is 3.6% for CNN-TDNNF and 13.6% for XLSR models for ATCO2-test-set-1h and 30.8% for CNN-TDNNF and 24.6% for XLSR models for DLR test set, respectively. When the G-boosting method is used, the rare words' weights are boosted directly in the LM (G.fst) and before decoding. Thus, the probability of them appearing in the short list during decoding also gets higher. The relative improvement of $F1$ on rare words compared to the baseline is 32.7% for CNN-TDNNF and 55.7% for XLSR models for ATCO2-test-set-1h and 430.8% for CNN-TDNNF and 31.1% for XLSR models for DLR test set, respectively.

Interesting results are observed when G-boosting and lattice biasing are combined. In Table 3.4, we can see that $F1$ for combined G-boosting and lattice biasing stays unchanged or very similar compared to the G-boosting method alone; however, *Recall* values are the best of all the experiments. It means that when rare words are promoted to the lattices by G-boosting and then additionally boosted by lattice biasing, the number of correctly recognised rare words increases. At the same time, lattice biasing adds noise by biasing other words towards the target ones and causes *Precision* to decrease. We can compare these results with the performance of the lattice biasing on unigrams VS n-grams described above in Sec. 3.2.2. Rare words here are always single words (i.e. unigrams), thus, we can assume that if we extend them to n-grams, e.g. by adding context of surrounding words, we can reduce the noise caused by over-boosting.

Effect of tuning the discount factor. Figure 3.3 illustrates the impact of adjusting the discount factor for G-boosting. The x-axis represents the values of the discount factor indicated as p, and the y-axis shows both the scaled general WER and the rare word recall. These WER and recall values are scaled within the range of $[0 \dots 1]$ to facilitate comparison. Low p values do not enhance the recall of rare words. However, as p gradually exceeds 1.0, the recall improves progressively while the WER decreases. When p exceeds ≈ 1.3, the WER increases significantly as boosted words are incorrectly recognised in place of many other ground truth words. These findings indicate that there is a threshold for boosting certain words beyond which the overall WER of the test set is negatively impacted.

Results with different acoustic models. Additionally, the effect of contextualisation methods is compared on two different acoustic models (AMs), i.e. CNN-

Table 3.4.: Performance of biasing on waypoints (as rare words) for ATCO2-test-set-1h and DLR test sets using both CNN-TDNNF and XLSR based acoustic models. The best results are **highlighted in boldface**.

G boosting	Lattice biasing	CNN-TDNNF				XLSR			
		WER	Rare word			WER	Rare word		
			Prec	Rec	F1		Prec	Rec	F1
ATCO2-test-set-1h									
-	-	26.8	**0.91**	0.39	0.55	17.5	**0.94**	0.42	0.58
✓	-	**26.2**	**0.91**	0.61	**0.73**	**16.4**	0.92	0.85	**0.88**
-	✓	26.8	0.86	0.42	0.57	18.1	0.89	0.52	0.66
✓	✓	**26.2**	0.8	**0.66**	**0.73**	17.1	0.83	**0.88**	0.85
DLR-test-set									
-	-	13.9	**1.0**	0.07	0.13	8.5	0.95	0.45	0.61
✓	-	9.6	0.93	0.55	0.69	**7.0**	0.93	0.7	0.8
-	✓	14.0	0.93	0.09	0.17	8.2	**0.97**	0.63	0.76
✓	✓	**9.4**	0.88	**0.58**	**0.7**	7.2	0.93	**0.73**	**0.82**

TDNNF and XLSR, where the XLSR AM has superior performance. The worst performance on rare entities is observed with the CNN-TDNNF model and on the DLR test set. Such low F1 scores might be explained by the fact that no DLR data was used in the training data. At the same time, the XLSR model achieves considerably better F1 scores for rare words with the same dataset. It can be explained by the better generalisation ability of XLSR AM. Overall, the results show that both contextual methods exhibit similar efficiency across the AMs.

3.3.3. Summary

The described experiment investigates how well biasing methods enhance the recognition of rare words. The experiment is conducted using data from the ATC domain, where the demand for accurate recognition of area-dependent words is critically important. We examine two techniques for improving the recognition of rare words, namely G-boosting and lattice biasing, and two different acoustic models, i.e. a smaller CNN-TDNNF model trained from scratch and a larger pre-trained XLSR model fine-tuned on the same dataset. While boosting generally exhibits superior performance with the larger acoustic model, the observed performance trends remain consistent across both model types.

G-boosting emerges as the top-performing approach for rare words boosting, and lattice biasing can further improve the recognition of rare words when combined with G-boosting. The choice of the discount factor in G-boosting significantly influences

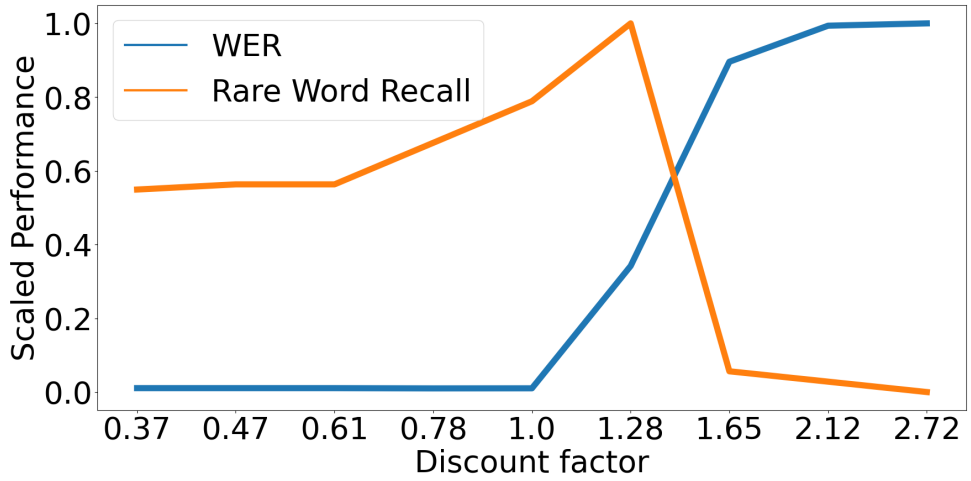

Figure 3.3.: Effect of discount factor on effective word boosting. The x-axis represents the values of the discount factor and the y-axis shows both the scaled general WER and the rare word recall.

the recognition of rare words, with very low values having little impact and high values causing over-prediction of boosted words.

3.4. Experiments on HCLG-boosting in the GPU-decoding

The section is based on the publication (Nigmatulina et al., 2023). The code is publicly released[29].

Real-time decoding can work on central processing units (CPUs), as well as on graphics processing units (GPUs) that can considerably accelerate decoding (Braun et al., 2020). Previous experiments have proven the viability of lattice rescoring for decoding and biasing LM weights in both offline and online CPU scenarios. However, these approaches overlook the potential for accelerating real-time transcription through GPU-based decoding. While GPUs are already being used for online ASR decoding, post-processing and rescoring on GPUs have not been properly investigated yet. In real-time GPU decoding, partial recognition hypotheses are produced without lattice generation, which makes the implementation of biasing more complex.

Above, methods of contextual biasing with FST are introduced in Sec. 3.1, and their

[29] https://github.com/idiap/contextual-biasing-on-gpus

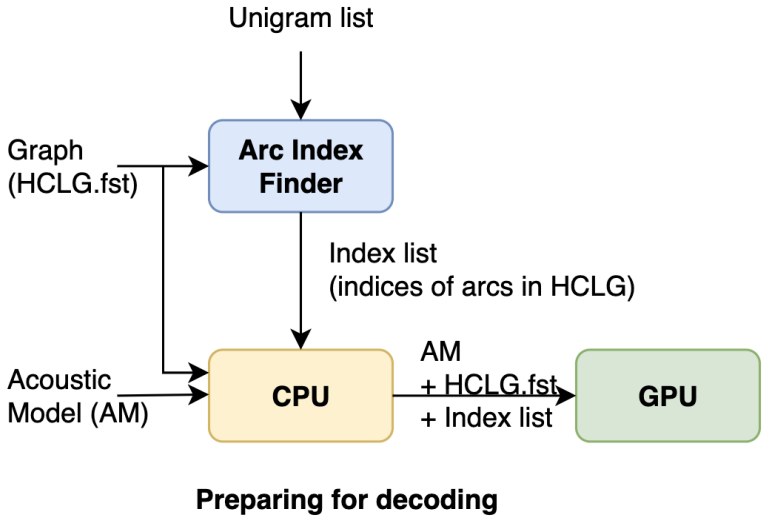

Figure 3.4.: Multiple stages involved in GPU decoding: Stage 1. Initially, the models are loaded and the necessary pre-processing is performed.

efficiency is proved in the experiments described in Sec. 3.2 and Sec. 3.3. The results demonstrate that rescoring with available contextual information can considerably improve ASR predictions and can be useful for many applications where prior information is available. For some applications, the efficiency and speed of contextual integration are crucial. For example, in the ATC domain, to assist pilot-ATCo conversations, online ASR is usually requested. While static contextual knowledge, such as a list of new waypoints, can be incorporated offline before inference, dynamically changing context — such as callsigns — must be gathered and integrated online during the inference process.

The section proposes an approach to integrate contextual biasing in real-time GPU decoding while exploiting the standard Kaldi GPU decoder. Our approach considerably accelerates contextual integration while biasing partial ASR predictions, and also permits dynamic context switching with flexible rescoring per speech segment directly on the GPU.

3.4.1. Implementation of rescoring in the GPU decoder

In our implementation of rescoring, we focus on three tasks: (1) unigrams boosting, (2) word sequence boosting, (3) dynamic update of contextual information, i.e. biasing FST. Another important aspect is to make sure that our implementation is optimal and that the decoding slowdown is minimised.

Rescoring without composition. In the GPU decoder, the *HCLG* graph is represented on GPUs with the special *cuda-fst* structure (see 3.1.5). When the

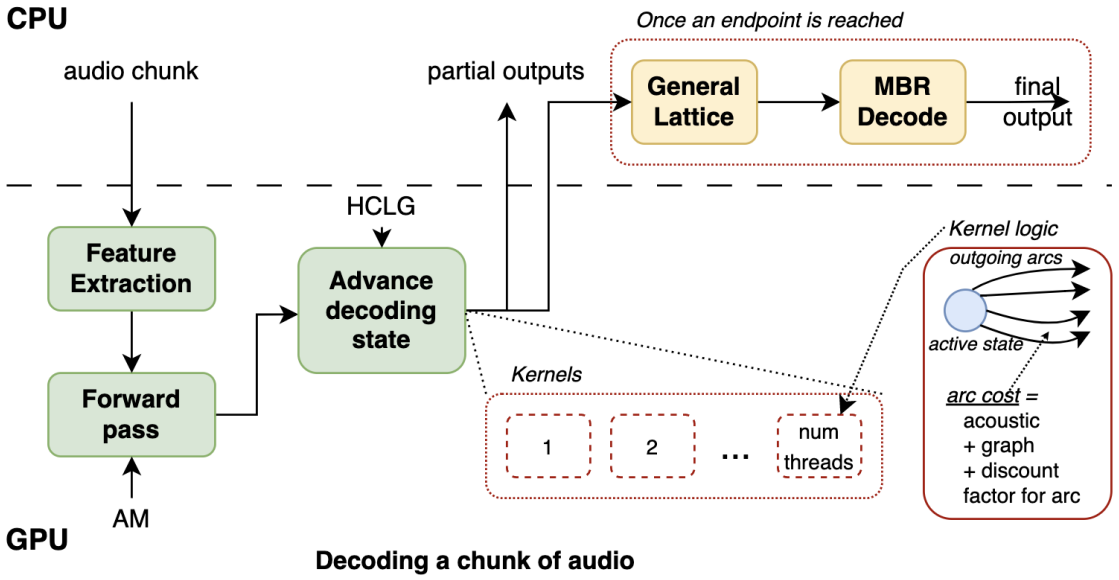

Figure 3.5.: Multiple stages involved in GPU decoding: Stage 2. When a chunk of audio is received, the decoding process follows with many steps split over both the CPU and the GPU.

graph is loaded, each outgoing arc is processed by its own thread with the *load balancing expand*, generating a number of candidate tokens. The *adaptive beam* is then adjusted and used to determine which candidates are added back to the main queue for further processing (Braun et al., 2020). GPU decoders can process multiple audio streams in parallel, and it is important to enable boosting specific to an audio stream. Pre-modifying the HCLG graph in advance will result in boosting all the streams.

As decoding on GPUs we cannot afford composition with HCL following the G boosting, it is not possible to add unknown word sequences to the graph. Thus, the n-gram set that we can boost is always limited by the LM, like in the *HCLG-boosting* method. With a decoding graph, the rescoring approach that would be the most suitable for our goal is the *HCLG-boosting* method or $HCLG \circ biasing\_FST$ composition (see Sec. 3.1.4). Instead of composition, weights in $HCLG$ could be adjusted iteratively, like for G in the *G-boosting* method, or by their indices (as shown in Fig. 3.4) that would allow flexible and less computationally expensive rescoring. The contextual *boosting* information for this method can be passed (1) as a *biasing* FST, or (2) as a list of entities we want to boost, where all words are replaced with their IDs from the symbol table.

Implementation. The $HCLG$ graph is represented as a set of *compressed sparse*

Algorithm 1: Pseudocode to find the arcs to be boosted given a word sequence ($w_1 w_2 ... w_k$)

```
Input  : fst: decoding graph; w_1 w_2 ... w_k: word sequence to boost
Output: arcs_indices: arcs that need to be boosted
arcs_indices = Set()
statesReached = GetStatesThatOutputToken(fst, w_1)
for t ← 2 to k do
    prevStatesReached = statesReached
    statesReached = set()
    for s_p ← prevStatesReached do
        // DepthFirstSearchSpecial takes first edge with w_t and then considers
          only edges that output ϵ until w_t is output
        reachableStates = DepthFirstSearchSpecial(fst, s_p, w_t)
        // now we know we can emit the next token
        statesReached.add(reachableStates)
        arcs_indices.Add(ArcsIndicesWithOutput(s_p, w_{t-1}))
    end
end
for (s_r, w_t) ← statesReached do
    arcs_indices.Add(ArcsIndicesWithOutput(s_r, w_t))
end
return arcs_indices
```

rows (CSRs) and additional metadata stored in memory. Before CSRs are generated, the arc information from the loaded $HCLG$ graph is temporarily kept in separate vectors: for input labels IDs, output labels IDs, next state IDs, and weights. This information is further copied to the GPUs (see Fig. 3.4). In order to rescore on GPUs, along with the decoding FST, we load the biasing FST, whose arc information is also saved in vectors but only for those arcs that should be boosted. Algorithm 1 gives the pseudocode of the procedure to determine which arcs to boost given a list of words. The algorithm is an extension of Depth First Search to find a sequence of arcs that would generate the desired sequence. We also consider the possibility that the first word in the sequence may start in the middle of the utterance. During decoding, if an arc index coincides with any token index saved from the current biasing FST, the arc's weight is adjusted by the discount factor. The boosting weight is the sum of the original arc's weight and the discount factor. The discount factor we use equals -2.0, which was empirically identified in the previous studies (Nigmatulina et al., 2021). The process is illustrated in Figure 3.5.

Boosting unigrams and word sequences. For every contextual biasing FST, we keep track of indices of the arcs to boost, which are identified by Algorithm 1. In the decoding kernels, this adds an extra cost of searching if the current thread is processing an arc to be boosted. To enable faster search, we store the indices sorted,

and perform a simple binary search. An additional complexity of $O(\log k)$ is added to each processing thread. This is negligible since k is significantly less than the total number of arcs in the graph. Compared to excessive memory requirements if storing separate decoding graphs for each context, we only store few 100s of integers.

As mentioned above, the size of biasing FST depends on the number of entities to boost. As with the increasing number of contextual entities, the biasing effect typically goes down, we assume that the size of biasing FST stays small not to exceed available memory. In our experiments, the largest FST has 1013 entities (for Earnings21; Table 2.6), and the number of *boosting* arcs we keep in memory per FST is significantly less than the total number of arcs since we aim to boost only the arcs related to the entities we are interested in.

Dynamic context update. To provide flexible biasing when the context is modified, we introduce the functionality of a dynamic switch between different biasing FSTs. We assume that certain context situations are anticipated in advance and the corresponding biasing FSTs are available before decoding starts. All expected biasing FSTs are pre-loaded and saved in separate vectors similar to how it is described in 3.4.1. Depending on the context a needed *biasing FST* indices are used to adjust the corresponding arc weights.

3.4.2. Experimental setup

For the experiments on contextual biasing on GPUs for hybrid ASR, we used two datasets — ATCO2 (with ATCO2-test-set-1h and ATCO2-test-set-4h test sets; Sec. 2.4.1.2) and Earnings21 (Sec. 2.4.2.1) — and two different ASR models trained with Kaldi framework for each of the datasets correspondingly.

Model 1 CNN-TDNNF-ATC. The first hybrid-based CNN-TDNNF model was trained from scratch on ATC labeled data and described in 3.2.1 as CNN-TDNNF-ATC-1.

Model 2 LSTM-TDNN. For the experiments on the Earnings21 set, we use the pre-trained chain LSTM-TDNN Kaldi model[30] with Gigaspeech-XL speech corpus (Chen et al., 2021).

Both ATCO2 and Earnings21 biasing lists contain word sequences, and for the unigram boosting, we converted them into lists of unique single words.

Relative decoding time. To demonstrate the lack of difference in the decoding

[30] https://kaldi-asr.org/models/m14

Table 3.5.: Contextual biasing with online CPU and GPU decoders on ATCO2-test-set-1h and ATCO2-test-set-4h. GT is a ground truth sequence; 'partial hypotheses' are real-time model predictions; NE-WER is a WER calculated for biased entities only.

	ATCO2-test-set-4h		**ATCO2-test-set-1h**	
	WER	**NE-WER**	**WER**	**NE-WER**
Online decoding on CPU				
No biasing	32.6	36.4	24.3	26.4
Biased unigrams (partial hypotheses)	34.6	35.4	25.0	25.7
Biased sequences (partial hypotheses)	32.5	34.3	24.0	24.2
Biased GT (partial hypotheses)	31.0	30.4	23.1	20.3
Online decoding on GPU				
No biasing	32.2	36.3	24.5	26.4
Biased unigrams (at endpoints)	34.1	35.7	25.0	25.4
Biased sequences (at endpoints)	31.2	34.4	24.0	24.1
Biased GT (at endpoints)	30.5	30.1	23.4	21.2
Biased unigrams (partial hypotheses)	33.2	35.5	24.7	25.1
Biased sequences (partial hypotheses)	32.9	34.6	24.9	25.3
Biased GT (partial hypotheses)	30.7	29.4	23.8	21.9

time with VS without biasing, we measured relative decoding time with the *inverse real-time factor (RTFX)*. The RTFX is measured with 1 GPU NVIDIA GeForce RTX 3090, with 2K clients, and on 81 minutes of Earnings21 data, which are split into 27 utterances, each 3 minutes in length.

3.4.3. Results

We compare WER results achieved (1) with contextual biasing on CPUs VS GPUs with lattice rescoring at *endpoints* VS dynamic biasing for *partial hypotheses* on GPUs (to see if there is performance degradation when rescoring is done without composition), (2) on GPUs with VS without applying contextual biasing (to see how the method improves the recognition). We do not compare the performance of our implementation to previous work, as there is no such results for ATCO2 sets, and biasing results on Earnings21 (Drexler Fox and Delworth, 2022) are achieved with an End-to-End model with a different biasing approach, which would be incomparable to our experiments. The results of the partial hypotheses biasing on GPUs are taken directly from the final server outputs, i.e. before it is sent for post-processing. Tables 3.5 and 3.6 report the results with utterance WER, and entities WER (NE-WER).[31] Comparing the performance of biasing with CPU decoder to the one on

[31] The baseline results on ATCO2 test sets slightly differ in Tab. 3.5 and Tab. 3.3. The difference is because in Tab. 3.3 the results are reported for the offline decoding and in Tab. 3.5 — for the

Table 3.6.: Contextual biasing with online CPU and GPU decoders on Earnings21 test set. 'Partial hypotheses' are real-time model predictions; NE-WER is a WER calculated for biased entities only; RTFX (Inverse Real Time Factor) measures the latency of a system.

	Earnings21		
	WER	**NE-WER**	**RTFX**
Online decoding on CPU			
No biasing	21.6	59.0	7.001
Biased sequences (partial hypotheses)	21.7	51.8	3.577
Online decoding on GPU			
No biasing	21.4	60.5	26.062
Biased sequences (at endpoints)	21.4	52.4	26.061
Biased sequences (partial hypotheses)	22.2	52.7	26.065

GPUs, the achieved improvement is almost the same, when rescored with lattices. Contextual biasing on GPUs always helps improve performance on the entities: e.g. NE-WER 52.4% instead of 60.5%, resulting in a relative improvement of 13.4% on Earnings21. The results in utterance WERs stay the same or slightly improve when sequences are boosted. Dynamic biasing of partial hypotheses on GPUs slightly differs from the other results, as weights are modified directly in the HCLG graph instead of decoder output candidates. Overall, the performance of dynamic biasing on GPUs shows similar improvement on entities over the baseline compared to the lattice composition approach.

The main advantages of our implementation are its speed and flexibility. Decoding on GPUs allows a considerable increase in speed compared to CPUs. Adding boosting inside the GPU decoder does not slow down the decoding process with the RTFX staying almost the same: 26.06. Pre-biasing the HCLG graph in advance would lead to similar improvements but does not allow dynamic context adaptation. The main limitation is that it is not possible to add unknown word sequences to the graph, and the n-gram set we can boost is always limited by the LM.

3.4.4. Summary

Motivated by the high effectiveness of contextual biasing on CPUs, we proposed an algorithm and its implementation for dynamic contextual biasing on GPUs for real-time hypotheses. Given the context words and word sequences as input, the method

online decoding performance.

adjusts target arc weights in the decoding graph in a distributed way and without lattice generation. This approach allows fast and flexible adaptation to a current context and is a step toward closer integration between inference and decoding.

3.5. Chapter summary

The contextualisation methods described in the chapter, i.e. (1) lattice biasing, (2) LM biasing with HCL and G-boosted composition, and (3) LM biasing with HCLG and biasing FST composition, allow dynamic integration of context to the hybrid ASR system. The main advantages of these methods are that (1) they require only text adaptation data, (2) no corpus is needed, as the context is a list of target entities, (3) context information is injected during inference or to the LM and involves no ASR retraining, (4) their integration is fast and can be used for streaming ASR.

In this chapter, we demonstrate the efficiency of these methods and the best application cases for each of them. For example, *lattice biasing* is the most flexible approach, as it operates during inference and performs the best when target words are present in the decoding lattice, i.e. well represented in the training data. Lattice biasing is a convenient method when light dynamic context integration is needed without modifying the LM. The *G-boosting* method allows adding new n-grams to the LM and biases the model's weights before the decoding lattice is built. This makes it suitable when dealing with OOV and rare words, as otherwise, underrepresented words get pruned at early stages of the decoding without reaching the top probable hypotheses. The *HCLG_boosting* method works the best when we do not have access to the components of the decoding graph, i.e. HCL, and we want to boost the predictions before the lattices are generated, for example, in the GPU decoding.

The results of the experiments show that contextualisation can benefit different use cases, including different domains, frequent and rare words recognition, and offline and online decoding. Another important advantage of the described methods is that their integration is easy to use, does not require a lot of data preparation and can be used as ASR system customisation by the end-users. The disadvantage of these methods is probably their applicability to the hybrid ASR models only, when nowadays ASR End-to-End models are developing and becoming the new SOTA.

4. Contextual knowledge for End-to-End ASR

Despite more straightforward unified training, End-to-End models are less flexible when talking about fast adjustment and tuning compared to hybrid models (see Sec. 2.1.2). In a hybrid model, additional textual knowledge can be directly incorporated into the LM. In standard End-to-End training, components are typically optimised jointly toward a single objective, and there is no separate LM. The necessity of adjusting the entire system to incorporate new textual knowledge depends on the specific architecture's approach to label context. For instance, CTC-based models do not explicitly model dependencies between output labels, behaving similarly to the acoustic models in hybrid NN-HMM systems; thus, they allow for relatively straightforward integration of external textual information. In contrast, for architectures that do model label context — such as Transducers or Attention-based Encoder-Decoders — the model learns an implicit Internal Language Model (ILM). While this often complicates modular updates, recent approaches for ILM estimation and compensation allow for the integration of external knowledge without necessarily retraining the entire system. Alternatively, contextual information can be introduced during decoding when final hypotheses are generated. This adjustment does not require model modification and is conceptually close to the *lattice biasing* method for hybrid ASR (see Sec. 3.1.2).

In this chapter, we examine two biasing methods for contextualisation of End-to-End models during inference, both of which are realised in decoding.

4.1. Methods of contextual biasing in End-to-End ASR

In the present section, we describe two methods for contextual biasing in End-to-End ASR used in our experiments: (1) *rescoring* and (2) *shallow fusion*. To understand how to inject contextual information during decoding, it is important to know the decoding search algorithm. Thus, first, we introduce basic decoding

search algorithms.

4.1.1. Search algorithms

The task of any decoding algorithm is to correctly interpret the probability distributions over tokens (characters, subwords, or words) outputted by an End-to-End ASR model. When generating the final output, a decoding *search algorithm* is typically used to select the best hypothesis from multiple model's suggestions. The most famous search algorithms are *greedy search* and *beam search*.

Greedy decoding. Greedy decoding is the simplest approach: at each time step, the most probable token is selected without considering alternative hypotheses or future steps. In CTC models, this means: (1) selecting the most probable token at each frame, (2) collapsing repeating tokens, and (3) removing blank symbols. Greedy decoding in CTC models is highly efficient because the predictions at each time step are conditionally independent. This allows the best path to be determined in parallel across the entire output sequence. While computationally efficient, greedy decoding often yields suboptimal results because it doesn't explore alternative paths. It is mostly used for fast inference or as a baseline.

Beam search. The beam search is the most widely used decoding strategy for End-to-End ASR that strikes a balance between exhaustive search and greedy decoding. Unlike greedy search, which selects the most probable token at each time step, beam search keeps track of multiple hypotheses (called beams) and explores multiple potential output sequences simultaneously (see Fig. 4.1). By maintaining multiple candidate sequences, the beam search reduces the risk of early incorrect decisions leading to suboptimal final outputs. It balances accuracy and computational cost by fixing the size of a list of the most probable hypotheses: a wider beam improves accuracy but increases decoding time and memory. The key steps of the beam search algorithm are depicted in Fig. 4.1 and are:

1. Initialise the beam with a start-of-sentence token.
2. At each time step, expand each hypothesis in the beam by appending the possible next tokens.
3. Calculate the score for each new hypothesis based on the log-probability from the model.
4. Prune the list, keeping only the top-**k** candidates (where **k** is the beam width).
5. This process repeats until an end-of-sequence token is reached or a maximum

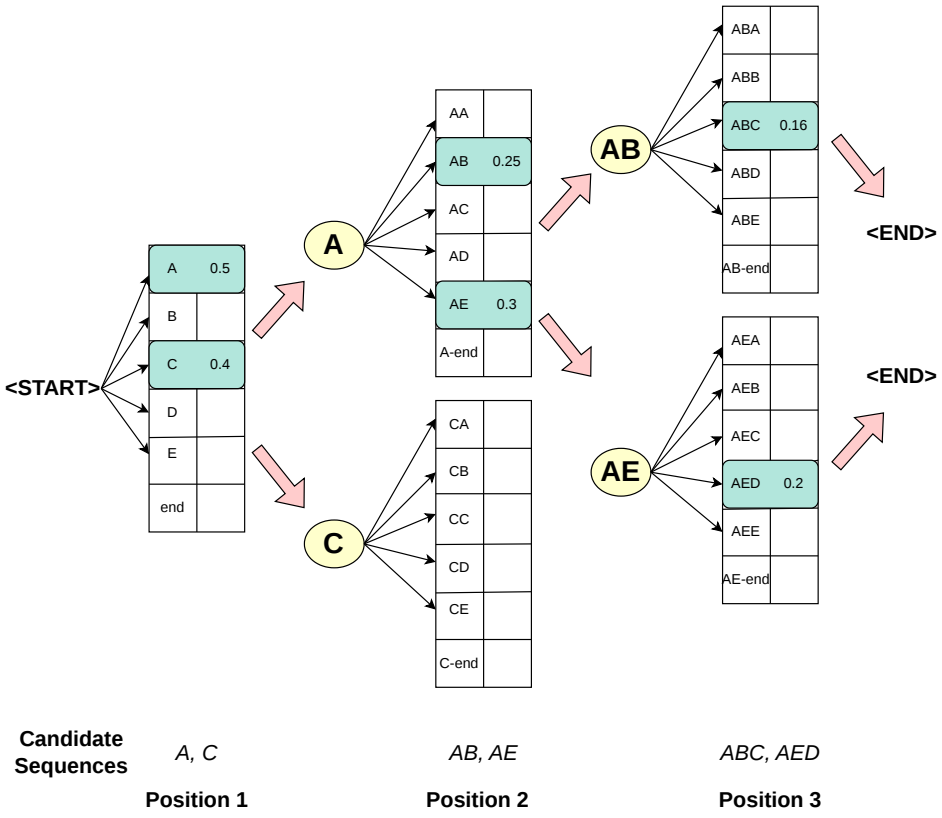

Figure 4.1.: Beam search example, with beam size **k=2**. *Position 1*: it starts with the $< START >$ token, defines probabilities for each word, and selects the two best units in that position (eg. **"A"** and **"C"**). *Position 2*: the algorithm is run twice to generate probabilities for sequences starting with **"A"** and **"C"**; then, the two highest probabilities are selected (eg. **"AB"** and **"AE"**) and the rest is discarded. *Position 3*: the process repeats again until the $< END >$ is reached and generates the final two best sequences **"ABC"** and **"AED"**.

output length is met.

There are many different implementations of the beam search depending (1) on the type of the model (CTC, attention-based, Transducer), (2) on the stopping criteria (to determine when to stop the search and select the final output), (3) on the control over the top-k hypotheses (for example, encouraging diversity among the top-k candidates by penalising sequences that are too similar to each other) (Vijayakumar et al., 2016; Drexler and Glass, 2019; Kasai et al., 2022).

A powerful extension of beam search is the incorporation of external language models (ELMs) to improve transcription quality, especially in low-resource or domain-specific settings.

Table 4.1.: Comparing rescoring and shallow fusion methods for ELM integration.

Feature	Rescoring (after pruning)	Shallow Fusion (before pruning)
When applied	After the first-pass decoding and pruning.	During first-pass decoding, at each step.
How It Works	An external language model (ELM) is applied to rescore a limited number of hypotheses (e.g., top-k beams or lattice paths).	The scores from the ASR model and an ELM are combined during beam search, influencing candidate selection in real time.
Integration Depth	Post-decoding step.	Real-time integration during decoding.
External LM	N-gram or neural LM.	Usually a neural LM.
Computational Load	Lower, since applied on pruned candidates.	Higher, because all candidates in the beam are scored at every decoding step.

4.1.2. Method 1: 'rescoring'

Rescoring is a method of contextualisation in two-pass decoding. Rather than directly choosing the highest-scoring hypothesis from the first decoding pass, the system generates multiple plausible candidates — usually in the form of lattices, n-best lists, or beams — and then re-evaluates them using a more refined scoring function or additional models. The advantages of rescoring are (1) operation on the word-level that allows straightforward use of n-gram LMs as ELM, and (2) low computational load, as it applies to the pruned list of candidates. The main disadvantage of rescoring is that being a post-processing step, it is applied only on the pruned hypotheses. Thus, it allows picking the correct prediction only if it has already been part of the n-best list.

4.1.3. Method 2: 'shallow fusion'

Shallow fusion (SF) can be seen as a dynamic rescoring strategy that happens during beam search decoding and before pruning. In practice, the SF refers to log-linear interpolation between the ASR outputs and a separately optimised LM at each step of the beam search (see an illustration of a SF at Fig. 4.2; Eq. 2.7; see more on SF in Section 2.2.2). Compared to rescoring, with the SF, the weights are updated during the beam search itself and, therefore, SF can also save the correct prefix from being pruned in earlier stages, leading to overall better performance than rescoring (see Table 4.1). In this section, we present four SF sub-methods, each differing in the type of contextual objects used — such as various forms of ELMs, keywords, or context-specific entities — and in the strategies employed to match these objects

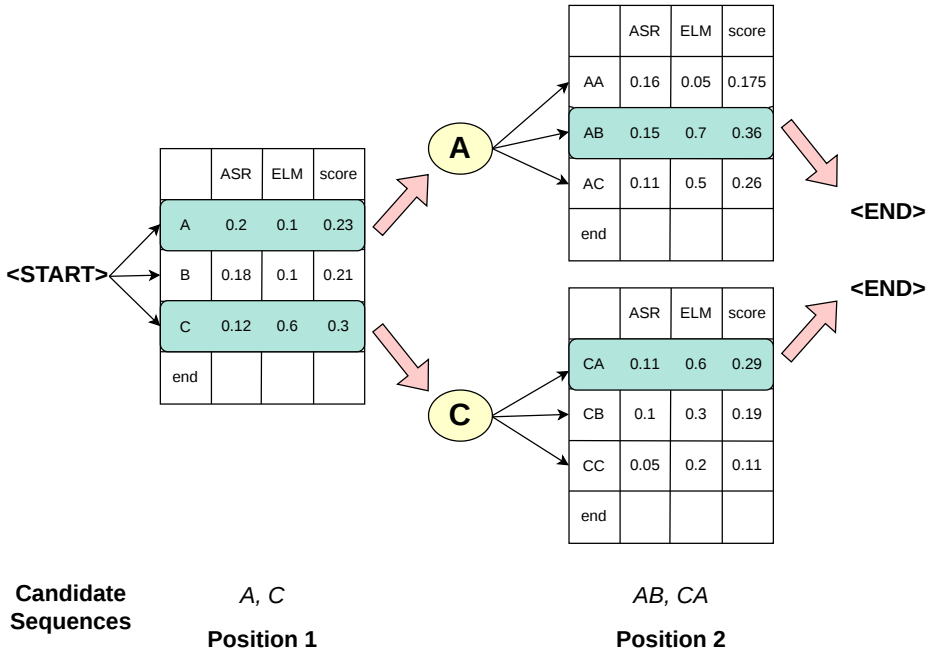

Figure 4.2.: An illustration of a shallow fusion algorithm. The final score for units at each decoding position is defined based on the combined scores from ASR and ELM: $y^* = \text{argmax} \log P_{ASR}(y|x) + \lambda \log P_{ELM}(y)$. In this example, with the $\lambda = 0.3$.

with ASR hypotheses. Methods 2.3 and 2.4 are proposed by us.

Method 2.1: Shallow fusion of subword-level ELMs. SF can be used to incorporate either ELM or a list of context biasing entities. Incorporated ELM can be (1) trained with the same data as used for the ASR training, and in this case it aims to improve the effect of the internal LM, or (2) trained with more additional data or with the data from the same domain as the test set, and in this case it works as the domain adaptation as well. Typically, ELMs are trained on the subword level (i.e. BPE) and, thus, if the tokeniser is the same, match the unit level of ASR models. This allows straightforward mapping between ASR and ELM units and adjustment of their scores at each step of beam search (Fig. 4.2).

Method 2.2: Shallow fusion of keywords and context entities. Biasing phrases or words are usually predefined and boosted when they are expected in a specific context. Rescoring of biasing entities differs from that with an ELM because the fusion happens only when the full biasing entity is detected. How to match the predefined contextual phrases with the ASR hypotheses being generated, so that their probability can be boosted via the LM score, refers to *string matching algorithms*.

String matching algorithms. Inside any string matching algorithm, there are two main steps: (1) during decoding, the hypothesis is compared to the biasing phrases at each time step; (2) if a partial match is found, the LM score is adjusted (usually boosted) to prefer completing the matched phrase. To efficiently support biasing in real-time decoding, there are various string matching algorithms (Aho and Corasick, 1975; Knuth et al., 1977; Wang et al., 2023) and different data structures to encode the context entities:

- Trie-based matching. Fast and memory-efficient approach when biasing phrases are stored in a prefix tree (Trie) (Knuth et al., 1977; Ney and Ortmanns, 2002a; Wang et al., 2010; Si et al., 2013; Le et al., 2021b,a). As decoding progresses, partial hypotheses are checked against the Trie (see an example of a trie at Fig. 4.3). If a prefix matches a biasing phrase, a boost is applied to increase its probability.
- Finite-State Transducer (FST). In hybrid systems or attention-based ASR, an FST representation of bias phrases may be composed with the LM or decoder graph (Mohri et al., 2002; Hori et al., 2007; Williams et al., 2018; Zhao et al., 2019). Supports flexible matching, including synonyms or phonetic variants.
- Levenshtein Automata / Edit Distance. This approach is used in more advanced setups where matching isn't strictly exact (Yang et al., 2024a). It works well for fuzzy matching, accounting for minor spelling or pronunciation variations, but is less efficient.

In our experiments, we use the Aho-Corasick (AC) algorithm, which uses Trie-based matching and is one of the most efficient and fastest methods for multiple pattern matching (Svete et al., 2022; Guo et al., 2023).

Aho-Corasick algorithm. The AC algorithm proposed by (Aho and Corasick, 1975) is a text multiple pattern string matching algorithm that operates in linear time: the search time equals $O(n + z)$, where n is the text length and z is the total number of matches. The algorithm utilizes three data structures to represent the search set, or a *transition diagram*: a trie, an output table, and a failure function (Fig. 4.3). The output table stores suffixes reachable from any node that corresponds to the target search strings. The failure function handles cases where some search strings are suffixes of others (Meyer, 1985). For example, if a trie contains the word **"CAN"** but not the word "**CAT**", a failure transition will backtrack **"CAT"** to the prefix **CA-** from **"CAN"** upon a mismatch. The algorithm is implemented as a finite state machine with backoff arcs that not only return to the root or a lower-order n-gram (e.g., from a 4-gram to a 3-gram) when a string end is reached

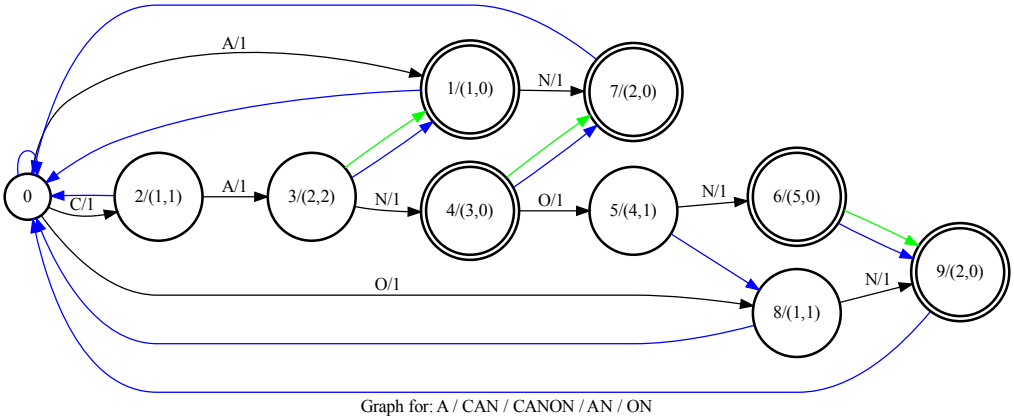

Figure 4.3.: Aho-Corasick trie: blue lines are fail arcs, green lines are output arcs.

but also perform *failure transitions* to backtrack upon a mismatch. This mechanism allows partial match detection while keeping the trie's structure sparse, improving efficiency by avoiding repeated prefix matches.[32]

Method 2.3: Shallow fusion of a word-level n-gram LM as a context trie.[33] Although SF is the most widely used method for contextualisation of ASR during decoding, fused ELMs are typically NNLMs trained at the subword level (Kannan et al., 2018). Introducing word-level ELMs is a less straightforward task, but it can benefit rescoring by leveraging richer word-level statistics and providing better semantic transparency for domain adaptation and biasing. To combine the statistics coming from character and word levels, fusion can be done with two character and word-level recurrent NNLMs at the same time (RNNLMs) Hori et al. (2017). However, NNLMs are unsuitable for online recognition and fast context adaptation due to their size. In previous studies, the n-gram word-level LM was adopted by converting subwords to words before rescoring (Tian et al., 2022). This conversion enables on-the-fly decoding; however, since a character-level sequence can correspond to multiple word sequences, the authors first needed to transform each partial character-level hypothesis into a word lattice during decoding, in order to account for all possible word-level interpretations.

Inspired by the application of string matching algorithms for SF of biasing words, we propose to integrate a *word-level n-gram LM* ("WL-ngram-LM") as a context

[32] Our work is based on the existing Aho-Corasick implementation available in the k2/icefall framework (https://github.com/k2-fsa/icefall/blob/master/icefall/context_graph.py).

[33] Methods 2.3 and 2.4 are introduced in the publication (Thorbecke et al., 2025).

prefix trie built with the AC algorithm. Recent advancements in E2E toolkits, such as RASR2 (Zhou et al., 2023), have demonstrated the effectiveness of integrating lexical prefix trees (tries) into Transducer decoding to handle large-vocabulary constraints. While RASR2 utilizes trie structures primarily for lexical search and decoding efficiency, our approach focuses on the SF of n-gram probabilities via a specialised Aho-Corasick (AC) trie. Specifically, we propose a 'WL-ngram-LM' context trie constructed directly from LM n-gram sequences, and their corresponding log probability scores are adapted as trie scores.

Since the ASR model produces hypotheses at the BPE level, we first convert the WL-ngram-LM n-grams into sequences of BPE units using SentencePiece[34]. During decoding, when a hypothesis string matches a stored n-gram in the AC trie, we incorporate the LM score as an additional cost into the log probability of the matched candidate. Algorithm 2 illustrates how the LM probability score (`bias_score`) is obtained before the context trie is created.

A key challenge in integrating an ARPA-formatted n-gram LM is determining how to convert its word-level scores into a bias score (bonus) that can be effectively combined with the ASR model's subword-level log-probability scores. An n-gram LM in ARPA format provides scores as log-base-10 probabilities, $s_w = \log_{10}(P_{LM}(w))$. Applying s_w directly as a bias cost would lead to poorly scaled biasing. However, through empirical evaluation on our development sets, we found that a non-linear transformation of the LM score yielded superior performance. We systematically compared several functions and discovered that the most effective bias score is derived by directly exponentiating the original log-base-10 score:

$$\text{bias\_score}(w) = \exp(s_w) = \exp(\log_{10}(P_{LM}(w))) \tag{4.1}$$

This transformation is mathematically equivalent to $P_{LM}(w)^{\frac{1}{\ln(10)}}$, which is approximately $P_{LM}(w)^{0.434}$. This effectively compresses the dynamic range of the original LM probabilities, emphasising higher probabilities less aggressively than a direct linear scaling, and sharply diminishing the influence of less probable ones. While the exponent $1/\ln(10)$ arises naturally from converting a log-base-10 probability into the argument of the natural exponential function, we acknowledge that this implicit compression factor could be considered an arbitrary choice and, in a more generalised framework, might be treated as a tunable hyperparameter γ (e.g., $P_{LM}(w)^{\gamma}$). However, for the scope of this work, this specific transformation proved robust and effective across our datasets.

[34] https://github.com/google/sentencepiece

Algorithm 2: Compute Context Scores from ARPA and Keyword Files

```
Function context_file, arpa_file, in_lm_cost, not_in_lm_cost:
    Initialize empty list contexts
    Initialize empty dictionary contexts_scores
    Initialize empty dictionary keywords
    foreach line in context_file do
        keywords[line.strip()] ← "not in the lm"
    end
    foreach line in arpa_file do
        Split line into [weight, ngram]
        Append ngram to contexts
        bias_score ← e^float(weight)
        if ngram in keywords then
            keywords[ngram] ← "in the lm"
            contexts_scores[ngram_ids] ← bias_score + in_lm_cost
        end
        else
            contexts_scores[ngram_ids] ← bias_score
        end
    end
    foreach (entity, label) in keywords.items() do
        if label is "not in the lm" then
            Append entity to contexts
            contexts_scores[ngram_ids] ← not_in_lm_cost
        end
    end
    context_trie ← ContextTrie(contexts_scores)
    context_trie.build(SentencePiece.encode(contexts))
```

Crucially, the bias_score(w) obtained through this transformation is a value in *probability space*, not log-probability space. When this score is applied, it is added to the log-probability of the ASR model's hypothesis. Specifically, we treat this bias_score(w) as a positive bonus that directly increases the overall log-likelihood of the matched word sequence during beam search. This unconventional direct addition of a probability-space value to a log-probability serves as an aggressive non-linear scaling, heavily rewarding high-probability n-grams while allowing for more nuanced integration than a simple linear addition in the log-domain.

A second implementation challenge is to apply this word-level bias_score(w), to a word w that has been tokenised into a sequence of subwords $(b_1, b_2, \ldots, b_k)$. We explored two strategies: (a) distributing the bonus evenly across all subwords (i.e., adding bias_score(w)/k at each subword step) and (b) applying the entire bonus at a single point. Our experiments showed that the latter approach is more effective. Specifically, the entire bonus is added to the hypothesis score upon the emission of the final subword, b_k, of the word w.

The proposed method allows light and on-the-fly fusion of a WL-ngram LM without the need for constructing word lattices.

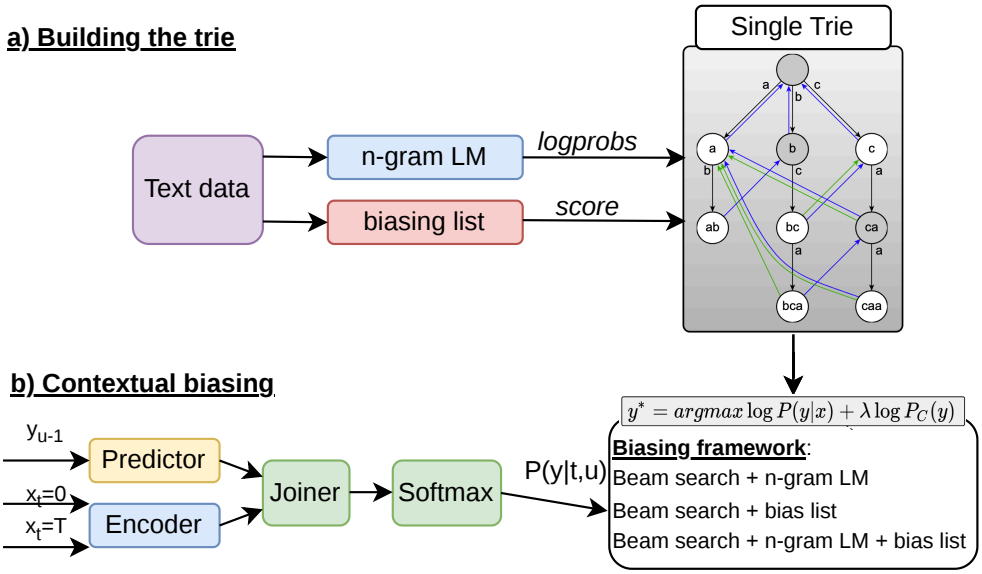

Figure 4.4.: Proposed biasing approaches at beam search time with Aho-Corasick string matching algorithm, inkl. biasing list and n-gram LM statistics.

Method 2.4: Shallow fusion with unifying n-gram LM and keyword biasing in a single trie. Keyword boosting performed at the subword level often introduces "noise" causing the overboost of other non-biasing words, which can lead to an overall decrease of WER. As an extension of the keyword biasing and WL-ngram-LM integration methods, we propose a novel implementation strategy that combines WL-ngram-LM n-grams and context entities in a *single biasing trie*, while maintaining traversal at the BPE level (see an illustration of the proposed method at Fig. 4.4). In the proposed approach, we aim to enhance keyword biasing while maintaining robust overall performance by leveraging *"near-context"* n-grams from an n-gram LM alongside the *"global-context"* entities.

During keyword biasing in decoding, a fixed positive cost is added to the log probability of a candidate in the hypothesis when it matches any of the context entities. To facilitate this, we employ a trie structure derived from the AC algorithm, which enables efficient lookup and biasing. This trie contains both LM n-grams and the keyword biasing information. The integration of keywords with LM n-grams is described in Algorithm 2. First, all n-grams from the LM are inserted into the trie with their respective scores. Then, for each keyword, a *bias score* is applied depending on whether the keyword's n-gram is present in the LM (*"inLM"*) or not (*"outLM"*). If the keyword n-gram is already present in the LM, its bias score is adjusted based on its associated LM score, while for keywords absent in the LM, a fixed bias score

is applied. The bias score is tuned on the development set, with different values for when the keyword n-gram is in the LM versus when it is not:

$$final\_bias\_score = \begin{cases} \alpha_{outLM} & \text{if NE is not in LM,} \\ exp(LMw) + \alpha_{inLM} & \text{if NE is in LM,} \\ exp(LMw) & \text{other words in LM.} \end{cases}$$

where LMw is the LM score, and α represents the fixed bias score.

4.2. Experiments on rescoring and shallow fusion with subword-level ELMs

In this section, we describe experiments on the integration of external subword-level LMs (ELMs) during decoding. We explore two methods introduced above, i.e. the *"rescoring"* method (Sec. 4.1.2) and the *"SF of subword-level ELMs"* method (Sec. 4.1.3).

4.2.1. Experimental setup

We conducted our decoding experiments with the code provided in the Icefall framework that contains recipes for training End-to-End Transducer models based on the K2 implementation of ASR-related algorithms.[35]

ASR model. For the model, we used the pretrained *Pruned-Stateless Transducer* ASR model (Ghodsi et al., 2020) with the Conformer encoder trained on 10k hours of GigaSpeech-XL data.[36]

Language models. Neural Network LMs (NNLM), i.e. Transformer-LM and RNN-LM, were trained with subword (BPE) units and with a mix of LibriSpeech (Panayotov et al., 2015), CommonVoice (Sec. 2.4.2.4), and GigaSpeech (Sec. 2.4.2.3) training data. The models were trained with the Icefall framework.

The evaluation was done on the GigaSpeech and DefinedAI dev and test sets (Sec. 2.4.2.2). In Icefall, several decoding methods are implemented. In addition to the greedy search, there are two implementations of the beam search: *fast-beam-search* and

[35] Icefall framework: https://github.com/k2-fsa/icefall; K2: https://github.com/k2-fsa (24.04.2025).

[36] Pruned-Stateless Transducer pre-trained on 10k hours of GigaSpeech-XL: https://huggingface.co/wgb14/icefall-asr-gigaspeech-pruned-transducer-stateless2/tree/main (24.04.2025).

modified-beam-search. In the fast-beam-search implementation, the decoding uses a lattice produced by the decoding graph and operates at the word level. This allows a straightforward use of word-level n-gram LMs. In the modified-beam-search, the decoding is done on the subword level (BPE), which makes it more difficult to use word-level LMs for the rescoring and SF. However, modified-beam-search consistently outperforms the fast-beam-search, so we report our results on SF with subword ELMs with the modified-beam-search decoding only.

The SF technique is implemented in modified-beam-search decoding to allow the integration of BPE-level NNLM and is available in two variants: with and without a *low-order density ratio method (LODR)* (Zheng et al., 2022) (the LODR method is described above in Sec. 2.2.2).

4.2.2. Results

The results with SF of ELMs for the Transducer ASR model are reported in Table 4.2. The baseline is the results when no ELM is used. The oracle results are obtained when the decoding is done with the correct hypothesis, in case it is among the top-k hypotheses.[37] Except the oracle results and the rescoring with WL-ngram LM that are computed with the fast-beam-search decoding, the rest experiments are done with the modified-beam-search decoding and with the $beam\_search = 20$[38]. SF consistently achieves better performance than the rescoring method, confirming the advantage of fusion before pruning. SF with LODR always outperforms the default SF. There is almost no difference between the SF with Transformer and RNN NNLMs. Overall, only marginal improvement is achieved by integrating ELMs into ASR decoding, with relative WER improvements of 1.9%, 5.4%, and 4.0% for GigaSpeech-test-set, DefinedAI-dev-set, and DefinedAI-test-set, respectively, and relative NE-A improvements of 4.0% and 7.4% for DefinedAI-dev-set and DefinedAI-test-set, respectively.

4.2.3. Summary

Integration of subword-level external NNLMs into the ASR decoding leads to a certain WER improvement. At the same time, this approach has the following drawbacks: (1) NNLMs training needs additional resources, including time, computational resources, and sufficient training text data, (2) ELM fusion considerably

[37] Oracle results are obtained with the default beam search equal 4.

[38] Decoding with a higher beam search does not bring noticeable improvement.

Table 4.2.: Results of SF experiments on the DefinedAI dataset reported in terms of WER and NE recognition accuracy (NE-A). † – the rescoring with word-level n-gram model and the oracle decoding are done with fast-beam-search decoding; the rest is obtained with the modified-beam-search decoding.

Decoding	**ELM**	**giga-test**	**DefinedAI-dev**		**DefinedAI-test**	
		WER	**WER**	**NE-A**	**WER**	**NE-A**
Baseline	-	10.5	9.7	67.9	10.3	68.0
Oracle†	-	*7.4*	*7.6*	*76.9*	*8.5*	*76.6*
Rescoring†	WL-3gram-LM	11.6	10.4	65.3	10.8	72.3
Rescoring	Transformer-LM	14.4	11.4	64.8	12.0	65.2
Rescoring	RNN-LM	14.5	9.5	68.4	10.3	69.6
Shallow fusion	Transformer-LM	10.4	9.5	68.8	10.1	70.2
Shallow fusion+LODR	Transformer-LM	**10.3**	**9.2**	**70.6**	**9.9**	**73.0**
Shallow fusion	RNN-LM	10.4	9.4	70.1	10.1	70.2
Shallow fusion+LODR	RNN-LM	**10.3**	**9.2**	70.4	**9.9**	**73.0**

slows down the decoding process, which makes it impossible to be used in online decoding. Looking for more efficient and light-weight ways of context integration into End-to-End ASR, we explore the other contextualisation methods introduced in Sec. 4.1.

4.3. Experiments on shallow fusion with word-level context (*Global-* and *Near-* Context in a Single Trie Pass)

The section is partly based on the publication (Thorbecke et al., 2025).

While the integration of subword-level ELMs in the previous section yielded some WER improvement, this approach comes with significant drawbacks: it requires extensive resources for training, it slows down decoding considerably, and the improvements are often marginal. These limitations highlight the need for a more efficient and lightweight approach to context integration, especially for on-the-fly and low-resource scenarios.

This section presents a new set of experiments focused on SF of word-level contextual objects, i.e. keyword and context entities biasing and WL-ngram-LM integration. While both count-based and neural LMs can theoretically utilise either subword or word-level token sets, our method specifically leverages word-level resources to

exploit the efficiency of an Aho-Corasick-based trie. This design is intended to overcome the computational bottlenecks often encountered with subword-level neural ELMs, where high-frequency token queries can significantly slow down inference. By utilising word-level representations within a unified trie-based search, we achieve a more precise and efficient application of context biases. By moving from complex, resource-intensive neural LMs to simpler, highly parallelisable word-level lookups, we aim to achieve a better trade-off between recognition performance and real-world computational practicality.

The contextual methods are described in more detail above and introduced as *"SF with keywords"* (Method 2.2), *"SF with WL-3gram-LM"* (Method 2.3), and *"SF with unified WL-3gram-LM and keywords"* (Method 2.4) in Sec. 4.1.3.

Most of the studies on SF are done with English language data. To explore the SF methods' effectiveness on languages other than English, we also run experiments on French, German, and Spanish data.

4.3.1. Experimental setup

Implementation. There is no Icefall implementation of SF with WL-ngram-LM, and we filled this spot with our implementation. The main conceptual question was how to go from the word level to the BPE units, as the model outputs its hypotheses on the BPE level, but the n-gram LM has its weights for words. To go from words to BPE units, it was decided to follow the ContextGraph implementation approach from Icefall. It was important to take into account that the n-gram LM weights (for example, in ARPA) are in the logarithmic scale, i.e. log probabilities, and they should be adjusted to match the model's predictions. The biases assigned to the predicted sequences are taken from the exponents of LM weights. Hence, all reported results are obtained using base-e exponentiation.

For the experiments with the *"SF with unified WL-3gram-LM and keywords"* method, where keyword biasing is combined with WL-3gram-LM, we conducted a grid search to determine the optimal values of the α hyperparameters, or fixed bias score (see Sec. 4.1.3). Specifically, we evaluated different bias scores for both cases: $inLM = \{0.5, 1.0, 1.5, 2.0\}$ and $outLM = \{0.5, 1.0, 1.5, 2.0\}$. Our results consistently showed that the combination of $inLM = 0.5$ and $outLM = 1.5$ achieved the best performance across all experimental settings.

We did an evaluation on four different datasets: DefinedAI (Sec. 2.4.2.2), Earnings21 (Sec. 2.4.2.1), CommonVoice (Sec. 2.4.2.4) to include the performance on different

languages, and ATCO2 (Sec. 2.4.1.2) to compare the performance to the results with contextualisation for hybrid models from Sec. 3. We used the Icefall framework to run the experiments, and for all experiments, we utilise their respective stateless version of the Zipformer-Transducer models (70M) (Ghodsi et al., 2020; Yao et al., 2024), with variations occurring solely in the decoding strategies. Accordingly, we benchmarked it against established methods for integrating external context during decoding. Notably, neither LODR nor ILM estimation was applied in these experiments, ensuring a clear comparison of the core biasing mechanisms.

Baselines. We implement three baseline methods, ranging from standard beam search to advanced SF techniques incorporating neural network language models (NNLMs) for contextual adaptation. • **BeamS**: Standard beam search decoding without any external contextual information (i.e., no SF). • **NNLM**: A conventional SF approach integrating a Transformer-based NNLM. • **BPE-ngram-LM**: A SF setup using a 5-gram LM trained at the BPE level.

Context Biasing with the context Trie (Aho-Corasick). We evaluate three variations of our proposed contextual adaptation method, leveraging the Aho–Corasick (AC) algorithm for efficient integration of external context. • **WL-3gram-LM**: SF with a 3-gram word-level LM, integrated via an AC-based trie for fast lookup and retrieval. • **KB**: Context biasing using an AC-based trie for keyword matching, without an explicit LM and with the bias score fixed to 2.0. • **WL-3gram-LM+KB**: a unified fusion approach that integrates a 3-gram word-level LM and a keyword biasing list through a single AC-based decoding trie pass.

ASR model 1 Zipformer-GigaSpeech-XL. The first model is the pre-trained Zipformer model trained on the GigaSpeech-XL dataset[39] for evaluation on the DefinedAI (Sec. 2.4.2.2) and Earnings21 (Sec. 2.4.2.1) test sets. This choice is justified because the Earnings21 dataset provides only test data, while for DefinedAI, we have access to limited training data (40 hours). Our choice of the model aligns with prior work on Earnings21, where the authors also used the Gigaspeech dataset for training (Drexler Fox and Delworth, 2022). This setup can be viewed as a domain adaptation scenario, given that both DefinedAI and Earnings21 consist of domain-specific data. During the decoding, we use a beam size of 4.

ASR model 2 Zipformer-CommonVoice (for CommonVoice models on English, German, French, and Spanish data). For experiments on CommonVoice, we train Zipformer models, with a stateless predictor (Ghodsi et al., 2020) and Zipformer encoder (Yao et al., 2024), for each language using the corresponding training

[39] Zipformer model pre-trained with 10k hours of GigaSpeech-XL data: yfyeung/icefall-asr-gigaspeech-zipformer-2023-10-17 (24.04.2025).

set. Training is performed from scratch using the latest Icefall Transducer recipe with its default hyperparameters. This includes the *ScaledAdam* optimizer (Kingma and Ba, 2014) and a learning rate scheduler with a 500-step warmup phase (Vaswani et al., 2017), followed by a decay phase determined by the total number of steps (7,500) and epochs (3.5) (Yao et al., 2024). The neural Transducer model is jointly optimized using an interpolated loss function, combining simple and pruned RNN-T loss (Kuang et al., 2022; Graves, 2012) with CTC loss (Graves et al., 2006) ($\lambda = 0.1$), according to:

$$\mathcal{L} = (1 - \lambda) \cdot \mathcal{L}_{RNNT} + \lambda \cdot \mathcal{L}_{CTC}. \tag{4.2}$$

We use an effective batch size of 600s with a gradient accumulation of 1, the peak learning rate is set to $lr = 5.0 \times 10^{-2}$, and each model is trained for 30 epochs on a single RTX 3090 GPU. During the decoding, we use a beam size of 4 that provides the best trade-off between the performance and the speed of decoding.

ASR model 3 Zipformer-ATCO2-1.5K. The third model is the same Zipformer model with a stateless predictor as Model 1 and Model 2, trained by us on the 1,5K hours of the ATCO2 data. For the training, we used a batch size of 1000s, the base learning rate is set to $lr = 4.5 \times 10^{-2}$, and each model is trained for 30 epochs on a single RTX 3090 GPU. During the decoding, we use a beam size of 4.

LM Training. For training the NNLM used in our baselines, we train Transformer-based (Vaswani et al., 2017) language models (LMs). Specifically, we use GigaSpeech-XL for DefinedAI and Earnings21, while for CommonVoice, we utilise the corresponding language-specific training sets. Each Transformer LM is trained for 10 epochs and consists of approximately 38M parameters.[40] Regarding the BPE-5gram-LM, we train a 5-gram BPE-based LM using the SRILM (Stolcke, 2002) toolkit. Similarly, for the word-level LMs required for the **SF+AhoC** experiments, we train 3-gram word-level LMs with SRILM. To construct these models, we use text data from the corresponding training sets for all test sets except Earnings21. For Earnings21, we instead use transcriptions from Earnings22 (Del Rio et al., 2022), a dataset from the same domain, ensuring domain consistency.

For NE-A results with KB and WL-3gramLM+KB experiments and the WER results with WL-3gramLM+KB experiments, we report statistical significance tests against the baseline (BeamS).

To further analyse the impact of word-level statistics on subword-level decoding,

[40] Transformer-LM recipe: https://github.com/k2-fsa/icefall/tree/master/icefall/transformer_lm

we evaluate the model's ability to recognise out-of-vocabulary (OOV) words on the CommonVoice dataset. The OOV counts for each dataset are summarised in Table 2.6. Notably, because the NEs for CommonVoice were automatically extracted directly from the test transcriptions, they are inherently included in the vocabulary and do not appear in the OOV list. The AC algorithm's capability to efficiently locate partial matches proves especially beneficial for improving OOV word recognition, reinforcing the advantage of our word-level LM integration over conventional subword-only approaches.

As fast and flexible context integration is critical in many practical scenarios, we include an estimate of decoding time using the *inverse real-time factor (RTFX)* (see definition in Sec. 2.5.3). RTFX is measured on the DefinedAI test set with one RTX 3090 GPU.

4.3.2. Results

The impact of word-level n-grams. To distinguish between out-of-domain and in-domain performance, we present results on DefinedAI and Earnings21 (Tab. 4.3) separately from CommonVoice (Tab. 4.4) and ATCO2 (Tab. 4.5). For both setups, the results of fusion n-gram LM with AC trie (i.e., WL-3gram-LM) lead to relative WER reduction w.r.t beam search alone: 3.8% for DefinedAI, 10.4% for Earnings21, 1.5%, 1.3%, 2%, and 2.6% for EN, DE, FR, and ES from CommonVoice, respectively. For the out-of-domain sets, SF of WL-3gram-LM improves the performance compared to the fusion with Transformer-LM (NNLM): i.e., from 10.2 to 10.0 for DefinedAI and from 14.9 to 12.9 for Earnings21. In addition to improved performance, training n-gram LM is fast and easy and can be done even with a small corpus (e.g., 40 hours in the case of DefinedAI), which will benefit low-resource scenarios.

However, the results are not consistent for the in-domain setup on the ATCO2 dataset: the relative WER reduction for the ATCO2-test-set-1h is 1.3%, but there is a 1.3% relative WER *increase* for ATCO2-test-set-4h.

Trade off between NE-Accuracy and WER. The biggest improvement on NEs is always achieved with the KB setup: 14.6% for DefinedAI and 6.7% for Earnings21 of relative improvement w.r.t. BeamS baseline (Tab. 4.3). However, the overall WER does not improve or even degrade, e.g., 16.7% WER in the case of Earnings21. Notably, the overall WER is improved when keyword biasing is combined with word-level n-gram LM (WL-3gram-LM + KB): relative improvement w.r.t. keyword biasing alone is by 3.8%, 21.6%, 3.6%, 3.9%, 1%, and 2.6% for DefinedAI, Earnings21, and CommonVoice EN, DE, FR, and ES, respectively. It is

Table 4.3.: Results of SF (bias score fixed to 2.0) experiments on DefinedAI-test and Earnings21 datasets reported in terms of WER (general WER, biased words WER (NE-WER), and NE recognition accuracy (NE-A). Best performance appears in **bold** font. †The result is statistically significant with $p < 0.07$.

Decoding type	Context source	WER (↓)	NE-WER (↓)	NE-A (↑)
DefinedAI-test				
BeamS	n/a	10.4	25.3	68.0
SF	NNLM	10.2	24.8	69.3
	BPE-5gram-LM	10.2	25.9	68.2
SF + AhoC	WL-3gram-LM	**10.0**	24.3	70.0
	Keyword boosting (KB)	10.4	**19.3**	**77.9**†
	WL-3gram-LM + KB	**10.0**†	22.8	73.3†
Earnings21				
Beam	n/a	14.4	49.2	59.5
SF	NNLM	14.9	49.9	58.5
	BPE-5gram-LM	16.8	48.4	59.0
SF + AhoC	WL-3gram-LM	**12.9**	45.3	61.7
	Keyword boosting (KB)	16.7	**38.4**	**63.5**†
	WL-3gram-LM + KB	13.1†	42.3	62.8†

important to note that the datasets vary in the number of named entities (NEs) and the proportion of utterances containing them (see Tab. 2.6). This variation explains the smaller impact of fusion on WER for the FR and ES sets, which contain the fewest NEs. Nevertheless, even in these cases, our approach maintains WER performance without degradation.

The ATCO2-test-set-1h dataset follows the same tendency as for the abovementioned test sets. The best overall WER performance is achieved when SF is done with KB and WL-3gram-LM together: 5.1% relative WER improvement w.r.t. the baseline and 1.3% relative WER improvement w.r.t. to the KB alone. Moreover, the best performance on unbiased words (U-WER) is also achieved with WL-3gram-LM: 1.9% relative WER improvement w.r.t. the baseline and 4.9% relative WER improvement w.r.t. to the KB alone. For the ATCO2-test-set-4h, however, the results are different again; there is almost no improvement in WER with the combined KB+WL-3gram-LM compared to KB. At the same time, the effect of WL-3gram-LM integration is noticeable when WER is calculated on the unbiased words only (*U-WER*): 3.7% of relative U-WER improvement on ATCO2-test-set-4h w.r.t. to the U-WER with KB only.

OOV performance. The incorporation of word-level statistics via WL-3gram-LM does not adversely affect OOV recognition at the subword level (as indicated by the

OOV accuracy (OOV-A) column in Table 4.4). Despite the KB yielding optimal performance, it may not be the most suitable option if overall WER is a concern. Similar to named entities, the WL-3gram-LM+KB configuration strikes an effective balance, preserving overall ASR performance while enhancing OOV accuracy by up to 51% for the German dataset. These outcomes substantiate that, even with the integration of word-level data into the rescoring process, fusion continues to advantage unseen words, given that the biasing occurs at the subword level.

Table 4.4.: Results of SF (bias score fixed to 2.0) experiments on 4 languages (English, German, French, Spanish) of CommonVoice reported in terms of WER, recognition accuracy of NEs (NE-A) and OOV (OOV-A) words. Recent Whisper performance is given as a reference (Radford et al., 2023). Best performance appears in **bold** font. †The result is statistically significant with $p < 0.07$.

Decoding type	Context source	WER (↓)	NE-A (↑)	OOV-A (↑)
CommonVoice-English				
Whisper-S (244M)/Whisper-M (769M)		14.5/11.2	-	-
BeamS	n/a	13.5	45.2	6.5
SF	NNLM	13.3	46.7	6.0
	BPE-5gram-LM	13.3	45.4	6.6
SF + AhoC	WL-3gram-LM	13.3	46.9	5.9
	Keyword boosting (KB)	13.7	**69.3**†	**36.0**
	WL-3gram-LM + KB	**13.2**†	59.3†	24.1
CommonVoice-German				
Whisper-S (244M)/Whisper-M (769M)		13.0/8.5	-	-
BeamS	n/a	7.7	55.5	19.4
SF	NNLM	7.6	55.5	18.8
	BPE-5gram-LM	7.6	55.8	20.4
SF + AhoC	WL-3gram-LM	7.6	57.0	19.7
	Keyword boosting (KB)	7.7	**79.0**†	**53.4**
	WL-3gram-LM + KB	**7.4**†	70.8†	39.7
CommonVoice-French				
Whisper-S (244M)/Whisper-M (769M)		22.7/16.0	-	-
BeamS	n/a	10.0	35.6	12.4
SF	NNLM	**9.8**	36.1	11.5
	BPE-5gram-LM	**9.8**	35.8	12.9
SF + AhoC	WL-3gram-LM	**9.8**	37.3	13.2
	Keyword boosting (KB)	9.9	**69.8**†	**53.9**
	WL-3gram-LM + KB	**9.8**†	53.6†	33.0
CommonVoice-Spanish				
Whisper-S (244M)/Whisper-M (769M)		10.3/6.9	-	-
BeamS	n/a	7.8	67.7	20.0
SF	NNLM	7.7	67.6	21.4
	BPE-5gram-LM	**7.6**	66.9	23.1
SF + AhoC	WL-n3ram-LM	**7.6**	66.9	18.8
	Keyword boosting (KB)	7.8	**86.8**†	**58.3**
	WL-3gram-LM + KB	**7.6**†	80.2†	35.7

RTFX. The RTFX results in Tab. 4.6 show that decoding with *WL-3gram-LM +*

Table 4.5.: Results of SF experiments on ATCO2-test-set-1h and ATCO2-test-set-4h datasets reported in terms of WER (general WER, biased words WER (NE-WER), unbiased words WER (U-WER)), and NE recognition accuracy (NE-A). Best performance appears in **bold** font.

Decoding type	Context source	WER (↓)	NE-WER (↓)	U-WER (↓)	NE-A (↑)
ATCO2-test-set-1h					
Beam	n/a	15.7	14.0	15.9	59.2
SF	BPE-5gram-LM	15.7	14.3	15.7	59.4
SF + AhoC	WL-3gram-LM	15.5	13.6	15.6	60.4
	Keyword boosting (KB)	15.1	**9.1**	16.4	**73.7**
	WL-3gram-LM + KB	**14.9**	11.1	**15.6**	66.7
ATCO2-test-set-4h					
Beam	n/a	23.6	21.8	22.7	46.7
SF	BPE-5gram-LM	**23.4**	22.0	**22.5**	46.7
SF + AhoC	WL-3gram-LM	23.9	21.6	22.8	47.5
	Keyword boosting (KB)	24.4	**17.9**	24.5	**58.9**
	WL-3gram-LM + KB	24.3	19.9	23.6	52.8

KB is only a bit slower compared to the BeamS approach alone, and thus it can be used on-the-fly: RTFX of the *WL-3gram-LM + KB* method is 6.0% lower than the *baseline* (BeamS) and there is no degradation w.r.t *keyword biasing* (KB). Finally, while the WER performance of BPE-5gram-LM and WL-3gram-LM is generally comparable across datasets, a key distinction lies in their computational overhead. WL-3gram-LM is 31% faster than BPE-5gram-LM, with an RTFX of 77.8 compared to 111.1, offering a significant advantage for production-level ASR solutions.

Table 4.6.: Ablation of decoding speed on the DefinedAI test set.

Decoding type	Context source	RTFX(↑)
BeamS	n/a	120.7
SF	BPE-5gram-LM	77.8
SF + AhoC	WL-3gram-LM	111.1
	Keyword boosting (KB)	113.5
	WL-3gram-LM + KB	117.3

4.3.3. Summary

In this section, we presented an efficient approach for integrating word-level context with a Transformer-Transducer ASR model using SF and an Aho-Corasick-based trie. Fusing word-level context, biasing does not happen at each decoding step but only when hypotheses match the target word entities. This approach enables more precise and efficient boosting. While it performs well for keyword biasing, trie-

based SF also proves to be applicable for integrating word-level n-gram LMs that are small in size and need only a little training data compared to the NNLMs. This method enables faster decoding than standard SF with NN-LM while maintaining competitive WER and real-time factor (RTF) efficiency. Additionally, we show that combining n-gram LMs with keyword biasing in a unified context graph improves, along with the NE accuracy, the WER for unbiased entities, including for OOV words. Experiments across four languages and four datasets confirm the effectiveness of our approach in both in-domain and out-of-domain scenarios. Finally, the results on CommonVoice demonstrate that SF proves to be useful not only when a list of target ground truth entities is available (often mixed with distractors) but also when the biasing entities are generated fully automatically.

Despite the improvement, SF methods are limited by the hypotheses produced by a default model before any biasing or contextualisation is applied. In the following section, we present experiments with the method aiming to overcome this limitation.

4.4. Chapter summary

In this chapter, we presented two methods for integrating textual context into End-to-End models: (1) rescoring and (2) shallow fusion. The latter includes four variations: (2.1) SF of subword-level ELMs, (2.2) SF of keywords and context entities, (2.3) SF of a WL-3gram-LM as a context trie, and (2.4) a unified approach combining the WL-3gram-LM and keyword biasing in a single trie. All methods aim to enhance ASR performance with external text information: either the overall WER with an ELM or recognition of target NEs with a biasing list. The methods do not require any specific contextual module added to the ASR architecture and operate flexibly during decoding. Although convenient in use, rescoring is strictly limited by the model's initial hypotheses (e.g., the N-best list). Consequently, while biasing can significantly improve NE-specific metrics, these methods often result in only marginal improvements in overall WER if the correct candidates are not sufficiently ranked during the first pass. Injecting contextual information directly into the encoder (e.g., via deep biasing) could more effectively influence the model's initial predictions. When the base hypotheses are already aligned with the relevant context, SF can achieve substantially higher performance.

5. Domain adaptation of End-to-End ASR

The objective of text-based adaptation is to enable the deployment of an ASR model in a target domain whose data distribution diverges from the training domain. Given that only textual samples from the target domain are available, we make the assumption that the difference exists solely in the marginal distribution over text, while the conditional distribution of speech given text remains unchanged.

In the previous Chapter 4, we saw that contextual text information can be (1) a list of biasing entities to improve the performance of keywords and key entities, and (2) a text corpus used to improve the internal LM (ILM) or for domain adaptation. Both types of textual data can be integrated during decoding and without modifying the base model, i.e. either *rescoring* (Sec. 4.1.3), or *shallow fusion* (SF) (Sec. 4.1.3). However, introducing an external LM (ELM) with rescoring or SF considerably slows down the inference time while bringing only marginal improvement. Moreover, ELM integration during decoding violates the main End-to-End principle of being a unified model. In this Chapter, we address other methods for leveraging text corpora data for End-to-End ASR adaptation.

The Chapter presents exploratory results that have not been published. We discuss language modelling in End-to-End ASR and examine why it presents unique challenges rather than being a straightforward task. We explore two approaches for the domain adaptation of the Transducer ASR model with text-only data: (1) predictor adaptation with a separate LM-layer and (2) model fine-tuning with the text and quasi-audio features generated from text.

5.1. Language model adaptation in End-to-End ASR

The success of an ASR model depends on how well the model can perform on unseen data. To improve it, one needs either to increase the generalisation ability of a model or use flexible model adaptation to the new data. Although End-to-End models are

strong in generalisation, they are often limited by the amount of paired data needed for End-to-End training.

LM adaptation in traditional vs. End-to-End ASR systems. A traditional hybrid ASR model has a cascaded structure and the acoustic model, which is responsible for learning a correspondence between sound and lexicon is trained separately from the language model, which defines the best possible sequences of words for the acoustic model outputs (see Sec. 2.1.1). As these two models are trained independently from each other, a lot of unlabeled text data can be leveraged for training LM.

An End-to-End ASR model is trained as a unified block, and it is common to compare the function of an encoder to an acoustic model and the function of a decoder in AED or a predictor in Transducer models to a language model. Yet, it is an approximation and decoder and predictor differ from a stand-alone LM (see more on ILM in Sec. 2.2.2). For example, in Transducer models, while the prediction network may resemble an LM — given that it receives the previously predicted token as input — it does not function as a true LM. The main difference compared to an LM is that the predictor's output must be non-linearly combined with the acoustic encoder's output to produce posteriors over the vocabulary, which also includes an additional *blank* token. Thus, adapting the prediction network with text-only data is not trivial. Similar to general LM adaptation, dynamic contextualisation also requires incorporating knowledge from external text sources. In both scenarios, End-to-End ASR models offer less flexibility in language modelling compared to hybrid ASR systems.

Moreover, to train an End-to-End architecture, only paired speech-text data can be used, and the amount of available training data for language modelling is limited to the labelled data only, instead of large-scale text corpora. When no audio data is available for the target domain, it is always easier to find more text data instead, as annotated audio-text data is expensive and time-consuming to obtain. Thus, domain adaptation with only text data is an attractive way to go.

In order to improve LM performance without using more parallel data, one first needs to understand how to transfer knowledge from the text data into End-to-End ASR.

Text pre-training. One way to leverage more text data is to pre-train the decoder. In the past few years, a few studies on ASR pre-training with unpaired data were published (Wang et al., 2020; Gao et al., 2021; Ao et al., 2022b; Arunkumar and Umesh, 2022; Gong et al., 2022). Gao et al. (2021) pre-train the transformer

decoder with the unpaired text data as a conditional LM with empty or artificial states to replace the real encoder hidden states. Pre-trained as a conditional LM, the decoder learns to generate grammatical text sequences before learning how to generate correct transcriptions. A similar study is done by Wang et al. (2020), where the authors use LAS (Chan et al., 2015) as the base model, additionally training the decoder on text data. While training the model's decoder on audio-text labelled data, the decoder is further trained on the text-only data. In the case of text-only examples, the encoder-generated context vectors are replaced with specialised vectors that explicitly indicate the absence of acoustic input. This guides the decoder layers to rely exclusively on internal model representations rather than audio-based contextual information. The context vector can be of two types: (1) constant, where all the values are set to zeros, or (2) learnable, which is shared for different frames, and its weights are learned using the ASR objective function. In contrast to other methods, which use text-only data to improve ASR performance, these methods are one-pass recognition models and do not introduce extra components like text-to-speech (TTS) or text-to-encoder (TTE).

An extension of the pre-trained decoder method is a joint pre-training of the encoder and decoder parts. For ASR, the joint encoder-decoder self-supervised pre-training has been recently proposed by Arunkumar and Umesh (2022). The idea is to use the SSL pre-training (i.e. HuBERT) combined and jointly pre-trained with the decoder. Widely used at the moment, SSL models provide powerful short-time representations of the speech signal, and the decoder on top of an SSL model helps it to learn an acoustic unit-based language model. The decoder, known for its ability to generate sequences, attends to the SSL encoder outputs through source attention like in any other Transformer encoder-decoder model. Targets for the decoder are attained from the discovered target cluster IDs of all the encoder input frames, both masked and unmasked (HuBERT architecture).

Bai et al. (2021) propose to integrate the knowledge from an ELM into the AED through teacher-student learning. The LM, as a 'teacher' model, provides a probability distribution corresponding to each token in the transcription ('soft labels'). Then, the Kullback-Leibler divergence (KLD) between the decoder of the AED model and the LM is minimised to train the 'student' AED model.

While pre-training allows the model to benefit from additional data and enhances its overall performance, determining how to effectively adapt and finetune an already trained model remains a challenge.

Internal Language Model (ILM) adaptation. In Sec. 2.2.2, we have defined an ILM that is trained together with the whole End-to-End ASR model. Meng

et al. (2021a) proposed to directly adapt an ILM with text-only data. To allow the acoustically-conditioned LM of an End-to-End model to behave like a standalone ILM, Meng et al. (2021b) propose the internal language model training (ILMT). Through ILMT, the standard End-to-End loss and an ILM loss are jointly minimised. The standard End-to-End loss is the summation of the negative log posteriors of the reference token sequences over the training corpus and is formulated in Eq. 2.5. The ILM loss of an End-to-End model is the summation of negative log probabilities of the ILM over training transcription corpus and is conditioned only on the parameters of the decoding part (θ_{dec}).[41]

During the training with the ILMT loss, the internal End-to-End LM is learned to behave like a standalone LM while maintaining its capability to collaborate with the encoder to compute the correct End-to-End model probabilities. The performance can be further improved with the internal language model adaptation (ILMA) proposed by Meng et al. (2021a). ILMA enables localised adaptation by applying a text corpus solely to the ILM, while keeping the End-to-End ASR component intact. Although this method works to some extent, the ILM is only an approximation of a stand-alone ELM, and the ILM loss is still conditioned on the decoder part of the model.

Factorised language modelling. As one of the main differences between a standalone LM and a Transducer ILM is the necessity to model a blank symbol by the predictor, the factorised neural transducer (FNT) architecture solves it by factorising the blank symbol and vocabulary prediction (Variani et al., 2020; Chen et al., 2022; Zhao et al., 2023). During training, two predictors are trained — blank-predictor and vocabulary-predictor — where the vocabulary-predictor is explicitly optimised towards a standard neural LM loss. As a result, the vocabulary-predictor can be further adapted with a text corpus in the same way as a stand-alone LM. A model with a factorised predictor, when adapted to in-domain data, generally achieves better performance on in-domain test sets compared to a non-factorised transducer model. However, enabling such adaptation requires the model to be trained in a specific manner from the outset, which may negatively affect its performance on general-domain data.

Another solution is proposed by Pylkkönen et al. (2021), where a separate neural network LM (NN-LM) layer is attached to the prediction network within a transducer architecture. The NN-LM layer is specifically designed to process text-only

[41] Here, the formula of the ILM loss is kept in a general form and θ_{dec} is used to define either the parameters of the decoder ((θ_{pred}) for AED model or the parameters of the prediction and joint networks (θ_{pred}, θ_{joint}) for RNN-T depending on the model.

inputs and is trained "on top" of the predictor with the standard autoregressive LM loss, i.e. cross-entropy loss. When NN-LM is trained, the remainder of transducer parts, including the predictor, are frozen. During the adaptation with text, only the prediction network is updated through the NN-LM layer and LM loss, when the NN-LM itself is frozen. This approach can be a quick solution for domain adaptation without re-training the main ASR model and when neither annotated data nor a text-to-speech (TTS) system is available.

5.1.1. Experiments

As the predictor is an approximation of an LM, the goal of the first experiment is to see how much improvement in the in-domain test set can be achieved when only the predictor network is updated during text fine-tuning. The advantage of such an approach is that the rest of the model stays unchanged and only the predictor is trained.

The experiments described in this Section are based on the method proposed by Pylkkönen et al. (2021) and introduced above. For the experiments, we use the GigaSpeech-S subset of 250 hours as the training data, the Earnings21 (39 hours) as a test set, and the Earnings22 (119 hours) text data as an adaptation set. With the Icefall framework 4.2.1, we trained two Conformer-Transducer models.

Model 1. The first model is a Conformer-Transducer model with a stateless predictor. The stateless predictor has no recurrent connections and, by default in Icefall, processes only 2 BPE tokens before. The stateless predictor is the default predictor used in the transducer models in Icefall.

Model 2. The second model is the same Conformer-Transducer model but with an LSTM predictor. The LSTM predictor has one recurrent LSTM layer that, due to the recurrent connections, allows better language modelling.

Both models are first trained with GigaSpeech-S training set for 15 epochs. Then, all parts of the model are frozen and a single linear layer is trained on top of the predictor with cross-entropy loss and only with the transcriptions from GigaSpeech-S. When NN-LM layer is trained, the predictor of the model is adapted with Earnings22 text data, while other parts of the model and the NN-LM layer stay frozen.

Table 5.1.: Results with Conformer-Transducer model with stateless and LSTM predictor and with NN-LM layer adaptation.

Model	**Adapted**	**Giga-Test**	**Earnings21**	
		WER	WER	NE-WER
Transducer-stateless	no	16.5	26.9	77.7
Transducer-LSTM	no	21.2	35.4	82.6
Transducer-LSTM	yes	21.3	33.8	79.9

5.1.2. Results

In the experiments done with the stateless predictor and Pruned-Stateless Transducer, we saw no effect of adaptation. It can be explained by the nature of the stateless predictor that can see only 2 symbols and cannot learn much from the adaptation.

Based on these observations and following (Pylkkönen et al., 2021), we replaced the stateless predictor with the LSTM predictor. LSTM predictor can learn longer text contents and can be adapted. With the LSTM predictor, we can see only little effect of adaptation with the LM-layer: WER on the in-domain test set improves from 35.4% to 33.8% which is the relative improvement of only 4.5%.

At the same time, the overall WER goes up after replacing the stateless predictor with the LSTM one. The reason the LSTM predictor might not work well in this particular Transducer architecture could be that it doesn't give good enough gradients to train the encoder. Thus, the results show that updating only the predictor weights and keeping the rest of the model unchanged is not enough to make the model perform well on the new domain.

5.1.3. Summary

Several strategies for incorporating text-only data into End-to-End systems are examined. These include decoder pre-training on unpaired text, teacher-student learning from external LMs, internal LM training and adaptation to strengthen the model's linguistic knowledge, and factorised architectures, which decouple blank and vocabulary prediction to enable standard LM-style adaptation. Additionally, methods like adding a NN-LM layer on top of the predictor — trained only on text data—are evaluated as lightweight adaptation techniques that avoid retraining the full model.

Experimental results using Conformer-Transducer models on domain-specific data (Earnings21/22) show that while adapting only the predictor with an NN-LM layer leads to modest improvements (e.g., WER reduction from 35.4% to 33.8%), the gains are limited — particularly with stateless predictors. LSTM-based predictors show slightly better adaptability due to their sequential memory, but overall performance still lags behind expectations, suggesting that shallow adaptation of the predictor alone is insufficient for robust domain transfer.

The findings highlight the need for better decoupling of acoustic and linguistic components in E2E models. Future directions include pseudo-audio generation and unified modality modelling to bridge the gap between text and speech in low-resource, domain-specific scenarios.

5.2. Adaptation with pseudo-audio representations from a unified-modal model

The previous adaptation experiments with the NN-LM layer demonstrate that updating only the predictor's weights can work less efficiently than adapting the whole model by training it further with paired audio-text data. To update the encoder and the joint networks with text-only, any substitution of the audio would be necessary. For example, the neural transducer model can be fine-tuned with artificial paired audio and text data. Instead of collecting real audio, artificial audio can be generated directly from text.

Audio generation can be achieved using a multi-speaker neural text-to-speech (TTS) model (Sim et al., 2019; Hori et al., 2019; Li et al., 2020b; Deng et al., 2021; Zheng et al., 2021; Fazel et al., 2021) or a spliced data method (Zhao et al., 2021). Yet, these methods often suffer from high computational demands. To mitigate this, recent research on text-only adaptation has explored the use of simplified pseudo-speech representations known as `textograms`, which are created by repeating one-hot encodings of output labels over a fixed duration (Thomas et al., 2022). The transducer model is modified to accept textograms as an additional input channel. However, this approach requires training the transducer from scratch and tends to increase decoding latency. Mittal et al. (2022) propose an imputation model design that utilises existing text-to-speech representations and alignments learned by the pre-trained transducer ASR model. Then, speech features from the final layer of the speech encoder network are used along with the text data to fine-tune the model.

In the experiments presented in this section, we investigate the potential of pseudo-

audio features generated by a model that unifies speech and text modalities. This method is conceptually similar to creating synthetic speech using a text-to-speech (TTS) system. However, by leveraging pseudo-audio features directly, we bypass the final stages of the TTS pipeline — specifically, vocoder-based waveform synthesis and subsequent feature extraction — thereby streamlining the process. We propose to leverage a pre-trained model with the unified speech and text modalities to generate pseudo-audio features directly from text and use them to replace the real audio features.

We first analyse different possibilities for pseudo-audio features extracted from text. Then, we train the Transducer model with both audio and pseudo-audio features. Finally, we fine-tune the model on the target domain with text-only input.

5.2.1. Pre-trained models with unified modalities

In this section, we describe two existing models that aim to learn a unified representation space by training on both audio and text modalities. The core idea behind these models is to map speech and text into a shared embedding space. Our hypothesis is that embeddings derived from audio and text inputs using a unified encoder will occupy the same or closely aligned representation space, making them interchangeable.

SpeechT5. SpeechT5 is a unified framework for self-supervised speech and text processing designed to handle multiple speech-related tasks — such as ASR, text-to-speech synthesis (TTS), speaker identification, speech translation, etc. — within a single, modular architecture (Ao et al., 2022a). At its core, SpeechT5 follows the encoder-decoder architecture of Text-to-Text Transfer Transformer (T5) (Raffel et al., 2020), but adapts it for multimodal inputs. The model supports both speech and text modalities by employing separate pre/post-nets to process different types of inputs and outputs. It is trained using multi-task learning and can be fine-tuned for specific downstream tasks.

The architecture of SpeechT5 is illustrated at Fig. 5.1, which is a single transformer-based encoder-decoder model shared across multiple tasks and modalities. The backbone is designed to process sequences of either speech or text by mapping them into a common latent space. SpeechT5 is first pre-trained using a denoising sequence-to-sequence approach, leveraging large-scale unlabeled text and speech data. The encoder is pre-trained using masked acoustic modelling, where certain segments of the input spectrogram are masked and the model learns to reconstruct them. The decoder is pre-trained on text infilling and sequence-to-sequence text tasks (e.g.,

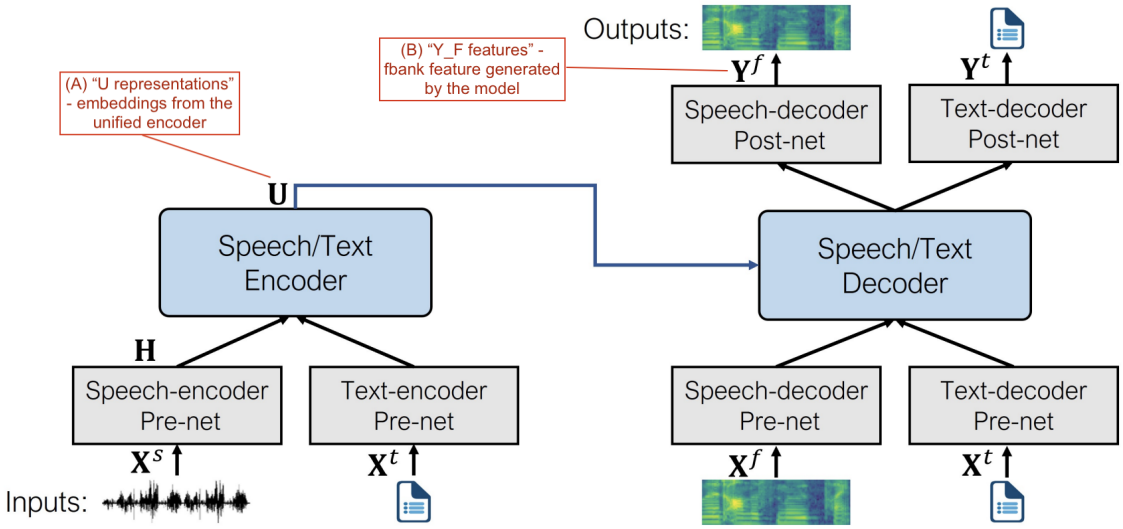

Figure 5.1.: The architecture of SpeechT5 (the illustration is taken from (Ao et al., 2022a). Red text indicates the two types of possible audio representations to be extracted from text with SpeechT5: (A) U representations and (B) Y_f features.

machine translation, summarisation). To align textual and acoustic information within a unified semantic space, the model employs two key strategies: (1) it projects both text and speech inputs into a shared vector quantization (VQ) space, and (2) it randomly blends the quantised latent codes with contextual encoder states, enabling the quantiser to more effectively capture and represent cross-modal features. After the pre-training step, the model is fine-tuned to the target downsteam tasks.

SONAR. SONAR is a multimodal and massively multilingual representation model with the goal of creating a *unified sentence embedding space* for text and speech across over one hundred languages (Duquenne et al., 2023). The model is designed to power a wide range of cross-lingual and cross-modal applications such as speech translation, cross-lingual retrieval, and speech-text alignment. The text encoder is a transformer trained to encode sentences into a shared embedding space. The speech encoder is a Conformer model, which captures temporal and frequency features in speech and maps them into the same embedding space as text. During the training both text and speech encoders are combined to provide a single sentence embedding space.

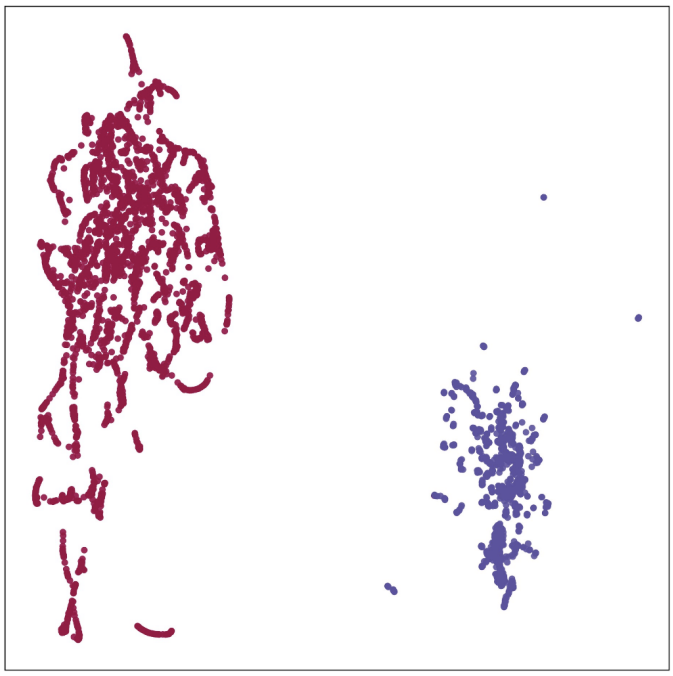

Figure 5.2.: Distribution of the SpeechT5 U-representation extracted from audio (red) and text (violet).

5.2.2. Feature analysis

In this section, we investigate three types of feature representations extracted directly from text using the unified-modality models introduced earlier and that can potentially substitute real audio features. To evaluate their suitability, we visualise the embedding distributions for each feature type using Uniform Manifold Approximation and Projection (UMAP) dimensionality reduction method[42] (McInnes et al., 2018). Each plot compares embeddings derived from text data (shown in red) and speech data (in violet). If the text and speech embeddings largely overlap in the projected space, they may be considered interchangeable. Conversely, if they form clearly separate clusters, substitution is likely infeasible.

SpeechT5 U-representations. SpeechT5 U-representations are extracted from the SpeechT5 unified audio-text encoder (see Fig. 5.1-A). In Fig. 5.2, the features extracted from text and audio form two distinct clusters, indicating that they follow different underlying distributions.

When the transducer model is trained with U-features extracted from audio and evaluated with U-features from text, the ASR performance fails with a WER of 98%.

[42] "UMAP uses local manifold approximations and patches together their local fuzzy simplicial set representations to construct a topological representation of the high dimensional data. <...> UMAP then optimizes the layout of the data representation in the low dimensional space, to minimize the cross-entropy between the two topological representations." (McInnes et al., 2018)

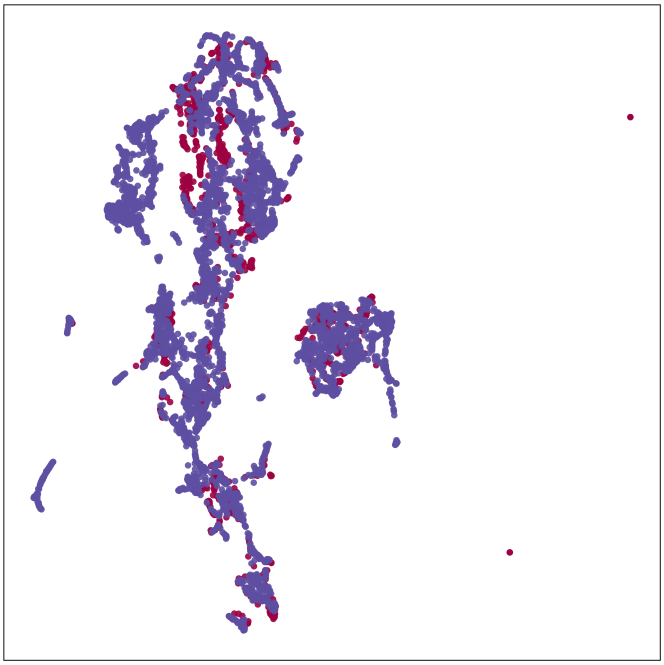

Figure 5.3.: Distribution of the SpeechT5 Y_f features extracted from the decoder-postnet: from audio (red) and text (violet).

The features are too different to be used for mutual replacement in adaptation.

SpeechT5 Y_f features. SpeechT5 Y_f features are synthetic Fbank features generated by the decoder-postnet, audio-generating decoder (see Fig. 5.1-B). Fig. 5.3 shows that the distributions of Y_f features extracted from text and audio are overlapping. Additionally, when the model is trained on audio features, its performance when evaluated with the SpeechT5 Y_f Fbank features extracted from text is comparable to the performance evaluated with the Fbank features extracted from audio. These results assume that the SpeechT5 Y_f features can be used as a replacement for real audio features to create pseudo-parallel adaptation data.

SONAR frame-level features. As the SONAR model provides the aligned embeddings for audio and text on the *sentence* level, it can not be used directly for training, where the frame-level features are expected. To obtain the frame-level representations, we extracted the embeddings before the pooling layer, which combines frame-level embeddings into a final sentence embedding.[43] We build the distribution plot for both (A) sentence-level and (B) frame-level embeddings (Fig. 5.4). From two distributions, we can see that if text and speech modalities are well aligned on the sentence level (Fig. 5.4-A), it is not the case for the frame-level embeddings that are grouped differently in the embedding space (Fig. 5.4-B). From this observation, we conclude that this type of text-speech embedding does not suit our goals.

[43] We use `output.encoded_seqs` instead of the final `output.sentence_embeddings`.

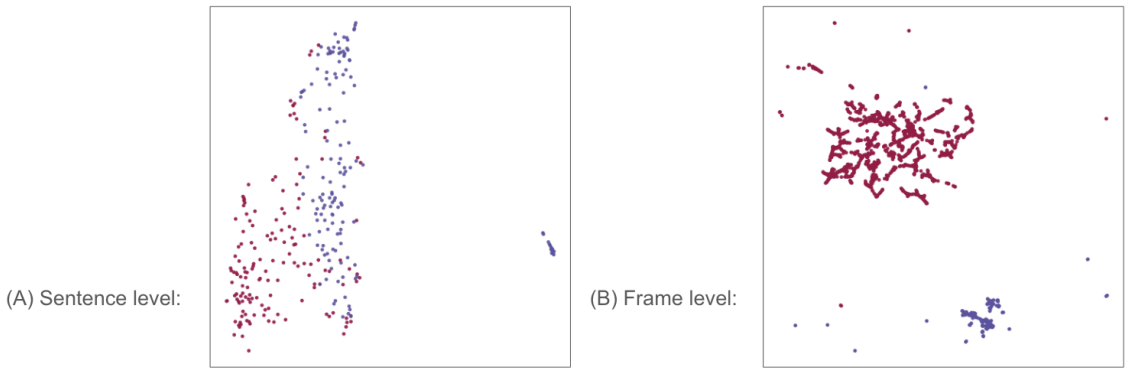

Figure 5.4.: Distribution of the SONAR feature extracted on the sentence-level (A) and on the frame-level (B): from audio (red) and text (violet).

According to the distributions, both SpeechT5 U-representations and SONAR frame-level features are too different when generated from text and speech input to be used for real audio features replacement. Thus, based on the observations obtained from the analysis of different features extracted from text data, we proceed with text-only domain adaptation experiments with SpeechT5 Y_f Fbank features.

5.2.3. Experiments

For the experiments, we use the GigaSpeech-S subset of 250 hours as the training data and the DefinedAI data (Sec. 2.4.2.2) for the adaptation (48 hours) and test sets.

Model. As the base architecture, we use the pruned stateless transducer model from Icefall. We modify the encoder in a way that it can (1) accept both audio and text inputs and (2) process Fbank features extracted from speech and text. We train the model from scratch for 30 epochs on 250 hours of the paired audio-text data (GigaSpeech-S). We use a batch size of 120s with a gradient accumulation of 1, and the peak learning rate is set to $lr = 3.0 \times 10^{-3}$. For the evaluation, we use a single model checkpoint without model averaging and a beam size of 8.

Features. The pseudo-audio features (SpeechT5 Y_f Fbank features) are extracted from text with the decoder-postnet before the vocoder[44] is used to generate the audio. Since SpeechT5 was trained with the Fbank features extracted with the *librosa* package (McFee et al., 2015), we train our base model also with the librosa Fbank features instead of the default in Icefall kaldi-based Fbank features.

[44] A vocoder is a device or technique that analyzes and synthesizes a sound signal, most often a human voice, to create a unique, often robotic-sounding effect.

Table 5.2.: Domain adaptation evaluated on DefinedAI test set and adapted with text-only data: 48h (health, insurance, banking) + 299h (insurance) + 173h (banking). Predictor (P), joiner (J), decoder linear projector (LP-LM); LinearProjector for feature projection. The base model is trained on GigaSpeech-S data of 250 hours and with librosa audio Fbank features.

Exp	**Fine-tuning:**			**DefinedAI**			
	audio features	**layers**	**data,h**	**all domains**	**health**	**insurance**	**banking**
(0)	-	-	-	25.0	24.3	24.9	26.0
(1)	Audio (Fbank)	all layers	48	*11.7*	*12.7*	*10.5*	*11.7*
(2)	Text (SpT5-Yf)	all layers	48	28.7	27.4	28.0	30.6
	With LinearProjection:						
(3)	Text (SpT5-Yf)	P, J, LP-LM	48	22.6	**21.7**	22.3	23.9
(4)	Text (SpT5-Yf)	P, J, LP-LM	512	**22.5**	22.8	**21.8**	**23.1**

Experiments. We first trained the base transducer model with the GigaSpeech-S train set. Then, we conducted the following fine-tuning experiments with the DefinedAI adaptation data set:

1. All the parameters are fine-tuned for 10 epochs on supervised audio-text data (48 hours).

2. All the parameters are fine-tuned for 5 epochs on text-only data (48 hours) with audio replaced with SpeechT5 Y_f features extracted from the same text.

3. Predictor, Joiner, LinearProject-LM, and an additional feature LinearProjector, are fine-tuned for 18K steps on text-only data (48 hours) with audio replaced with SpeechT5 Y_f features extracted from the same text.

4. Predictor, Joiner, LinearProjector-LM, and an additional feature LinearProjector, are fine-tuned for 18K steps with extra text data for two target domains: "insurance" (299 hours) and "banking" (173 hours) domains. Since the amount of extra text data is different for different domains and since we did not have any extra text data for the third "health" domain, the evaluation is done separately for each domain.

5.2.4. Results

Table 5.2 summarises the WER results on domain adaptation with text-only input data and pseudo-audio features. Updating all the model parameters with text data leads to performance degradation, when further training increases the WER even more. This behaviour happens because updating the encoder's layers with non-

audio data negatively influences its ability to predict on real audio. With a frozen encoder but updating the text-related components — the projector, joiner, decoder linear projector, and an additional feature linear projector — the model benefits from the adaptation with text data. The table shows that with text-only data, we gain 10.7%, 12.4%, and 11.1% relative WER improvement compared to the baseline (experiment-0) for health, insurance, and banking domains, respectively.[45]

The experiments demonstrate that incorporating additional in-domain text data leads to a further reduction in WER. However, the gains are modest: increasing the data volume more than tenfold (from 48 to 512 hours) yields only a 2.2% relative improvement in the insurance domain and a 3.3% improvement in the banking domain.

Overall, the performance gains achieved using text-only data and pseudo-audio features remain significantly lower compared to those obtained with real audio data. These results highlight the crucial role of encoder adaptation in effective domain transfer. While pseudo-audio can be beneficial for adapting the model's text-related components, it provides little advantage for updating the encoder itself and, therefore, limits its impact on overall ASR performance.

5.2.5. Summary

An analysis of three different types of representations reveals that frame-level embeddings produced by uni-modal encoders — such as those from SpeechT5 and SONAR — exhibit distinct distributions when derived from speech versus text inputs. Among these, the representations generated by the decoder postnet of the SpeechT5 model most closely resemble traditional speech features like log Mel filterbanks (Fbank). As a result, SpeechT5 postnet features were selected for adaptation experiments to generate synthetic paired data from text-only inputs.

The adaptation experiments demonstrate that fine-tuning the text-related components of the model using text-only data and pseudo-audio features leads to measurable performance improvements. Incorporating additional text data provides further gains, though these are relatively modest. At the same time, pseudo-audio features contribute minimally to encoder adaptation and even cause degradation of the overall performance. It might be the main reason why the enhancements achieved through text-only adaptation remain significantly below those obtained with real audio data. The findings indicate the importance of adapting the encoder

[45] Table A.3 includes additional evaluation on the GigaSpeech test set and on the DefinedAI test set but with the text input instead of speech.

for effective domain transfer.

5.3. Chapter summary

This chapter explored the key challenge of domain adaptation for End-to-End ASR systems, focusing specifically on leveraging text-based contextual information. We first established the inherent difficulties in LM adaptation within End-to-End architectures, contrasting them with the more flexible, cascaded nature of traditional hybrid ASR systems. Unlike standalone LMs, the internal LMs (ILMs) of End-to-End models are acoustically conditioned and intricately linked to the entire model's parameters, making text-only adaptation a non-trivial task.

To address these challenges, we explored two distinct approaches for text-only adaptation. Our initial experiments investigated adapting the predictor network of a Transducer model by adding an NN-LM layer. While this method aimed to provide a quick adaptation solution by updating only the predictor weights, our results demonstrated its limited efficacy. The stateless predictor showed no measurable adaptation, and even with an LSTM predictor, the NE WER improvements were marginal, and in some cases, the overall WER performance degraded. This indicated that updating the predictor in isolation is insufficient for robust domain adaptation, underscoring the need for a more comprehensive model update.

Consequently, we shifted our focus to methods that could enable broader model fine-tuning with text-only data by leveraging pseudo-audio representations. We explored the potential of features generated by unified speech and text modality models, specifically SpeechT5 and SONAR. Through detailed feature analysis and visualisation using UMAP, we determined that SpeechT5's Y_f Fbank features, which showed significant overlap between text-derived and audio-derived distributions, were the most promising candidates for replacing real audio features. Conversely, SpeechT5 U-representations and SONAR frame-level features proved unsuitable due to distinct modality distributions. The subsequent experimental setup outlined our approach to fine-tune the entire Transducer model using these synthesised Y_f features, providing a pathway to update the encoder and joint networks, which was not possible with predictor-only adaptation.

In summary, this chapter highlights the challenges of domain adaptation in End-to-End ASR using text-only data and presents preliminary results on two adaptation strategies: adding an NN-LM layer and leveraging pseudo-audio features from SpeechT5. Both approaches yield only marginal improvements on in-domain perfor-

mance compared to standard fine-tuning, underscoring the difficulty of effectively integrating textual context without paired audio. As a result, robust text-based adaptation for End-to-End models remains an open problem. While factorised architectures offer a promising direction by enabling more flexible language model integration, they represent a departure from the unified, joint training paradigm that defines End-to-End systems, raising trade-offs between adaptability and architectural purity.

6. Effect of contextualisation on ASR-related tasks

In the previous two Chapters (Chapters 3 and 4), we demonstrated how contextual knowledge can be successfully integrated into different hybrid and End-to-End ASR models. In this Chapter, we take a step further to see how context can be leveraged to improve the performance of ASR-related tasks. For example, downstream Natural Language Processing (NLP) tasks on top of ASR outputs or ASR training in the low-resource scenario. First, we provide experimental evidence and discussion on how contextual biasing improves the *named entity recognition* (NER) task for both hybrid and End-to-End ASR. Second, we demonstrate how contextual biasing can benefit ASR training when only limited resources are available, i.e. (1) *semi-supervised learning* (SSL) and (2) boosting the performance of streaming ASR models trained from scratch on pseudo-labels.

6.1. Leveraging contextual knowledge for NER from speech (hybrid ASR-NER pipeline)

The section is partly based on the publication (Nigmatulina et al., 2022).

While contextual information has been utilised in previous ATC studies (Schmidt et al., 2014; Shore et al., 2012; Oualil et al., 2015, 2017) and more recently in (Kocour et al., 2021; Nigmatulina et al., 2021; Zuluaga-Gomez et al., 2021), it has never been simultaneously adapted for both ASR and concept extraction outputs without requiring additional knowledge (e.g., manual annotations, predefined classes, etc.). In this section, we investigate the potential benefit of ASR contextual biasing for the downstream NLP task of NER. We stay within the same use case of the ATC domain and prove that combining the benefits of ASR and NLP methods to make use of surveillance data (i.e. contextual knowledge of a different modality) helps to considerably improve the named entity (callsigns) recognition.

The experiments described in this section seek to leverage contextual information through the integration of ASR and NLP techniques. Rather than treating ASR and NLP as isolated tasks, we consider them complementary: ASR transcribes spoken language into text, while NLP exploits the structural characteristics of text to extract meaningful insights. Many End-to-End ASR models – particularly those optimised for local acoustic-phonetic alignment – may still struggle to maintain global semantic coherence across extended utterances. In contrast, modern NLP approaches are capable of identifying critical information across extended text spans, such as complete ATC utterances.

We investigate a two-step callsign boosting approach and iteratively incorporate contextual data by combining ASR and NLP modules.

6.1.1. Step 1 (ASR): integrating contextual knowledge during ASR

At the 1st step (ASR), the probabilities of callsigns are boosted in the ASR system. The biasing is done with one of the contextual biasing methods introduced above or with their combination: (1) *"lattice biasing"* (Method 1 ; Sec. 3.1.2) when the weights of probable callsign n-grams are boosted in the decoding FST, i.e. lattices, (2) *"G-boosting"* (Method 2; Sec. 3.1.3) when the probable callsign n-grams are boosted in G.fst, and (3) combination of lattice biasing and G-boosting.

In the biasing experiments described in the sections above (Sec. 3.2 and 3.3), the same target entities are used as contextual knowledge for biasing with both Methods 1 and 2. Using the same biasing entities allowed a fair comparison of the methods and was possible because of the annotations available per utterance. However, in real life, the G-boosting method is less flexible in adaptation than lattice biasing, as it needs access to HCL, and is more suitable for the static context available before decoding. At the same time, lattice rescoring can use the dynamic context and bias the entities of the current time point.

In the experiment on ASR- and NLP-boosting, our setup is close to the real scenario when we know (1) a broad static list of possible callsigns (a few thousand) and (2) a narrow dynamic list of air crafts changing depending on the situation in the airspace. Our first integration of contextual knowledge into ASR is done at the LM level (*G-boosting*). The idea is to boost callsign n-grams already available in LM, and even more important to add those callsign n-grams, which are absent (e.g., >3 word sequences in 3-gram LM). We build a contextual *FST* that includes all possible callsigns from the tower: all callsigns registered by the radar at different time stamps (from 17K to 280K callsigns to boost in different test sets; see the last column in

Table 2.4). Then, the main $G.fst$ is composed with the contextual $G\_biased.fst$, and the result of the composition is used in the final decoding $HCLG$ graph. The second integration of dynamic context is done with the lattice biasing method, as described in Sec. 3.1.2.

6.1.2. Step 2 (NLP-boosting): integrating contextual knowledge post-ASR decoding

At the 2nd step (NLP), callsigns identified by the NER model from the enhanced recognition outputs are matched against surveillance data to determine the most appropriate candidate. Our approach for integrating contextual knowledge on top of the ASR transcripts (e.g., 1-best hypothesis) is based on a two-step pipeline. Each step conveys an independent module.

Step 2.1: Named Entity Recognition (NER) module. ATC communications carry rich information such as callsigns, commands, values and units; they can be seen as 'named entities'. We defined callsigns, commands, units, values, greetings OR the rest (e.g., 'None' class) as tags for the NER task, as depicted in Fig. 6.1. Then, we apply an NLP-based system to extract such information from ASR transcripts. We use two different NER models: (1) the data-driven **BERT** model (Devlin et al., 2019) and (2) a rule-based extractor with the base **spaCy**[46] model (Honnibal et al., 2020). For BERT, we downloaded a pre-trained masked language model from Huggingface (Wolf et al., 2020) and fine-tuned it on an NER task with 12k sentences (∼12 hours of speech), where each word has a tag.

Then, we developed a data augmentation pipeline to increase the amount of training data: 1M samples from 12k sentences. The pipeline has four actions that modify the training sample: *add*, *delete*, *swap*, or *move* the `callsign` across the utterance -sentence-. *Delete* and *move* actions remove and keep the same callsigns, respectively; *add* and *swap* generate a sentence with a new callsign picked randomly from a callsign list. The callsign list is pre-defined by a user, which makes the approach easy to deploy in out-of-domain data (i.e., callsigns from different airports/countries). For the spaCy model, we use the rule-based extractor. First, the token patterns are defined based on the structure of callsigns. Then, the pre-defined patterns are integrated into the NER system with spaCy's *EntityRuler*.

Step 2.2: Re-ranking module based on Levenshtein distance. The NER

[46] SpaCy is an open-source library that offers efficient and scalable tools for text processing and entity extraction: https://spacy.io/.

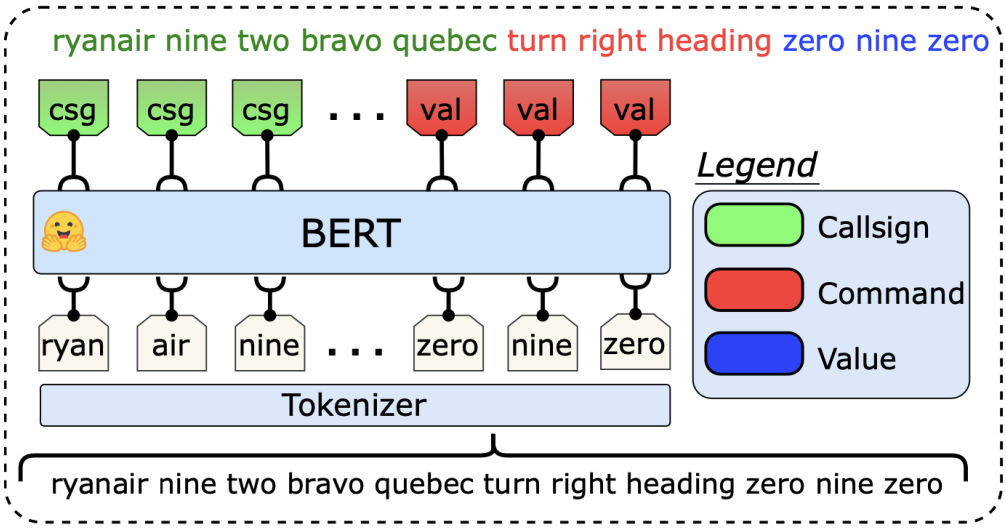

Figure 6.1.: BERT-based model (Huggingface) fine-tuned on NER task.

systems allow callsign extraction from a given transcript or ASR 1-best hypotheses. A single error produced by an ASR system affects the recognition of the whole entity that is normally composed of three to eight words. Additionally, speakers regularly shorten callsigns in the conversation, making it impossible for an ASR system to generate the full entity (e.g., *'three nine two papa'* instead of *'austrian three nine two papa'*, *'six lima yankee'* instead of *'hansa six lima yankee'*). One way to overcome this issue is to re-rank entities extracted by the NER system with the surveillance data. The output of an NER system is a list of tags that match words or sequences of words in an input utterance. As our only available source of contextual knowledge is callsigns registered at a certain time and location, we extract callsigns with the NER system and discard other entities. Consequently, each utterance has a list of callsigns expanded into word sequences (shown in Table 2.2). As input, the re-ranking module takes (i) a callsign extracted by the NER system and (ii) an expanded list of callsigns. The re-ranking module compares a given n-gram sequence against a list of possible n-grams and finds the closest match from the list of surveillance data based on the weighted Levenshtein distance. We skip the re-ranking in case the NER system outputs a 'NO_CALLSIGN' flag (no callsign recognised).

6.1.3. Experiments

We report the results from two sets of experiments on contextualisation with the two-step (ASR-NLP) approach. The first experiments were conducted in 2021 and are published in (Nigmatulina et al., 2022). The second experiments were performed

in 2024 with an updated AM. The experiments are not directly comparable because (1) the training data is different, even if the amount of supervised data is the same, the distribution of included ATC datasets is different, and the latest model is trained on more balanced data; (2) different NER extractors are used, when in the earlier experiments, we use BERT model and in the later experiments we apply only the rule-based spaCy extractor, because it proved to be more accurate compared to the BERT NER.

Experiments 1 (2021). We use four test sets: LiveATC (Sec. 2.4.1.2), MALORCA Prague (Sec. 2.4.1.2), MALORCA Vienna, NATS (Sec. 2.4.1.2). All of them have utterances both with and without callsigns (see Table 2.4). It is important to note that the data sets are used differently in training ASR and NER models. The ASR training data includes MALORCA sets but not LiveATC and NATS. The data for fine-tuning the NER system contains LiveATC data, but neither MALORCA, nor NATS sets. Thus, NATS data can be considered as a fully out-of-domain test set. **ASR model.** The CNN-TDNNF model is trained on ATC labelled data and the Kaldi framework, as described in Sec. 3.2.1 as Model 1 CNN-TDNNF-ATC-1. **NER model:** BERT model fine-tuned with the labelled ATC data.

Experiments 2 (2024). We use three test sets: MALORCA Prague, NATS, and ATCO2-test-set-1h (Sec. 2.4.1.2). All of them have utterances both with and without callsigns (see Table 2.4). Both MALORCA and NATS data are used during training, and thus are considered as "in-domain" datasets. The ATCO2 data is "out-of-domain" because no ATCO2 data has been seen during training. **ASR model.** The same CNN-TDNNF model is trained on ATC supervised data and the Kaldi framework; the model is described in Sec. 3.2.1 as Model 2 CNN-TDNNF-ATC-2. The difference from the ASR model trained for the first set of experiments is the data: see the difference between models CNN-TDNNF-ATC-1 and CNN-TDNNF-ATC-2 in Sec. 3.2.1. 500 hours of untranscribed "in-domain" ATC data[47] is added to further improve the performance. The training data was modified to include the new data from other ATC domains from new available datasets (i.e. from different airports) and to keep the training set balanced.[48] **NER model:** rule-based spaCy extractor.

Evaluation. As this work aims to enhance callsign detection, we evaluate the proposed methods based on the accuracy of callsign extraction. The evaluation is performed using the ICAO format, which is the standardised form displayed to air

[47] The data is from the following ATC providers: (1) NATS for London approach and (2) ISAVIA for Icelandic en-route

[48] More details on the data in the updated training set can be found in (Zuluaga-Gomez et al., 2023b).

traffic controllers and pilots. The results are assessed in binary terms: a callsign is considered either 'correctly' or 'incorrectly' recognised according to its conformity with the ICAO format. In previous studies (Kocour et al., 2021; Nigmatulina et al., 2021), the accuracy of callsign recognition is evaluated by matching the ground truth callsign n-grams to those in utterances. However, this approach does not reflect real-world conditions, where ground-truth callsigns are typically unavailable. In our experiments, we go beyond speech recognition by performing callsign extraction, and therefore assess performance directly on the extracted entities. Additionally, evaluating with the ICAO format mitigates issues related to pronunciation variability within callsigns. For instance, while the full callsign form may be automatically extracted, speakers often use shortened variants that are reflected both in the ASR output and the reference transcriptions (see example above 6.1.2).

6.1.4. Results

As a baseline, we use callsign extraction done directly on the outputs of our ASR system. Then, we apply the proposed boosting techniques (G-boosted, lattice biasing, NLP-boosting) in different combinations to see how they can benefit from each other.

Experiments 1 (2021). In Table 6.1, the results of the experiments from 2021 are presented on four different test sets with the accuracy of callsign (ICAO) detection. Overall, the proposed metrics help to improve the baseline accuracy from 30.6% to 53.7% absolutely, or from 32.1% to 60.4% relatively (for the test sets Prague and NATS correspondingly; when the NATS set gets the highest improvement, being the out-of-domain data). The best results are consistently obtained when using NLP-based boosting. For the LiveATC and NATS test sets, both of which are out-of-domain relative to the ASR training data, the highest performance is achieved through a combination of NLP-boosting and ASR-based boosting (lattice biasing) techniques.

Conversely, G-boosting exhibits a mixed impact. While it improves performance over the baseline for the LiveATC and Vienna test sets, its combination with lattice rescoring results in lower accuracy than using lattice rescoring alone. A potential limitation of the G-boosted approach in this context is the extensive number of callsigns incorporated into the LM FST (see the last column in Table 2.4). This large set may introduce ambiguity when combined with lattice rescoring, which is designed to prioritise only the currently relevant callsigns. Nevertheless, G-boosting has the advantage of requiring no modifications to the decoding process and can

Table 6.1.: Results of callsign extraction with ASR-boosting (Lattice biasing and G-boosted) and NLP-boosting (post-boosting) (Nigmatulina et al., 2022): the accuracy of callsign recognition (%) is calculated for the callsigns in ICAO format (see Section 6.1.3). The best results are **highlighted in boldface**.

Method			**Test sets (callsign detection accuracy)**			
			LiveATC	Prague	Vienna	NATS
ASR outputs (CNN-TDNNF)						
Callsign extraction with BERT-NER (baseline)			42.8	64.4	48.4	35.2
Lattice biasing	G-boosted	NLP-boosting				
✓	-	-	53.1	66.9	59.6	37.1
-	✓	-	44.4	64.3	49.2	34.8
✓	✓	-	52.8	66.9	52.1	36.8
-	-	✓	88.4	**95.0**	**86.0**	87.0
✓	-	✓	**88.5**	94.8	84.3	**88.9**
-	✓	✓	87.7	**95.0**	85.6	88.2
✓	✓	✓	88.0	94.7	84.0	88.0
Manual transcriptions						
Callsign extraction (oracle)			**89.7**	72.2	59.6	67.4
+ NLP-Boosting			89.3	**95.4**	**87.0**	**94.0**
ASR WER (without boosting)			32.4	3.4	9.2	24.4

function as a general domain adaptation strategy. Therefore, it may be beneficial in scenarios where other methods are not applicable, but can otherwise be omitted. The number of callsigns used for ASR output boosting can potentially degrade the performance of the lattice rescoring method. Although the maximum number of boosted callsigns in our experiments did not exceed 50, we still examined its influence. The test sets included varying quantities of boosted n-grams, ranging from 5 to 50 (see Table 2.4). Nevertheless, even with as many as 50 boosted callsigns, recognition accuracy showed a substantial improvement over the baseline.

In addition to evaluating the boosting methods on ASR outputs, we also report 'oracle' results, where callsigns are extracted directly from ground truth transcriptions (see the second horizontal block in Table 6.1). This comparison provides insight into how much the proposed methods enhance callsign extraction in the absence of ground truth data. While the oracle scores consistently outperform the others, the accuracy achieved by our systems remains closely comparable. The lack of improvement from NLP-boosting on the ground truth transcriptions for the LiveATC test set is likely due to the already high callsign extraction accuracy, as LiveATC data was used during NER fine-tuning.

Table 6.2 gives examples of improvement where airline names and callsigns are

Table 6.2.: Examples of improved callsign detection, NER (bold part)

Baseline (incorrect ICAO)	**Boosted (correct ICAO)**
wizz air four one six (**WZZ**416)	**iceair** four one six (**ICE**416)
easy three delta (**EZY**3D)	**fraction eight eight** three delta (**NJE88**3D)
serbia one nine lima (**ASL**19L)	**stobart** one nine lima (**STK**19L)

detected correctly compared to the baseline predictions. Our methods demonstrate consistent results for data of different quality. The level of noise in the recordings of LiveATC and Malorca test sets is very different, as well as WERs achieved by their baseline ASR systems (the last line in Table 6.1; (Nigmatulina et al., 2021)). Nevertheless, we see considerable improvement for all test sets, and the general tendency stays the same.

Experiments 2 (2024).[49] In Table 6.3, the results of the experiments from 2024 are presented on three different test sets with the accuracy of callsign (ICAO) detection. In this set of experiments, we focused exclusively on G-boosting and NLP-boosting methods. Notably, the rule-based spaCy model outperforms the fine-tuned BERT model on the NER task when evaluated on in-domain datasets. The accuracies for callsign detection are 86.0% and 70.2% for the in-domain test sets. It is harder to make any assumptions for the out-of-domain data sets, as each set of experiments has only one out-of-domain test set: 29.7% (ATCO2-test-set-1h) with spaCy and 35.2% (NATS) with BERT model.

Given that earlier results indicated G-boosting performs poorly when applied to a large set of callsigns, these experiments restrict G-boosting to ground truth callsigns only. When limited in this way, G-boosting consistently improves the accuracy of callsign extraction especially for the out-of-domain test set: 3.6%, 6.7%, and 20.9% relative improvement for MALORCA-Prague, NATS, and ATCO2-1h datasets, respectively.

Overall, the results are consistent with the previous experiments. The best enhancement is reached with the NLP-boosting contextualisation that can be further improved when combined with G-boosting: 9.0%, 20.9%, and 88.9% relative improvement for MALORCA-Prague, NATS, and ATCO2-1h datasets, respectively.

[49] The results were obtained with the help of Driss Khalil.

Table 6.3.: Results of callsign extraction with CNN-TDNNF+spaCy-NER (hybrid ASR, pipeline). The accuracy of callsign recognition (%) is calculated for the callsigns in ICAO format (see Section 6.1.3). The best results are **highlighted in boldface**.

Method		**Test sets (callsign detection accuracy)**		
		Prague	NATS	ATCO2-1h
ASR outputs (CNN-TDNNF)				
Callsign extraction with spaCy-NER (baseline)		86.0	70.2	29.7
G-boosting	NLP-boosting			
✓	-	89.1	74.9	35.9
-	✓	92.2	88.6	52.4
✓	✓	**93.8**	**89.9**	**56.1**
ASR WER (without boosting)		3.6	6.5	29.1

6.1.5. Summary

We investigated a two-step approach to integrate contextual radar data in order to dynamically improve the recognition of callsigns per utterance. We demonstrated that the best result is achieved with the NLP-boosting and with the combination of NLP-boosting and lattice rescoring or G-boosting methods on all test sets of different recording quality with the significant improvement, i.e., from 32.1% to 60.4% of relative improvement in callsign recognition accuracy across the evaluated data sets. The main advantage of the proposed approach compared to the others is its simplicity and flexibility. The NER-system can be fine-tuned to different data sets, which makes it easy to adapt to new out-of-domain data. The introduction of contextual information considerably improves recognition of callsigns and, thus, the recognition of ATCo messages in general. As a noisy environment leading to lower recognition accuracy is often a reality in pilot-ATCo communication, the proposed methods and their combination will definitely benefit the recognition of the key information in ATCo speech.

6.2. Leveraging contextual knowledge for NER with a multitask End-to-End model

The experiments reported above demonstrate (1) how contextual knowledge can be injected into the End-to-End ASR models (Sec. 4.1) and (2) how contextualisation can be used for the NER tasks applied on top of hybrid ASR outputs in an

ASR-NER pipeline (Sec. 6.1). The preceding section demonstrated the effectiveness of our two-step ASR-NER pipeline for callsign recognition, showing significant improvements over a standard ASR baseline. However, as noted, this hybrid approach has inherent complexities. It relies on a multi-stage process where errors from the ASR system can propagate and affect the downstream NER module, and it requires managing two separate, independent components. These challenges can hinder real-time performance and system robustness.

In this section, we explore a more integrated and streamlined solution. We investigate how contextual information can be directly incorporated to enhance End-to-End NER from audio, thereby eliminating the need for a separate, post-decoding NLP pipeline. As an End-to-End NER model, we consider a multitask model called *TokenVerse* (Kumar et al., 2024b).

6.2.1. TokenVerse: multitask model for NER from speech

TokenVerse is a single transducer-based model capable of performing multiple tasks. In the original paper, TokenVerse (Kumar et al., 2024b) is tested on speaker change detection (SCD), endpoint detection (ENDP), and NER tasks. In our experiments, we adapt the approach to extract callsigns from speech with the NER task.

TokenVerse is able to perform multiple tasks by embedding task-specific tokens directly into the reference text during ASR training, as illustrated in Fig. 6.2. Task tokens inserted into the training text enable the model to identify and align tokens within transcripts. Training is performed using the XLSR-Transducer architecture (Conneau et al., 2020; Kumar et al., 2025a) that consists of an XLSR encoder, a stateless predictor (Ghodsi et al., 2020), and a joint network (linear layer). Subwords are trained with a SentencePiece tokeniser (Kudo and Richardson, 2018) with the training data that includes task-specific tokens; later in training, task tokens are treated as single subword units. During inference, tasks are solved by locating the task tokens, e.g. the words enclosed by the opening and closing tokens for the NER task. Therefore, the TokenVerse approach simplifies inference and removes the necessity for separate NLP components.

In our experiments, we place a $[CALLSIGN]$ token before each callsign and a $[/CALLSIGN]$ token after it, allowing the model to learn the boundaries of these entities explicitly.

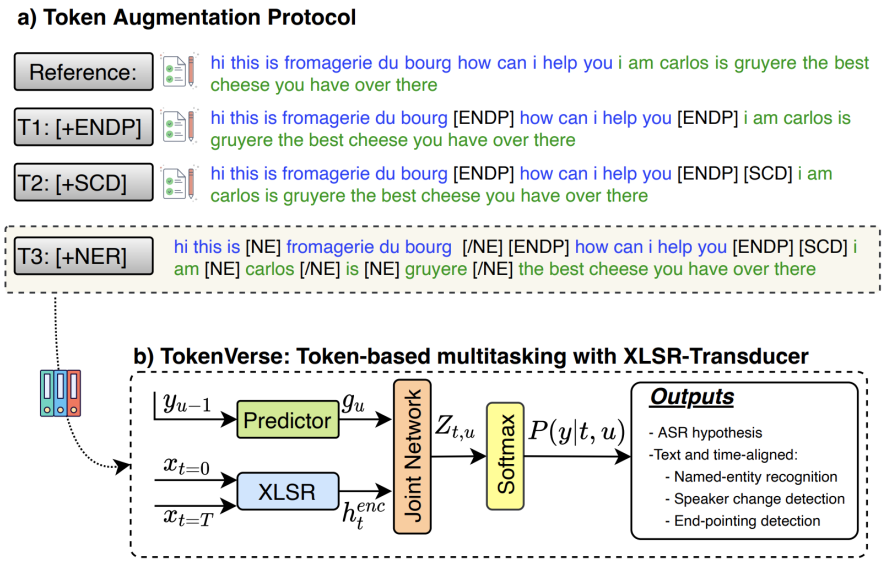

Figure 6.2.: TokenVerse data preparation and training. (a) Integrating task-specific tokens into the reference text for speaker change detection (SCD), endpoint detection (ENDP), and NER tasks. The XLSR-Transducer architecture as the backbone of TokenVerse (the illustration is taken from (Kumar et al., 2024b)).

6.2.2. Experiments

Model TokenVerse. In line with the original TokenVerse approach, we adopt the XLSR-Transducer architecture, utilising the pretrained XLSR model as the encoder. As in the experiments above 4.2.1, the model is trained with the Icefall framework. It is important to note that the training ATC data is the same as for the Model 2 CNN-TDNNF-ATC-2 described in Sec. 3.2.1. To label callsigns in the adaptation data, we employ the *spaCy* model as a rule-based extractor (Honnibal et al., 2020) in the same way as described in 6.1.2.

Contextual biasing. For the contextual biasing, three methods are adopted: (1) shallow fusion (or ASR-boosting), (2) NLP-boosting, and (3) entity matching with spaCy (spaCy matching). The first two methods have been introduced above: the shallow fusion method and the NLP-boosting method are described in Sec. 4.1.3 and Sec. 6.1.2, correspondingly. The *spaCy matching* method provides further improvement by applying entity prediction by spaCy on top of TokenVerse ASR. When combined with NLP-boosting, callsigns from TokenVerse-NER and TokenVerse-ASR+spaCy are both compared to the radar data and the closest prediction is chosen. Without NLP-boosting, the method checks if the callsigns from either TokenVerse-NER or TokenVerse-ASR+spaCy are present in a reference radar dataset. If a match is found, the corresponding callsign from the matched system is used; else, the callsign output from TokenVerse is selected by default.

Table 6.4.: Results of callsign extraction with ASR-boosting (shallow fusion), NLP-boosting (post-boosting), and entity matching (with spaCy): the accuracy of callsign recognition (%) is calculated for the callsigns in ICAO format (see Section 6.1.3). The best results are **highlighted in boldface**.

Method			**Test sets (callsign detection accuracy)**		
			Prague	NATS	ATCO2-1h
TokenVerse NER output					
Callsign extraction (baseline)			76.6	58.8	26.8
Shallow Fusion	NLP-boosting	spaCy matching			
✓	-	-	79.3	67.0	27.0
-	✓	-	96.5	90.6	48.7
✓	✓	-	96.2	90.2	48.3
-	-	✓	77.5	59.1	33.8
✓	-	✓	80.6	67.5	34.1
-	✓	✓	**97.5**	90.9	**63.6**
✓	✓	✓	**97.5**	**91.2**	63.4
Pipeline approach:					
XLSR+G-boosting+NLP-boosting+spaCy-NER			92.2	91.2	60.9
Whisper+NLP-boosting+spaCy-NER			90.8	88.1	64.8
TokenVerse ASR WER (without shallow fusion)			5.2	8.1	19.1
TokenVerse ASR WER (with shallow fusion)			4.9	7.0	19.1

For the evaluation, we use three test sets: MALORCA Prague (Sec. 2.4.1.2), NATS (Sec. 2.4.1.2), and ATCO2-test-set-1h. All of them have utterances both with and without callsigns (see Table 2.4).

6.2.3. Results[50]

Table 6.4 reports the results on callsign extraction with the End-to-End TokenVerse model. Even without applying any boosting techniques, the TokenVerse approach achieves callsign detection accuracy of 76.6% on the Prague test set, 58.8% on the NATS test set, and 26.8% on the ATCO2-1h test set. Subsequently, SF yields a modest relative improvement of 3.5% on the Prague test set and 0.7% on the ATCO2-1h test set, and a more substantial 13.9% on the NATS test set. The biggest improvement compared to the baseline extraction is reached with the NLP-boosting method: relative improvement of 26.0%, 54.1%, and 45.0% on Prague, NATS, and ATCO2-1h test sets, respectively. A little further improvement can be achieved when NLP-boosting is combined with entity-matching with spaCy: relative

[50] The results were obtained with the help of Driss Khalil.

improvement of 27.3% and 55.1% compared to the baseline on Prague and NATS test sets, respectively. For the out-of-domain ATCO2-1h test set, the relative improvement after combining the two NLP methods is 30.6%. Overall, the End-to-End TokenVerse model demonstrates competitive NER results compared to the pipeline approach with the boosted XLSR-LFMMI ASR model and the Whisper model combined with the spaCy NE extractor: 97.5% vs. 92.2% vs. 90.8%, 91.2% vs. 91.2% vs. 88.1%, and 63.6% vs. 60.9% vs. 64.8% for the Prague, NATS, and ATCO2-1h test sets.

6.2.4. Summary

Our experiments showed that NER performance can be improved when contextual information is integrated with ASR and NLP-boosting methods. The multitask End-to-End model for NER can reach and even surpass the performance achieved with the pipeline ASR-NER approach. However, NLP-boosting and entity matching are additional steps on top of the model's outputs, and this contradicts the End-to-End principle of unity. At the same time, both methods are light and fast in their application, as they are based on the string matching algorithm. Moreover, the TokenVerse approach simplifies training and inference, which makes this method attractive for practical applications. Because of the difference in training data (see Sec. 6.1.3), the TokenVerse results cannot be directly compared to the NER experiments reported in Table 6.1 but are comparable to the callsign detection performance on hybrid models reported in Table 6.3. We use the same training data to train hybrid CNN-TDNNF and End-to-End XLSR-Transducer (TokenVerse) models. The results demonstrate the advantage of the TokenVerse method over the hybrid ASR+NER pipeline approach.

6.3. Leveraging contextual knowledge for semi-supervised learning

The section is based on the publication (Zuluaga-Gomez et al., 2021). My contributions include describing and executing contextual biasing experiments.

The results in the previous sections (Sec. 6.1 and Sec. 6.2) show that leveraging contextual knowledge not only helps improve the performance of ASR, but also subsequent NLP tasks. In this section, we explore another machine learning task where

contextualising ASR can potentially bring further improvement over the baseline, i.e. *semi-supervised learning* (SSL). SSL is a widely used method to improve the performance of the ASR model when little supervised speech data is available. The goal of SSL is to use large amounts of non-annotated data (i.e. data augmented with automatically generated transcripts) to boost the performance of ASR trained in a supervised manner (Imseng et al., 2012; Dey et al., 2019). Many recent studies use untranscribed data during ASR training: e.g., pre-training and self-training methods in end-to-end ASR systems (Zhang et al., 2020b) or using non-annotated data for ASR in low-resource languages (Khonglah et al., 2020). In the ATC domain, the use of untranscribed ATC data with SSL is investigated in (Kleinert et al., 2018; Srinivasamurthy et al., 2017). In this section, we investigate how to further boost the performance gains by integrating contextual knowledge from air-surveillance data into the SSL pipeline.

6.3.1. Experiments

For the experiments, we use three test sets: LiveATC (Sec. 2.4.1.2), MALORCA Prague (Sec. 2.4.1.2), and MALORCA Vienna. All of them have utterances both with and without callsigns (see Table 2.4).

ASR model CNN-TDNNF. To train the seed model, the supervised database is composed of more than 100 hours of mostly clean speech recordings from public domain resources (ATCOSIM, UWBATCC, and LDCATCC and from Air Navigation Service Providers (ANSPs) such as in previous projects (Prague and Vienna airports for MALORCA and Toulouse-Blagnac for AIRBUS) (see more in Table 2.3). All experiments are performed using the Kaldi speech recognition toolkit, employing a hybrid CNN-TDNNF model architecture. Each model consists of six convolutional layers followed by 15 factorised time-delay neural network (TDNNF) layers. We adopt Kaldi's standard chain LF-MMI recipe to train both the seed and SSL-based models. The input features include 40-dimensional MFCCs and 100-dimensional i-vectors. To augment the training data, we apply additive noise to the entire dataset, generating two additional versions with Signal-to-Noise Ratios (SNRs) of 5–10 dB and 10–20 dB, effectively tripling the dataset size. The baseline ASR system, referred to as the 'seed system', is trained on all available data for five epochs.

Unsupervised data. For the untranscribed training data, we collected recordings from two types of Very High Frequency (VHF) receivers: (i) open-source sources such as LiveATC, and (ii) high-quality VHF recordings supplied by our project partner, ReplayWell. The audio quality varies based on the distance between the

VHF receiver and the speaker (either an air traffic controller or a pilot), as well as the hardware configuration. In total, we gathered 67 hours of ATCo-pilot communication (approximately 49,000 segments) using high-fidelity VHF setups installed at Prague (LKPR) and Brno (LKTB) airports, recorded between August 2020 and January 2021. This dataset, referred to as *'unsup_vhf_67h'*, is used as an untranscribed training set and is accompanied by relevant contextual metadata.

We adopt the standard SSL framework (Khonglah et al., 2020), where a seed ASR system is first used to generate word-level recognition lattices from the untranscribed dataset (e.g., *unsup_vhf_67h*). These lattices are then combined with those generated from manually transcribed data to train a new acoustic model. In hybrid ASR systems, lattices serve as intermediate search structures that encode richer information — such as timing and alternative hypotheses — than simple 1-best outputs or n-best lists. For this study, we focus on applying lattice biasing for contextualisation within the SSL pipeline. Specifically, the first-pass lattices produced from *unsup_vhf_67h* are rescored in a second-pass decoding step using a biasing WFST. These rescored lattices, which incorporate boosted callsign probabilities from the WFST, are then used to retrain the acoustic model. This process increases the likelihood of domain-relevant entities, such as expanded callsigns, appearing in the final transcription hypotheses.

We conduct four experiments to evaluate the proposed approach for leveraging contextual knowledge in SSL. First, a baseline acoustic model (i.e., the seed model) is trained without any SSL. Next, we train a new acoustic model from scratch using SSL, where the seed model is used to generate recognition lattices for the untranscribed dataset (*unsup_vhf_67h*). In the third step, we apply lattice rescoring by composing these lattices with pre-constructed WFSTs, one per utterance, which encode contextual knowledge. This lattice biasing method incorporates a 'discount' hyperparameter that controls the influence of the contextual information during rescoring. By default, we use a discount factor of 2.0, and additionally report results using a higher discount of 6.0 to assess the effect of stronger biasing.

6.3.2. Results

Table 6.5 presents the results of SSL experiments incorporating contextual information. The benefits of SSL are more pronounced on test sets that match the untranscribed data used during training, particularly in terms of SNR and airport origin. For instance, standalone SSL yields approximately 20% relative WER reduction on the LiveATC test set and a 13.6% relative improvement on the Prague

Table 6.5.: WERs results of several ASR systems for LiveATC and MALORCA Prague and Vienna test sets. DP — the discount parameter for lattice biasing; the default DP is 2.0. The best results are **highlighted in boldface**.

System	LiveATC	Prague	Vienna
Baseline (seed model)	49.7	4.4	**6.8**
+ SSL	38.3	3.8	8.2
+ lattice re-scoring (DP: 2.0)	37.3	3.8	8.4
SSL + lattice re-scoring (DP: 6.0)	**36.4**	**3.6**	8.4

test set (see the second row of Table 6.5). In contrast, the Vienna test set exhibits a degradation in WER, likely due to a mismatch in data quality — specifically, the untranscribed VHF data used for SSL is noisier than the manually transcribed training data. Additionally, the Vienna data originates from an airport not represented in the unsupervised training set. Notably, the substantial WER gains on a challenging set like LiveATC highlight the robustness of the SSL approach under real-world conditions, where data is typically noisier and less domain-matched.

Incorporating contextual knowledge into the SSL pipeline yields an additional $\sim$5% relative WER reduction on the LiveATC and Prague test sets, beyond the improvements achieved by standard SSL alone (as shown in the third and fourth rows of Table 6.5). This gain is likely due to the presence of both LiveATC and Prague data in the transcribed and untranscribed training sets, which facilitates more effective adaptation.

We further assess the effectiveness of the proposed method using Named Entity Word Error Rate (NE-WER) on the LiveATC and MALORCA Prague test sets. NE-WER is evaluated across a range of discount parameters — used to control the strength of contextual biasing in the lattices — from 1.0 to 7.0. As illustrated in Fig. 6.3, lattice biasing consistently improves performance on the LiveATC test set, and begins to benefit the Prague test set when the discount value exceeds 4.0. The Vienna test set is omitted from the figure due to minimal variation across different discount values. Despite a WER degradation observed on Vienna when contextual information is introduced, a 7.5% relative NE-WER improvement is achieved over the '+SSL' baseline, indicating the robustness of the proposed method. A discount parameter of 5.0 yielded the best results, delivering relative NE-WER improvements of 17.5% (from 39.9% to 32.9%) on LiveATC, 14% (from 3.5% to 3.0%) on MALORCA Prague, and 7.5% on MALORCA Vienna.

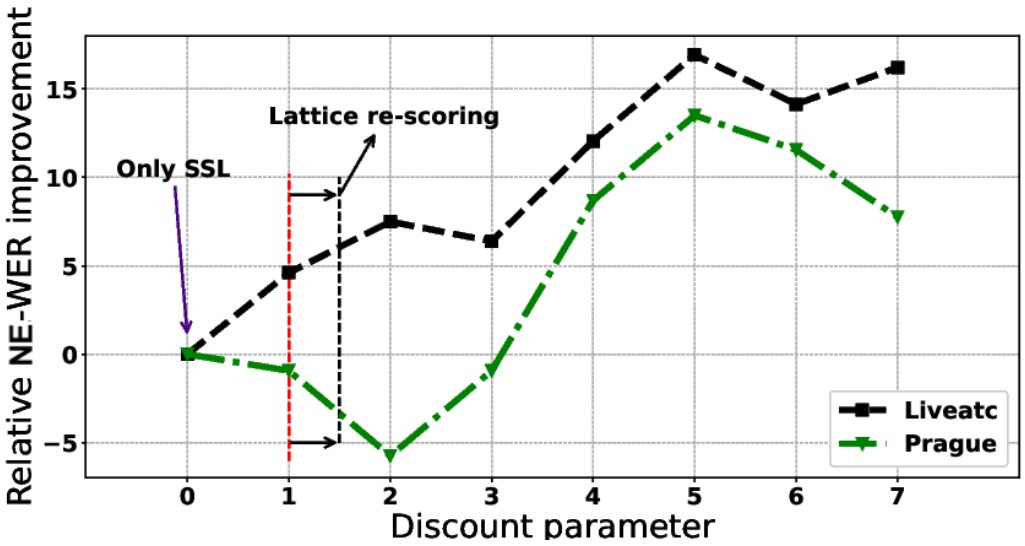

Figure 6.3.: NE-WER performance on LiveATC (noisy) and MALORCA Prague (clean) test sets for different discount parameters used at the moment of creating the biasing WFST.

6.3.3. Summary

The key innovation of the proposed approach lies in its ability to integrate contextual knowledge into the SSL framework, effectively mitigating the limitations posed by the scarcity of annotated data that is a common challenge in many ATC applications. For each utterance, we construct a biasing FST that encodes n-gram sequences of verbalised callsigns derived from surveillance data. This contextual information is incorporated into the SSL training pipeline to further enhance the acoustic model. Using this method, we observe notable improvements in NE-WER across the LiveATC, Prague, and Vienna test sets compared to standard SSL. Given the widespread availability of ATC speech and air-surveillance data across numerous European airports, this approach is highly scalable and can be readily adapted to other domains and locations.

6.4. Fast streaming ASR prototyping with pseudo labels and an integrated shallow fusion

The section is based on the publication (Thorbecke et al., 2024).

In Sec. 4.3, we showed how SF with word-level context can enhance overall WER

performance and improve the recognition of target words and named entities. In this section, we investigate how incorporating contextual information through SF impacts the performance of ASR models trained on pseudo-labelled data. The training of ASR with little to no supervised data remains an open question. Although in some cases, fine-tuning large pre-trained models can become a solution for offline ASR, the online recognition requires specific streaming models that often need to be trained from scratch. In this section, we demonstrate that streaming Transformer-Transducer (TT) models can be trained from scratch in consumer and accessible GPUs in their entirety with pseudo-labelled speech from foundational speech models. This allows training a robust ASR model in one stage and does not require large data and computational budget compared to the two-step scenario with pre-training and fine-tuning. Thus, our core contribution with the present experiments is the demonstration of fast prototyping of TT streaming ASR trained exclusively on PL data. Additionally, we show that integration of WL-ngram-LM with SF and keywords biasing with automatically generated named entities can further improve the performance of such models.

6.4.1. Transducer architecture for streaming ASR

In industrial applications, large supervised databases in target domains are not always available, thus several techniques have been proposed to develop robust ASR models with small supervised corpora: (1) data augmentation (Park et al., 2019; Bartelds et al., 2023); (2) only-audio self-supervised pre-training with large databases and fine-tuning with small corpora (Baevski et al., 2020; Conneau et al., 2020; Zuluaga-Gomez et al., 2023c); (3) pseudo-label then fine-tune, e.g., semi-supervised learning (Zhu et al., 2023; Lugosch et al., 2022; Zuluaga-Gomez et al., 2021) and weakly supervised learning (Radford et al., 2023). Most of the approaches target the attention-based encoder-decoder (AED) (Watanabe et al., 2017a) or CTC models. Even though these two architectures have shown impressive results on multiple benchmarks (e.g., Whisper (Radford et al., 2023)), they still lag in streaming settings (Prabhavalkar et al., 2023).

The Transformer-Transducer (TT) architecture (Battenberg et al., 2017; Yeh et al., 2019; Zhang et al., 2020a) is widely exploited for industrial uses that require streaming decoding because the transducer decoder naturally supports streaming (Li et al., 2021a, 2020a). However, only recently has it been demonstrated that these models can surpass standard AED systems (Sainath et al., 2020). There have been multiple breakthroughs that have made transducer training easier, such as (1) pruned transducer loss (Kuang et al., 2022), (2) better architectures, e.g., FastConformer

(Rekesh et al., 2023), and (3) from the modelling side, e.g., model pruning, sparsification (Yang et al., 2022a), and quantisation (Sainath et al., 2020). Despite this progress, little to no work has been done on fast TT model prototyping (few GPU-days) with pure pseudo-labelled data.

6.4.2. Pseudo-labeling in ASR

Semi-supervised learning (Zhang et al., 2020b; Park et al., 2020; Higuchi et al., 2021), pseudo labelling (Zavaliagkos and Colthurst, 1998; Likhomanenko et al., 2020; Hwang et al., 2022), and weakly supervised learning (Radford et al., 2023) are a family of methods aiming to partly alleviate the burden of the lack of labelled data for supervised ASR training. These methods have shown promising WER improvement in multiple settings and languages. A teacher model is trained on an audio-text paired corpus $D_l = \{X_i, Y_i\}$. Then, it is used to pseudo label a much larger unlabeled only audio corpus, $D_{pl} = \{X_i, Y_i^*\}$. Afterward, usually a smaller model (Barrault et al., 2023) can use D_l and D_{pl} for supervised training or fine-tuning (Hsu et al., 2021).

Unlike previous studies that commonly use *iterative training* (Zhang et al., 2020b; Park et al., 2020; Hwang et al., 2022), our approach avoids repeated retraining cycles, offering a more efficient alternative. A multi-stage strategy combining self- and semi-supervised learning eventually results in strong pseudo labels (PLs). In the present experiment, we focus on the performance that can be achieved "out-of-the-box" by using already available foundational speech models and training transducer models from scratch and only once. Thus, this approach minimises the overall computational cost, i.e., one-stage training and applying improvement methods, such as decoding with SF. We aim to reveal the potential of the foundational speech model in PL generation and demonstrate what performance can be reached with minimal training efforts. The overall proposed approach is illustrated in Fig. 6.4.

However, PLs are often noisy and limited by the quality of the teacher model, which can lead to suboptimal performance in the resulting models. This issue can be addressed either by filtering out the most unreliable samples or by enhancing the teacher model, for example, by increasing its size to improve its overall quality.[51] Several approaches to improve the PL quality include improving loss functions (Zhu et al., 2023; Gao et al., 2023), pairing online and offline models at training time (Higuchi et al., 2021), and continuous single-language (Likhomanenko et al.,

[51] We assume that a larger model, trained under the same conditions and with increased data, will attain lower WERs.

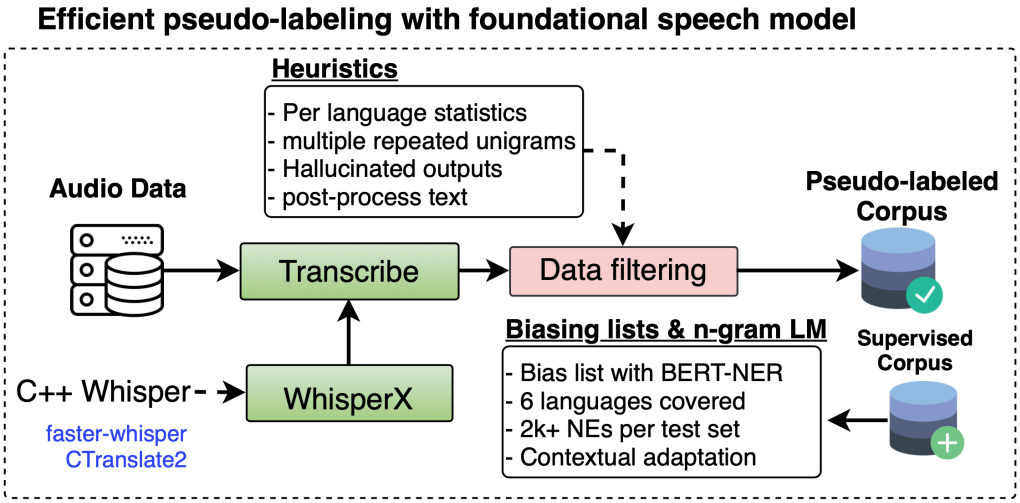

Figure 6.4.: Proposed approach for efficient and fast streaming ASR prototyping with pseudo-labelled data.

2022; Berrebbi et al., 2022) and multilingual pseudo-labelling setting (Lugosch et al., 2022).

6.4.3. Knowledge Distillation with Large Models

Knowledge distillation (KD), also known as teacher-student training (Watanabe et al., 2017b), is a widely recognised technique for transferring knowledge from a larger model (the *Teacher*) to a smaller model (the *Student*) (Hinton et al., 2015). In this approach, the teacher model is first trained either using supervised learning with ground-truth labels (Takashima et al., 2018) or through self-supervised methods. The student model is subsequently trained using the posterior probabilities generated by the pre-trained teacher (Chebotar and Waters, 2016). Prior studies have explored KD for CTC models (Takashima et al., 2018), AED models using Whisper (Radford et al., 2023; Gandhi et al., 2023; Ferraz et al., 2024), and Transducer models (Panchapagesan et al., 2021). Additionally, KD has been used to compress offline transducer models into online versions (Kurata and Saon, 2020) and to distill knowledge from self-supervised models (Yang et al., 2022b).

Our work focuses on sequence-level KD, where we use the one-best hypothesis from the teacher model instead of relying on its full posterior distribution. This method offers several advantages: (1) it eliminates the need to cache the teacher model or its outputs in memory; (2) it requires no modifications to existing ASR training pipelines; and (3) it enables faster ASR training compared to traditional teacher-student KD approaches. Additionally, we can take advantage of highly optimised inference pipelines — such as model quantisation — for generating PLs, as demon-

strated by WhisperX (Bain et al., 2023). Altogether, this approach supports fast pseudo-labelling, making it well-suited for rapid prototyping in standard industrial applications.

6.4.4. Experiments

As the Teacher model, we choose the Whisper model (Radford et al., 2023) because of its strong benchmark performance across a variety of languages. Moreover, Whisper offers models at multiple parameter scales, allowing us to investigate how the quality of PLs from different-sized models influences downstream TT models. We pseudo-label each training corpus with 5 Whisper model sizes: whisper-tiny, base, small, medium, and large-v3. Across all experiments, we employ the WhisperX pipeline (Bain et al., 2023), which consists of the following components: (1) voice activity detection for segmenting long-form audio; (2) batch processing of multiple segments to enable efficient inference; and (3) model quantisation of Whisper combined with a C++-based FasterWhisper implementation[52], which uses CTranslate2 for fast decoding[53]; (4) model inference and word level alignment.

We conducted all our experiments on the CommonVoice dataset (Sec. 2.4.2.4) on the six following languages: Catalan (CA), English (EN), German (DE), French (FR), Spanish (ES), and Italian (IT). To the best of our knowledge,[54] the CommonVoice data was not used for training Whisper model and can be used for zero-shot evaluation. In our case, it is an important point, as using unseen data for generating PLs provides a more realistic estimation of the proposed approach performance.[55]

Data filtering. To filter out noisy and hallucinated PLs, we adapted several data selection heuristics (H) that are applied for every training corpora; similar heuristics are proposed in (Barrault et al., 2023):

1. $H1$**:** remove PL if composed of the same unigram three or more times.
2. $H2$**:** compute maximum word length from supervised training corpus and remove utterances with one or more PLs larger than the max threshold. See the thresholds proposed per each language in Table 6.6; the threshold is consider-

[52] https://github.com/SYSTRAN/faster-whisper

[53] https://github.com/OpenNMT/CTranslate2/

[54] According to the discussions in the official OpenAI-Whisper GitHub repository: https://github.com/openai/whisper/discussions/349, https://github.com/openai/whisper/discussions/2305.

[55] Although officially the CommonVoice data is not included in the training data for Whisper, we realise the possibility of some CommonVoice data still being seen by the model through other sources and in a small amount.

Table 6.6.: Maximum number of characters allowed in each pseudo-labeled word with Whisper.

CA	EN	DE	FR	ES	IT
16	16	30	20	25	22

ably higher in the languages with compound words, i.e. German (DE).

3. $H3$**:** compute $word_{ratio}$, i.e. number of words divided by utterance duration (seconds), and filter out samples with $word_{ratio}$ less than 1 or more than 4.

4. $H4$**:** verbalise all the numbers from the PLs, remove punctuation and normalise following the CommonVoice recipe in Lhotse (Żelasko et al., 2021).

ASR model Zipformer. For the ASR models for each language, we trained the same Zipformer described in detail in Sec. 4.3.1 as *"ASR model 2"*. We trained each model for 30 epochs on a single RTX 3090 GPU with only PLs. Training takes between 1 and 2 days. All the models proposed in this work can perform streaming decoding. This is achieved by performing chunk-wise multi-chunk training. During training, we use causal masking of different sizes to enable streaming decoding under different low-latency configurations (Swietojanski et al., 2023; Kumar et al., 2025a). Specifically, we rely on two lists: chunk-size={640ms,1280ms,2560ms,full} and left-context-frames={64,128,256,full}.[56] At training time, we randomly select the chunk size and the left context chunks for each batch. This enables the final model to work on a wide variety of streaming settings. At test time, we select 13 different decoding configurations ranging from 320 ms[57] to 2560 ms chunks (see all 13 decoding configurations listed in Appendix A.1).

LMs. For the LMs for each language, we used the same LMs as described in Sec. 4.3.1

Regularisation with supervised data. Besides the models fully trained with PLs, we perform experiments where along with PLs we mix in 100h of randomly selected supervised data from the train set D_l during training. We compute mixing weights between D_l and D_{pl} so each training batch contains at least one sample from D_l. This is achieved with *CutSet.Mux* function from Lhotse (Żelasko et al., 2021).[58] All the experiments that uses PL and supervised data are denoted with ***+sup. [100h]***, otherwise, the model is trained with PL only. As an ablation

[56] The effective number of left context chunks is computed as $left\_context\_frames//chunk\_size$.

[57] Decode chunk size of 320ms is more challenging as it has not been used during training.

[58] It lazily loads two or more datasets and mixes them on-the-fly according to pre-defined mixing weights.

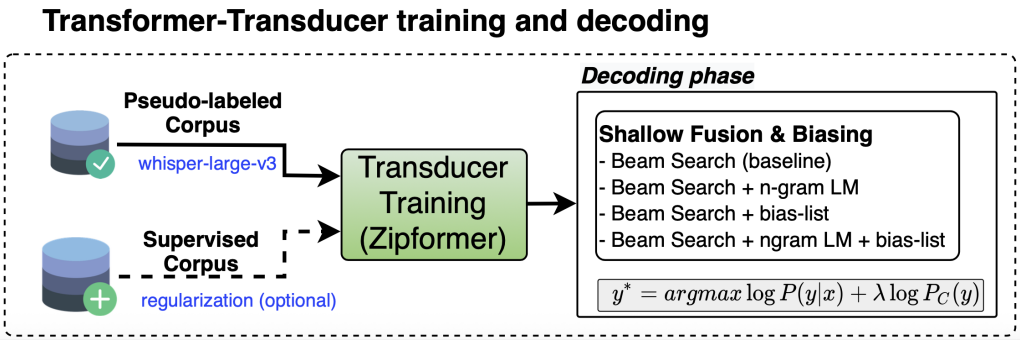

Figure 6.5.: Transducer models are further improved via shallow fusion of n-gram LMs and contextual biasing of target named entities.

experiment, we also test the performance by scaling up supervised data to 200h and 400h when using the weakest foundational speech model, i.e., whisper-tiny. This experiment aims to (1) compensate for very low-quality PLs, and (2) demonstrate that Whisper PLs (from the largest models) are of sufficient quality for transducer training without any supervised data.

Contextual biasing to improve the performance of models trained with PLs. Leveraging more text data and context information with language model and keywords integration can considerably improve ASR performance. Since in our setup, we assume that we have zero (or very little) supervised data, using extra unpaired text data would not contradict the original constraints. At the same time, relying mainly on PLs, we see text knowledge integration as an opportunity to make our models more reliable and robust. To improve the performance of models trained with PLs, we apply SF with word-level context information as it is shown in Fig. 6.5 and described above as Method 2.2 *"SF of keywords and context entities"*, Method 2.3 *"SF of a WL-ngram-LM as a context trie"*, and Method 2.4 *"SF with unifying WL-ngram-LM and keyword biasing in a single trie"* in Sec. 4.1.3 and Sec. 4.3.

6.4.5. Results

We report our results in two parts. The first part is the offline performance of the models and the effect of regularisation achieved with additional supervised data. The second part is the evaluation of the streaming models and the impact of SF.

Offline models and regularisation with supervised data. The blue graphs in Fig. 6.6 depict the offline performance for Zipformer models trained from scratch on PL data only and for six languages. Offline evaluation determines the upper bound WERs achievable by training with PLs of varying qualities. As the size of

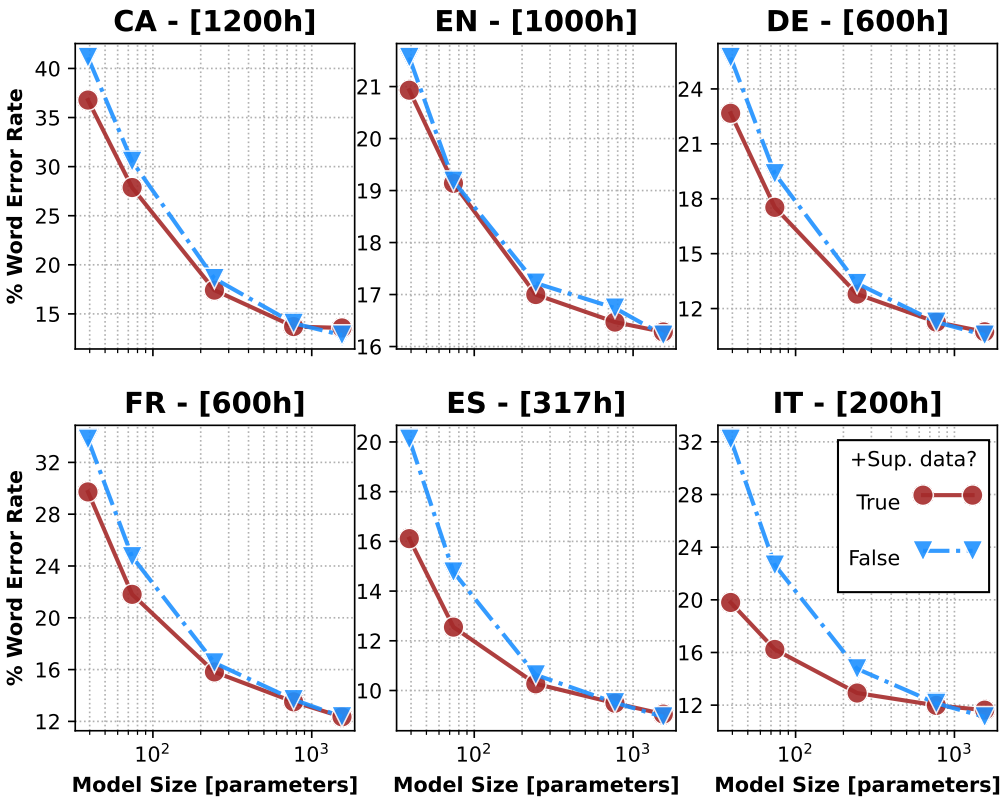

Figure 6.6.: WERs for offline Zipformer models on six languages of CommonVoice. Models are trained with pseudo-labels from different Whisper model sizes (blue graphs). Adding 100h of supervised data during training (red graph) regularizes the training up to models with 700M params, especially for languages with less data.

the Whisper Model increases (from *tiny* to *large-v3*, as shown on a log-scaled x-axis), there is a corresponding improvement in WERs, also on a log-scale. The best performance is observed for ES, with the least favourable results for EN. These results show that our approach adapts across a spectrum of PL data quantities and qualities, ranging from 200h for IT to over 1000h for CA and EN.

Red graphs in Fig. 6.6 show WERs for offline models that, along with PLs, include a small amount of supervised data, up to 100h, for regularisation. This strategy proves to be beneficial in cases with noisier PLs, particularly for smaller Whisper models like Whisper-tiny, Whisper-base, and Whisper-small when WER goes down for all the languages. The benefits, however, decrease or are absent with more accurate PLs generated by larger models, such as Whisper-medium and Whisper-large-v3. Thus, with our results on six languages, we can conclude that when supervised data is available, regularisation is recommended for models with weak PLs and can be omitted with strong PLs. The results with 100h regularisation are also available in Table A.1 for offline models and are consistent with the performance on streaming

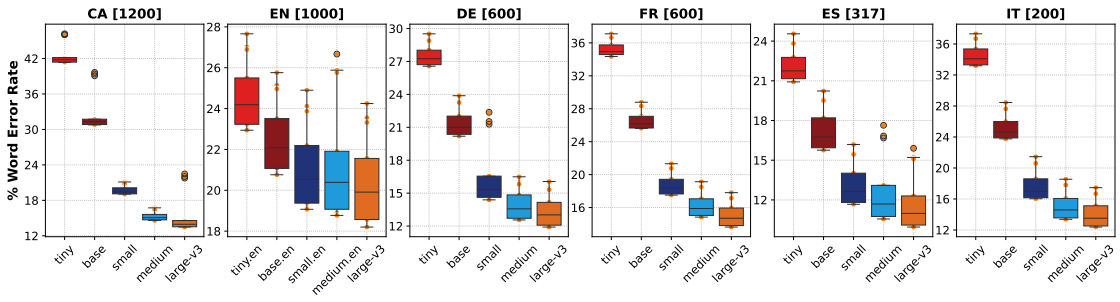

Figure 6.7.: Box plots of WERs for six languages of CommonVoice. Streaming Zipformer models are trained from scratch, with only PLs generated with different Whisper model sizes. Each box denotes 13 decoding configurations, ranging from challenging (320ms chunk with limited left context) to more relaxed (2560ms chunk with full left context) streaming settings. (Note different WER scaling on the y-axis.)

models reported in Table 6.7.

Streaming models and SF with n-gram LM. Fig. 6.7 lists the streaming decoding results across six CommonVoice languages, testing 13 different decoding configurations. We establish an upper performance bound with models tested in non-streaming mode and also include a box plot for each TT model trained with PLs derived from various Whisper model sizes. The results show how model performance can fluctuate under different streaming conditions, with smaller chunk sizes or limited left context posing greater challenges. The results with the configuration with cs=320ms and lf=2.5s are also reported in Tables 6.7 and A.2 and demonstrate consistently better performance when the full left context is used. This tendency stays independent of language and SF.

Performance on different models and languages with SF is presented in Table 6.7. Zipformer models in the table are our baselines, i.e., streaming models trained on PL data. We use the further improved models with an additional 100h of supervised data for decoding with SF. For all the languages, we can see a WER decrease when decoded with an external LM. The WER also always improves when context biasing with NEs is introduced. It is an important observation since all NEs are extracted automatically with no human supervision involved. Moreover, all biasing lists are rather long, which often is an obstacle to improvement when biasing methods are used (Chen et al., 2019). Nevertheless, our approach proves to work on biasing lists of large sizes as well. According to our results, external LM and context NE fusions are complementary methods, gaining the best WER when they are combined during decoding.

The improvement with SF is consistent through the languages and has the biggest

Table 6.7.: WERs for streaming evaluation with n-gram LM and bias-lists (BL). Listed on four CommonVoice languages and two decoding configurations. The Zipformer models are trained with pseudo-labelled data from different Whisper models and 100h of supervised data ("sup. [100h]") from the original train set. All experiments show additive WERs improvement when adding either (or both) n-gram LM or biasing lists.

	cs=320ms;lf=2.5s				**cs=320ms;lf=∞**			
Experiment	CA	DE	ES	IT	CA	DE	ES	IT
Whisper-tiny ($39M$)								
Zipformer ($70M$)	46.2	29.5	24.5	37.3	46.0	28.9	23.8	36.7
+sup. [100h]	38.7	27.0	21.6	26.7	38.3	26.4	21.0	25.9
+n-gram LM	38.6	26.2	21.2	25.7	38.2	25.5	20.5	25.0
+bias-list	34.4	26.3	21.3	25.5	33.9	25.7	20.5	24.7
+n-gram LM+(BL)	34.0	25.7	21.0	24.7	33.6	25.1	20.2	23.9
Whisper-base ($74M$)								
Zipformer ($70M$)	39.7	23.9	20.2	28.4	39.4	23.3	19.5	27.6
+sup. [100h]	29.9	22.2	18.3	23.0	29.6	21.5	17.6	22.3
+n-gram LM	29.4	21.4	17.8	22.2	29.1	20.8	17.1	21.5
+bias-list	26.0	21.6	17.6	22.0	25.6	21.0	16.9	21.2
+n-gram LM+(BL)	25.7	21.0	17.2	21.3	25.2	20.4	16.5	20.7
Whisper-small ($244M$)								
Zipformer ($70M$)	21.1	22.4	16.2	21.5	20.8	21.3	15.4	20.6
+sup. [100h]	20.1	17.8	16.0	20.1	19.8	17.2	15.3	19.3
+n-gram LM	19.6	17.1	15.5	19.3	19.3	16.5	14.8	18.5
+bias-list	17.7	17.2	15.4	19.3	17.3	16.6	14.7	18.5
+n-gram LM+(BL)	17.4	16.7	15.0	18.8	17.1	16.1	14.3	17.9
Whisper-medium ($769M$)								
Zipformer ($70M$)	16.7	16.5	17.6	18.6	16.4	15.8	16.7	17.8
+sup. [100h]	16.5	16.6	14.9	19.4	16.2	15.8	14.3	18.4
+n-gram LM	16.1	15.8	14.6	18.6	15.8	15.1	13.9	17.7
+bias-list	14.8	16.0	14.6	18.6	14.5	15.3	13.9	17.7
+n-gram LM+(BL)	14.6	15.5	14.3	18.1	14.3	14.8	13.6	17.2
Whisper-large-v3 ($1.5B$)								
Zipformer ($70M$)	22.5	16.1	15.9	17.5	21.8	15.3	15.1	16.7
+sup. [100h]	16.6	16.3	14.6	18.4	16.4	15.6	14.0	17.6
+n-gram LM	16.2	15.6	14.2	17.6	16.0	14.9	13.6	16.8
+bias-list	15.0	15.8	14.1	17.8	14.8	15.1	13.5	17.0
+n-gram LM+(BL)	14.8	15.3	13.8	17.2	14.5	14.6	13.2	16.4
Baseline streaming Zipformer (only supervised data)								
Zipformer ($70M$)	7.8	13.8	13.5	17.5	7.6	13.1	12.8	16.6

impact when models are trained on weaker PLs generated from the smaller Whisper models. This behaviour is expected, as the models that saw less training data have more potential to still benefit from any additional data given during regularisation and/or decoding. On the other hand, the improvement decreases with the PLs generated by Whisper-medium and Whisper-large-v3.

Another remarkable observation is that the models trained on PLs are more competitive with the models trained on the supervised data only when less training data is given. For example, CA language has 1200h of training data, and supervised models are considerably winning over the PL models even after all the improvements we introduce: 7.8% VS 14.8% for supervised and PL models correspondingly.[59] When double less training data is used for DE language, i.e., 600h, the difference is less prominent but still considerable: 13.8% VS 15.3% for supervised and PL models correspondingly. When the amount of training data is further reduced to 317h for ES and 200h for IT, we observe either little or no degradation from supervised models to PL models: 13.5% VS 13.8% for ES and 17.5% VS 17.2% for IT for supervised and PL models correspondingly. These results illustrate well the advantages and strengths of the proposed framework and methods for the low-resource scenarios. Due to space constraints, in Table 6.7, we show the performance only on four languages; SF impact on offline models for all six languages can be found in Table A.1.

6.4.6. Summary

In this section, we propose a framework to meet the challenge of training streaming ASR systems with few-to-no supervised data by leveraging PLs from foundational speech models. We conduct multiple experiments on the efficacy of PL-based Transducer models across various dimensions, including offline and chunk-wise decoding for streaming applications, the influence of the Teacher model's size and the regularisation of supervised data on the performance of the resulting Transducer models. We introduce robust heuristics to filter out unreliable and hallucinated PLs and show that Transducer models can be effectively trained from scratch on noisy PLs.

Additionally, we demonstrate that decoding with the SF of external n-gram LM and automatically generated named entities always further improves the performance of models, independent of the quality of PLs. Thus, SF proves to be useful not only for local improvement but as a part of a large training pipeline, even when only automatically generated named entities are used.

[59] Here and below in this paragraph, we report WERs for the configuration with cs=320ms and lf=2.5s. We observe the same tendency in the results with the other configuration as well.

6.5. Chapter summary

Context integration can benefit not only the ASR performance but also the ASR related and downstream tasks, such as NER and SSL. The experiments described in this Chapter reveal its applicability for (1) the callsign detection in the ATC domain, (2) iterative SSL, and (3) fast prototyping for streaming ASR. While for SSL, we used the same methods of contextual biasing as for the ASR (lattice rescoring and SF), for the NER task, we introduced an additional NLP-boosting method for matching the extractor's outputs to the context data. The NLP-boosting leads to a considerable improvement in the final detection of target entities that can be further improved by combining it with the ASR biasing. We demonstrate the efficiency of contextualisation for pipeline ASR-NER (with both data-driven and rule-based entity extractors), as well as for End-to-End systems. Finally, the experiments on SSL for streaming ASR and with the CommonVoice dataset showed that even when named entities are generated automatically with an NER extractor, their boosting is still useful for the overall ASR improvement.

7. Conclusion

In this thesis, we provide an overview of various approaches for the integration of contextual knowledge into ASR systems and their related applications. We focus on those contextualisation methods when only text context data is available and without retraining the base ASR model. We aim to enhance overall ASR performance, with a particular focus on accurately recognising key target entities — namely, (1) word sequences (n-grams) that are more likely within a given context, and (2) named entities (NEs). To achieve this, we propose context-aware ASR biasing methods that leverage textual information to steer ASR model predictions toward the desired target entities. These methods are explored across a range of settings, including various ASR model types, different levels of contextual integration, both online and offline ASR, and ASR-related tasks.

7.1. Contributions

Each chapter (except the background chapter 2) presents and analyses methods of contextual integration aiming to enhance ASR performance and the recognition of the target entities, but applied in different setups: (1) pipeline (hybrid) ASR, (2) End-to-End ASR, (3) ASR-related tasks (named entity recognition (NER) from audio, semi-supervised learning (SSL), and streaming ASR training with pseudo-labelled data), (4) domain adaptation. Thus, our first contribution is a comprehensive evaluation and analysis of various contextual methods, serving as a practical guide for selecting the most suitable approach.

For hybrid ASR, we describe three different methods: injecting context through decoding lattices, into a decoding graph, or inside the grammar FST by adding new entities and boosting the existing ones. We prove the usefulness of the methods for both frequent and rare (or OOV) words. Another contribution is the proposed algorithm, along with its GPU-based implementation, for dynamic contextual biasing in real-time ASR hypotheses. Taking context words and word sequences as input, the method adjusts the weights of target arcs in the decoding graph in a dis-

tributed manner, without requiring lattice generation. This approach enables rapid and flexible adaptation to the current context and represents a step toward tighter integration between inference and decoding.

Our main contribution to the End-to-End contextualisation is the extension of the use of the Aho-Corasick (AC) algorithm to integrate word-level n-gram LMs. We propose an efficient contextual adaptation approach using a word-level n-gram LM and keyword biasing within a single context trie. The proposed method enables faster decoding compared to standard SF with neural network language models, while maintaining competitive WER and real-time factor (RTF) efficiency. We demonstrate that integrating n-gram LMs with keyword biasing into a unified context graph not only improves NE recognition accuracy but also enhances WER for non-biased entities, including OOV words. Experiments conducted across four languages and four datasets validate the effectiveness of our approach in both in-domain and out-of-domain scenarios. We show that SF remains effective not only when a predefined list of target ground-truth entities is available (even when mixed with distractors), but also when biasing entities are generated fully automatically. Additionally, we (1) benchmark multiple SF strategies across four languages and three datasets and (2) provide preliminary results on domain adaptation with text data and pseudo-audio representations.

For the NE extraction from speech, we contribute by proposing an iterative approach that leverages contextual data by combining ASR and NLP modules. For the Air Traffic Control (ATC) use case, we first boost the recognition probability of active callsigns within the ASR system using FST-boosting. Second, we enhance the ASR outputs through NLP-boosting, aiming to correct predicted callsigns that are not found in the surveillance data. We demonstrate that this two-step approach can be successfully applied with cascades of ASR-NER systems, as well as with End-to-End multitask models.

For the SSL, we contribute by proposing a two-step approach to incorporate contextual knowledge during iterative semi-supervised training, aiming to reduce ASR error rates specifically in segments of the utterance that convey key information. A seed ASR system generates word recognition lattices from untranscribed data, which are then combined with lattices derived from manually transcribed data to train a new acoustic model. The lattices from the untranscribed data are further enhanced using a lattice rescoring technique that emphasises available contextual entities. These rescored lattices are subsequently used to retrain the acoustic model. Experimental results demonstrate that contextual SSL provides clear benefits in 'unseen' domains compared to conventional standalone SSL.

We propose a framework to address the challenge of training streaming ASR systems with little to no supervised data by leveraging pseudo-labels (PLs) generated from foundational speech models. Our results demonstrate that Transformer-Transducer models can be successfully trained from scratch using noisy pseudo-labels. Moreover, we further enhance the performance of these weakly supervised models through SF with an external n-gram language model and automatically extracted named entities.

Finally, the majority of our experiments are conducted on ATC data, leveraging radar surveillance information as contextual knowledge for biasing. We regard our results on ASR contextualisation in the ATC domain as a significant contribution toward the advancement of ASR-assisted air traffic control, aimed at enhancing communication efficiency and reducing the risk of miscommunication.

Summarised contributions:

- improve recognition of target entities by integrating contextual knowledge;
- improve recognition of frequent as well as rare and OOV words;
- propose an implementation algorithm for the contextual biasing of decoding graph in the online GPU decoding;
- propose an efficient contextual adaptation method using a word-level n-gram LM and keyword biasing within a single context trie;
- propose a method of combining ASR and NLP boosting for enhanced entity extraction;
- propose an improved semi-supervised learning approach by leveraging contextual knowledge;
- propose a framework for full-stack rapid development of ASR streaming solutions from scratch with low-to-zero supervised resources;
- propose a method for domain adaptation with text data and pseudo-audio representations;
- overview static and dynamic contextualisation methods for hybrid and End-to-End models;
- evaluate contextualisation methods on conversational datasets from different domains, including air traffic communication, banking, healthcare, insurance, and earnings, and different languages — English, German, French, Spanish.

7.2. Future directions

Integrating context and adapting to new domains using text-only data remains a significant challenge for End-to-End ASR models and represents one of their key weaknesses compared to hybrid systems. The primary difficulty lies in the unified architecture of End-to-End models, where acoustic and language modelling components are jointly optimised, making it difficult to incorporate additional text data specifically for the language component. We believe that both *context integration* and *domain adaptation* in End-to-End ASR still offer substantial opportunities for improvement and future research.

The magnitude of the resulting WER improvement from integrating subword-level external neural network language models (NNLMs) into ASR decoding depends strongly on the ASR architecture and the predictor network's specific label context modelling. For instance, in Transducer-based models, the predictor acts as a strong internal language model (ILM). If this internal bias is not effectively estimated and compensated for – using methods such as ILM subtraction or Density Ratio – the gains from an external LM may be significantly diminished. Moreover, this approach comes with several drawbacks: (1) training NNLMs requires substantial resources, including time, computational power, and a large amount of training text, and (2) ELM fusion significantly slows down the decoding process, making it unsuitable for real-time (online) applications. Shallow fusion (SF) methods are also constrained by: (1) the initial hypotheses generated by the acoustic model before any biasing is applied, and (2) the need to tune the biasing score for each use case. Other context-injection techniques, such as cold fusion, deep fusion, and conditioned encoders, tend to yield better performance but are computationally intensive and require specialised training procedures. Our findings suggest that effective domain adaptation benefits from adapting both the encoder and decoder components. However, domain adaptation using text-only data in End-to-End models still falls significantly short of the performance achieved with real audio data. Pseudo-audio features offer limited benefit for encoder adaptation compared to genuine audio input.

One promising direction not explored in this thesis, but with potential to enhance text-based context integration and domain adaptation in End-to-End ASR, is factorised ASR architectures. These architectures decouple the language model component, enabling the application of various language model adaptation techniques in a more straightforward and modular way. While factorised modelling improves accessibility to individual ASR components, making their modification more flexible and targeted, it can also be seen as a step back toward hybrid ASR systems, where

components such as the acoustic and language models are trained separately.

References

2011. All clear phraseology manual. In *Eurocontrol, Brussels, Belgium*. "[Online; accessed 10-September-2021]".

Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, et al. 2025. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*.

Alfred V Aho and Margaret J Corasick. 1975. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6):333–340.

Petar S Aleksic, Mohammadreza Ghodsi, Assaf Hurwitz Michaely, Cyril Allauzen, Keith B Hall, Brian Roark, David Rybach, and Pedro J Moreno. 2015. Bringing contextual information to google speech recognition. In *Interspeech*, pages 468–472.

Cyril Allauzen, Michael Riley, and Johan Schalkwyk. 2009. A generalized composition algorithm for weighted finite-state transducers. In *Interspeech*, pages 1203–1206.

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. Openfst: A general and efficient weighted finite-state transducer library. In *International Conference on Implementation and Application of Automata*, pages 11–23. Springer.

Junyi Ao, Rui Wang, Long Zhou, Chengyi Wang, Shuo Ren, Yu Wu, Shujie Liu, Tom Ko, Qing Li, Yu Zhang, et al. 2022a. Speecht5: Unified-modal encoder-decoder pre-training for spoken language processing. In *ACL 2022*, pages 5723–5738.

Junyi Ao, Ziqiang Zhang, Long Zhou, Shujie Liu, Haizhou Li, Tom Ko, Lirong Dai, Jinyu Li, Yao Qian, and Furu Wei. 2022b. Pre-training transformer decoder for

end-to-end asr model with unpaired speech data. *arXiv preprint arXiv:2203.17113*.

Rosana Ardila, Megan Branson, Kelly Davis, Michael Kohler, Josh Meyer, Michael Henretty, Reuben Morais, Lindsay Saunders, Francis Tyers, and Gregor Weber. 2020. Common voice: A massively-multilingual speech corpus. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4218–4222.

Jordi Armengol-Estapé, Casimiro Pio Carrino, Carlos Rodriguez-Penagos, Ona de Gibert Bonet, Carme Armentano-Oller, Aitor Gonzalez-Agirre, Maite Melero, and Marta Villegas. 2021. Are multilingual models the best choice for moderately under-resourced languages? A comprehensive assessment for Catalan. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4933–4946, Online. Association for Computational Linguistics.

A Arunkumar and Srinivasan Umesh. 2022. Joint encoder-decoder self-supervised pre-training for asr. *Proc. Interspeech 2022*, pages 3418–3422.

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

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. 2016. End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4945–4949. IEEE.

Ye Bai, Jiangyan Yi, Jianhua Tao, Zhengqi Wen, Zhengkun Tian, and Shuai Zhang. 2021. Integrating knowledge into end-to-end speech recognition from external text-only data. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:1340–1351.

Max Bain, Jaesung Huh, Tengda Han, and Andrew Zisserman. 2023. WhisperX: Time-Accurate Speech Transcription of Long-Form Audio. In *Proc. INTERSPEECH 2023*, pages 4489–4493.

Loïc Barrault, Yu-An Chung, Mariano Cora Meglioli, David Dale, Ning Dong, Paul-Ambroise Duquenne, Hady Elsahar, Hongyu Gong, Kevin Heffernan, John Hoffman, et al. 2023. Seamlessm4t-massively multilingual & multimodal machine translation. *arXiv preprint arXiv:2308.11596*.

Martijn Bartelds, Nay San, Bradley McDonnell, Dan Jurafsky, and Martijn Wieling. 2023. Making more of little data: Improving low-resource automatic speech recognition using data augmentation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 715–729, Toronto, Canada. Association for Computational Linguistics.

Eric Battenberg, Jitong Chen, Rewon Child, Adam Coates, Yashesh Gaur Yi Li, Hairong Liu, Sanjeev Satheesh, Anuroop Sriram, and Zhenyao Zhu. 2017. Exploring neural transducers for end-to-end speech recognition. In *2017 IEEE automatic speech recognition and understanding workshop (ASRU)*, pages 206–213. IEEE.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Dan Berrebbi, Ronan Collobert, Samy Bengio, Navdeep Jaitly, and Tatiana Likhomanenko. 2022. Continuous pseudo-labeling from the start. In *The Eleventh International Conference on Learning Representations*.

Mrinmoy Bhattacharjee, Petr Motlicek, Iuliia Nigmatulina, Hartmut Helmke, Oliver Ohneiser, Matthias Kleinert, and Heiko Ehr. 2023. Customization of automatic speech recognition engines for rare word detection without costly model re-training. *13th SESAR Innovation Days 2023, SIDS 2023*.

Mrinmoy Bhattacharjee, Iuliia Nigmatulina, Amrutha Prasad, Pradeep Rangappa, Srikanth Madikeri, Petr Motlicek, Hartmut Helmke, and Matthias Kleinert. 2024. Contextual biasing methods for improving rare word detection in automatic speech recognition. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 12652–12656. IEEE.

Hugo Braun, Justin Luitjens, Ryan Leary, Tim Kaldewey, and Daniel Povey. 2020. Gpu-accelerated viterbi exact lattice decoder for batched online and offline speech recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7874–7878. IEEE.

Rudolf A Braun, Srikanth Madikeri, and Petr Motlicek. 2021. A comparison of methods for oov-word recognition on a new public dataset. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5979–5983. IEEE.

Antoine Bruguier, Rohit Prabhavalkar, Golan Pundak, and Tara N Sainath. 2019. Phoebe: Pronunciation-aware contextualization for end-to-end speech recognition. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6171–6175. IEEE.

William Chan, Navdeep Jaitly, Quoc V Le, and Oriol Vinyals. 2015. Listen, attend and spell. *arXiv preprint arXiv:1508.01211*.

Feng-Ju Chang, Jing Liu, Martin Radfar, Athanasios Mouchtaris, Maurizio Omologo, Ariya Rastrow, and Siegfried Kunzmann. 2021. Context-aware transformer transducer for speech recognition. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 503–510. IEEE.

Yevgen Chebotar and Austin Waters. 2016. Distilling Knowledge from Ensembles of Neural Networks for Speech Recognition. In *Proc. Interspeech 2016*, pages 3439–3443.

Guoguo Chen, Shuzhou Chai, Guanbo Wang, Jiayu Du, Wei-Qiang Zhang, Chao Weng, Dan Su, Daniel Povey, Jan Trmal, Junbo Zhang, et al. 2021. Gigaspeech: An evolving, multi-domain asr corpus with 10,000 hours of transcribed audio. In *Interspeech*.

Shuo Chen, Hartmut Helmke, Robert M Tarakan, Oliver Ohneiser, Hunter Kopald, and Matthias Kleinert. 2023. Effects of language ontology on transatlantic automatic speech understanding research collaboration in the air traffic management domain. *Aerospace*, 10(6):526.

Xie Chen, Zhong Meng, Sarangarajan Parthasarathy, and Jinyu Li. 2022. Factorized neural transducer for efficient language model adaptation. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8132–8136. IEEE.

Zhehuai Chen, Mahaveer Jain, Yongqiang Wang, Michael L Seltzer, and Christian Fuegen. 2019. End-to-end contextual speech recognition using class language models and a token passing decoder. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6186–6190. IEEE.

Zhehuai Chen, Justin Luitjens, Hainan Xu, Yiming Wang, Daniel Povey, and Sanjeev Khudanpur. 2018. A gpu-based wfst decoder with exact lattice generation. In *Proc. of Interspeech*, pages 2212–2216.

Chung-Cheng Chiu and Colin Raffel. 2017. Monotonic chunkwise attention. *arXiv preprint arXiv:1712.05382*.

Jan Chorowski and Navdeep Jaitly. 2016. Towards better decoding and language model integration in sequence to sequence models. *arXiv preprint arXiv:1612.02695*.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in neural information processing systems*, pages 577–585.

Alexis Conneau, Alexei Baevski, Ronan Collobert, Abdelrahman Mohamed, and Michael Auli. 2020. Unsupervised cross-lingual representation learning for speech recognition. *arXiv preprint arXiv:2006.13979*.

José Manuel Cordero, Manuel Dorado, and José Miguel de Pablo. 2012. Automated speech recognition in atc environment. In *Proceedings of the 2nd International Conference on Application and Theory of Automation in Command and Control Systems*, pages 46–53.

Steven Davis and Paul Mermelstein. 1980. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on acoustics, speech, and signal processing*, 28(4):357–366.

Miguel Del Rio, Peter Ha, Quinten McNamara, Corey Miller, and Shipra Chandra. 2022. Earnings-22: A Practical Benchmark for Accents in the Wild. *arXiv preprint arXiv:2203.15591*.

Estelle Delpech, Marion Laignelet, Christophe Pimm, Céline Raynal, Michal Trzos, Alexandre Arnold, and Dominique Pronto. 2018. A real-life, french-accented corpus of air traffic control communications. In *Language Resources and Evaluation Conference (LREC)*.

Yan Deng, Rui Zhao, Zhong Meng, Xie Chen, Bing Liu, Jinyu Li, Yifan Gong, and Lei He. 2021. Improving rnn-t for domain scaling using semi-supervised training with neural tts. In *Interspeech*, pages 751–755.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Subhadeep Dey, Petr Motlicek, Trung Bui, and Franck Dernoncourt. 2019. Exploiting semi-supervised training through a dropout regularization in end-to-end speech recognition. *Proc. Interspeech 2019*, pages 734–738.

Hans JGA Dolfing and I Lee Hetherington. 2001. Incremental language models for speech recognition using finite-state transducers. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2001. ASRU'01.*, pages 194–197. IEEE.

Jennifer Drexler and James Glass. 2019. Subword regularization and beam search decoding for end-to-end automatic speech recognition. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6266–6270. IEEE.

Jennifer Drexler Fox and Natalie Delworth. 2022. Improving contextual recognition of rare words with an alternate spelling prediction model. In *Interspeech*, pages 3914–3918.

Paul-Ambroise Duquenne, Holger Schwenk, and Benoît Sagot. 2023. Sonar: sentence-level multimodal and language-agnostic representations. *arXiv preprint arXiv:2308.11466*.

Amin Fazel, Wei Yang, Yulan Liu, Roberto Barra-Chicote, Yixiong Meng, Roland Maas, and Jasha Droppo. 2021. Synthasr: Unlocking synthetic data for speech recognition. In *Interspeech 2021*, pages 896–900.

Thomas Palmeira Ferraz, Marcely Zanon Boito, Caroline Brun, and Vassilina Nikoulina. 2024. Multilingual distilwhisper: Efficient distillation of multi-task speech models via language-specific experts. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

GB Flores d'Arcais, Robert Schreuder, and Ge Glazenborg. 1985. Semantic activation during recognition of referential words. *Psychological research*, 47(1):39–49.

G David Forney. 1973. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Sanchit Gandhi, Patrick von Platen, and Alexander M Rush. 2023. Distil-Whisper: Robust Knowledge Distillation via Large-Scale Pseudo Labelling. *arXiv preprint arXiv:2311.00430*.

Changfeng Gao, Gaofeng Cheng, Runyan Yang, Han Zhu, Pengyuan Zhang, and Yonghong Yan. 2021. Pre-training transformer decoder for end-to-end asr model with unpaired text data. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6543–6547. IEEE.

Dongji Gao, Matthew Wiesner, Hainan Xu, Leibny Paola Garcia, Daniel Povey, and Sanjeev Khudanpur. 2023. Bypass temporal classification: Weakly supervised automatic speech recognition with imperfect transcripts. *arXiv preprint arXiv:2306.01031*.

Claudiu-Mihai Geacăr. 2010. Reducing pilot/atc communication errors using voice recognition. In *Proceedings of ICAS*.

Mohammadreza Ghodsi, Xiaofeng Liu, James Apfel, Rodrigo Cabrera, and Eugene Weinstein. 2020. Rnn-transducer with stateless prediction network. In *ICASSP*, pages 7049–7053. IEEE.

John Godfrey. 1994. The Air Traffic Control Corpus (ATC0) - LDC94S14A.

Zhuo Gong, Daisuke Saito, Sheng Li, Hisashi Kawai, and Nobuaki Minematsu. 2022. Can we train a language model inside an end-to-end asr model?-investigating effective implicit language modeling. In *Proceedings of the Second Workshop on When Creative AI Meets Conversational AI*, pages 42–47.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of *ACM International Conference Proceeding Series*, pages 369–376. ACM.

Alex Graves and Navdeep Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In *International conference on machine learning*, pages 1764–1772.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee.

Yachao Guo, Zhibin Qiu, Hao Huang, and Chng Eng Siong. 2023. Improved Keyword Recognition Based on Aho-Corasick Automaton. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE.

Keith B Hall, Eunjoon Cho, Cyril Allauzen, Francoise Beaufays, Noah Coccaro, Kaisuke Nakajima, Michael Riley, Brian Roark, David Rybach, and Linda Zhang. 2015. Composition-based on-the-fly rescoring for salient n-gram biasing. In *Interspeech*, pages 1418–1422.

Hartmut Helmke, Matthias Kleinert, Arthur Linß, Lucas Klamert, Petr Motlicek, Julia Harfmann, Nuno Cebola, Hanno Wiese, Hörður Arilíusson, and Teodor Simiganoschi. 2023. The haawaii framework for automatic speech understanding of air traffic communication. *13th SESAR Innovation Days 2023, SIDS 2023*.

Hartmut Helmke, Matthias Kleinert, Oliver Ohneiser, Heiko Ehr, and Shruthi Shetty. 2020. Machine learning of air traffic controller command extraction models for speech recognition applications. In *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, pages 1–9. IEEE.

Yosuke Higuchi, Niko Moritz, Jonathan Le Roux, and Takaaki Hori. 2021. Momentum Pseudo-Labeling for Semi-Supervised Speech Recognition. In *Proc. Interspeech 2021*, pages 726–730.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Konrad Hofbauer, Stefan Petrik, and Horst Hering. 2008. The atcosim corpus of non-prompted clean air traffic control speech. In *LREC*.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Takaaki Hori, Ramon Astudillo, Tomoki Hayashi, Yu Zhang, Shinji Watanabe, and Jonathan Le Roux. 2019. Cycle-consistency training for end-to-end speech recognition. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6271–6275. IEEE.

Takaaki Hori, Chiori Hori, Yasuhiro Minami, and Atsushi Nakamura. 2007. Efficient wfst-based one-pass decoding with on-the-fly hypothesis rescoring in extremely large vocabulary continuous speech recognition. *IEEE Transactions on audio, speech, and language processing*, 15(4):1352–1365.

Takaaki Hori and Atsushi Nakamura. 2005. Generalized fast on-the-fly composition algorithm for wfst-based speech recognition. In *INTERSPEECH*, pages 557–560.

Takaaki Hori, Shinji Watanabe, and John R Hershey. 2017. Multi-level language modeling and decoding for open vocabulary end-to-end speech recognition. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 287–293. IEEE.

Takaaki Hori, Daniel Willett, and Yasuhiro Minami. 2003. Language model adaptation using wfst-based speaking-style translation. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, volume 1, pages I–I. IEEE.

Wei-Ning Hsu, Yao-Hung Hubert Tsai, Benjamin Bolte, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. Hubert: How much can a bad teacher benefit asr pre-training? In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6533–6537. IEEE.

Rongqing Huang, Ossama Abdel-hamid, Xinwei Li, and Gunnar Evermann. 2020. Class lm and word mapping for contextual biasing in end-to-end asr. *arXiv preprint arXiv:2007.05609*.

Dongseong Hwang, Khe Chai Sim, Zhouyuan Huo, and Trevor Strohman. 2022. Pseudo label is better than human label. *arXiv preprint arXiv:2203.12668*.

David Imseng, John Dines, Petr Motlicek, Philip N Garner, and Hervé Bourlard. 2012. Comparing different acoustic modeling techniques for multilingual boosting. In *Thirteenth Annual Conference of the International Speech Communication Association*.

Alexei V Ivanov, Patrick L Lange, and David Suendermann-Oeft. 2016. Lvcsr system on a hybrid gpu-cpu embedded platform for real-time dialog applications. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 220–223.

Mahaveer Jain, Gil Keren, Jay Mahadeokar, Geoffrey Zweig, Florian Metze, and Yatharth Saraf. 2020. Contextual rnn-t for open domain asr. *arXiv preprint arXiv:2006.03411*.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Anjuli Kannan, Yonghui Wu, Patrick Nguyen, Tara N Sainath, Zhijeng Chen, and Rohit Prabhavalkar. 2018. An analysis of incorporating an external language model into a sequence-to-sequence model. In *2018 IEEE International*

Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 1–5828. IEEE.

Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Dragomir Radev, Yejin Choi, and Noah A Smith. 2022. A call for clarity in beam search: How it works and when it stops. *arXiv preprint arXiv:2204.05424*.

Banriskhem Khonglah, Srikanth Madikeri, Subhadeep Dey, Hervé Bourlard, Petr Motlicek, and Jayadev Billa. 2020. Incremental semi-supervised learning for multi-genre speech recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7419–7423. IEEE.

Jungsuk Kim and Ian Lane. 2014. Accelerating large vocabulary continuous speech recognition on heterogeneous cpu-gpu platforms. In *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3291–3295. IEEE.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Dietrich Klakow et al. 1998. Log-linear interpolation of language models. In *ICSLP*. Citeseer.

Matthias Kleinert, Hartmut Helmke, Gerald Siol, Heiko Ehr, Aneta Cerna, Christian Kern, Dietrich Klakow, Petr Motlicek, Youssef Oualil, Mittul Singh, et al. 2018. Semi-supervised adaptation of assistant based speech recognition models for different approach areas. In *37th Digital Avionics Systems Conference (DASC)*, pages 1–10. IEEE.

Matthias Kleinert, Oliver Ohneiser, Hartmut Helmke, Shruthi Shetty, Heiko Ehr, Mathias Maier, Susanne Schacht, and Hanno Wiese. 2023. Safety aspects of supporting apron controllers with automatic speech recognition and understanding integrated into an advanced surface movement guidance and control system. *Aerospace*, 10(7):596.

Donald E Knuth, James H Morris, Jr, and Vaughan R Pratt. 1977. Fast pattern matching in strings. *SIAM journal on computing*, 6(2):323–350.

M. Kocour, K. Veselỳ, A. Blatt, J. Zuluaga-Gomez, I. Szöke, J. Cernocky, D. Klakow, and P. Motlicek. 2021. Boosting of Contextual Information in ASR for Air-Traffic Call-Sign Recognition. In *Proc. Interspeech*, pages 3301–3305.

Fangjun Kuang, Liyong Guo, Wei Kang, Long Lin, Mingshuang Luo, Zengwei Yao, and Daniel Povey. 2022. Pruned rnn-t for fast, memory-efficient asr training. In *Interspeech*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Shashi Kumar, Srikanth Madikeri, Iuliia Nigmatulina, Esaú Villatoro-Tello, Petr Motlicek, Karthik Pandia, S Pavankumar Dubagunta, and Aravind Ganapathiraju. 2024a. Multitask speech recognition and speaker change detection for unknown number of speakers. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 12592–12596. IEEE.

Shashi Kumar, Srikanth Madikeri, Juan Zuluaga-Gomez, Iuliia Thorbecke, Esaú Villatoro-Tello, Sergio Burdisso, Petr Motlicek, Karthik Pandia, and Aravind Ganapathiraju. 2024b. Tokenverse: Towards unifying speech and nlp tasks via transducer-based asr. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, page 20988–20995.

Shashi Kumar, Srikanth Madikeri, Juan Zuluaga-Gomez, Esaú Villatoro-Tello, Iuliia Thorbecke, Petr Motlicek, Manjunath KE, and Aravind Ganapathiraju. 2025a. Xlsr-transducer: Streaming asr for self-supervised pretrained models. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Shashi Kumar, Iuliia Thorbecke, Sergio Burdisso, Esaú Villatoro-Tello, Manjunath KE, Kadri Hacioğlu, Pradeep Rangappa, Petr Motlicek, Aravind Ganapathiraju, and Andreas Stolcke. 2025b. Performance evaluation of slam-asr: The good, the bad, the ugly, and the way forward. In *2025 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*, pages 1–5. IEEE.

Gakuto Kurata and George Saon. 2020. Knowledge distillation from offline to streaming rnn transducer for end-to-end speech recognition. In *Interspeech*, pages 2117–2121.

Duc Le, Mahaveer Jain, Gil Keren, Suyoun Kim, Yangyang Shi, Jay Mahadeokar, Julian Chan, Yuan Shangguan, Christian Fuegen, Ozlem Kalinli, et al. 2021a. Contextualized streaming end-to-end speech recognition with trie-based deep biasing and shallow fusion. *arXiv preprint arXiv:2104.02194*.

Duc Le, Gil Keren, Julian Chan, Jay Mahadeokar, Christian Fuegen, and Michael L Seltzer. 2021b. Deep shallow fusion for rnn-t personalization. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 251–257. IEEE.

Bo Li, Shuo-yiin Chang, Tara N Sainath, Ruoming Pang, Yanzhang He, Trevor Strohman, and Yonghui Wu. 2020a. Towards fast and accurate streaming end-to-end asr. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6069–6073. IEEE.

Bo Li, Anmol Gulati, Jiahui Yu, Tara N Sainath, Chung-Cheng Chiu, Arun Narayanan, Shuo-Yiin Chang, Ruoming Pang, Yanzhang He, James Qin, et al. 2021a. A better and faster end-to-end model for streaming asr. In *ICASSP*, pages 5634–5638. IEEE.

Jinyu Li, Rui Zhao, Zhong Meng, Yanqing Liu, Wenning Wei, Sarangarajan Parthasarathy, Vadim Mazalov, Zhenghao Wang, Lei He, Sheng Zhao, et al. 2020b. Developing rnn-t models surpassing high-performance hybrid models with customization capability. In *Interspeech*, pages 3590–3594.

Ke Li, Daniel Povey, and Sanjeev Khudanpur. 2021b. A parallelizable lattice rescoring strategy with neural language models. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6518–6522. IEEE.

Tatiana Likhomanenko, Ronan Collobert, Navdeep Jaitly, and Samy Bengio. 2022. Continuous soft pseudo-labeling in asr. In *I Can't Believe It's Not Better Workshop: Understanding Deep Learning Through Empirical Falsification*.

Tatiana Likhomanenko, Qiantong Xu, Jacob Kahn, Gabriel Synnaeve, and Ronan Collobert. 2020. slimipl: Language-model-free iterative pseudo-labeling. *arXiv preprint arXiv:2010.11524*.

Loren Lugosch. 2020. Sequence-to-sequence learning with transducers.

Loren Lugosch, Tatiana Likhomanenko, Gabriel Synnaeve, and Ronan Collobert. 2022. Pseudo-labeling for massively multilingual speech recognition. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7687–7691. IEEE.

Erik McDermott, Hasim Sak, and Ehsan Variani. 2019. A density ratio approach to language model fusion in end-to-end automatic speech recognition. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 434–441. IEEE.

Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. 2015. librosa: Audio and music signal analysis in python. *SciPy*, 2015:18–24.

Ian McGraw, Rohit Prabhavalkar, Raziel Alvarez, Montse Gonzalez Arenas, Kanishka Rao, David Rybach, Ouais Alsharif, Haşim Sak, Alexander Gruenstein, Françoise Beaufays, et al. 2016. Personalized speech recognition on mobile devices. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5955–5959. IEEE.

Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Timothy P McNamara. 2005. *Semantic priming: Perspectives from memory and word recognition*. Psychology Press.

Zhong Meng, Yashesh Gaur, Naoyuki Kanda, Jinyu Li, Xie Chen, Yu Wu, and Yifan Gong. 2021a. Internal language model adaptation with text-only data for end-to-end speech recognition. *arXiv preprint arXiv:2110.05354*.

Zhong Meng, Naoyuki Kanda, Yashesh Gaur, Sarangarajan Parthasarathy, Eric Sun, Liang Lu, Xie Chen, Jinyu Li, and Yifan Gong. 2021b. Internal language model training for domain-adaptive end-to-end speech recognition. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7338–7342. IEEE.

Zhong Meng, Sarangarajan Parthasarathy, Eric Sun, Yashesh Gaur, Naoyuki Kanda, Liang Lu, Xie Chen, Rui Zhao, Jinyu Li, and Yifan Gong. 2021c. Internal language model estimation for domain-adaptive end-to-end speech recognition. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 243–250. IEEE.

Bertrand Meyer. 1985. Incremental string matching. *Information Processing Letters*, 21(5):219–227.

David E Meyer and Roger W Schvaneveldt. 1971. Facilitation in recognizing pairs of words: evidence of a dependence between retrieval operations. *Journal of experimental psychology*, 90(2):227.

Ashish Mittal, Sunita Sarawagi, and Preethi Jyothi. 2022. In-situ text-only adaptation of speech models with low-overhead speech imputations. In *The Eleventh International Conference on Learning Representations*.

Abdel-rahman Mohamed, George E Dahl, and Geoffrey Hinton. 2011. Acoustic modeling using deep belief networks. *IEEE transactions on audio, speech, and language processing*, 20(1):14–22.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational linguistics*, 23(2):269–311.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88.

Mehryar Mohri, Michael Riley, Donald Hindle, Andrej Ljolje, and Fernando Pereira. 1998. Full expansion of context-dependent networks in large vocabulary speech recognition. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, volume 2, pages 665–668. IEEE.

Petr Motlicek, Amrutha Prasad, Iuliia Nigmatulina, Hartmut Helmke, Oliver Ohneiser, and Matthias Kleinert. 2023. Automatic speech analysis framework for atc communication in haawaii. *13th SESAR Innovation Days 2023, SIDS 2023*.

Hermann Ney, Reinhold Haeb-Umbach, B-H Tran, and Martin Oerder. 1992. Improvements in beam search for 10000-word continuous speech recognition. In *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 9–12. IEEE.

Hermann Ney and Stefan Ortmanns. 2002a. Dynamic programming search for continuous speech recognition. *IEEE Signal Processing Magazine*, 16(5):64–83.

Hermann Ney and Stefan Ortmanns. 2002b. Progress in dynamic programming search for lvcsr. *Proceedings of the IEEE*, 88(8):1224–1240.

Iuliia Nigmatulina, Rudolf Braun, Juan Zuluaga-Gomez, and Petr Motlicek. 2021. Improving callsign recognition with air-surveillance data in air-traffic communication. Idiap-RR Idiap-RR-20-2021, Idiap.

Iuliia Nigmatulina, Srikanth Madikeri, Esaú Villatoro-Tello, Petr Motlíček, Juan Zuluaga-Gomez, Karthik Pandia, and Aravind Ganapathiraju. 2023. Implementing contextual biasing in gpu decoder for online asr. In *Interspeech 2023*, pages 4494–4498.

Iuliia Nigmatulina, Juan Zuluaga-Gomez, Amrutha Prasad, Seyyed Saeed Sarfjoo, and Petr Motlicek. 2022. A two-step approach to leverage contextual data: Speech recognition in air-traffic communications. In *ICASSP 2022-2022 IEEE*

international conference on acoustics, speech and signal processing (ICASSP), pages 6282–6286. IEEE.

Vahid Noroozi, Somshubra Majumdar, Ankur Kumar, Jagadeesh Balam, and Boris Ginsburg. 2024. Stateful conformer with cache-based inference for streaming automatic speech recognition. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 12041–12045. IEEE.

Josef R Novak, Nobuaki Minematsu, and Keikichi Hirose. 2012. Dynamic grammars with lookahead composition for wfst-based speech recognition. In *Thirteenth Annual Conference of the International Speech Communication Association*.

Josef R. Novak, Nobuaki Minematsu, and Keikichi Hirose. 2016. Phonetisaurus: Exploring grapheme-to-phoneme conversion with joint n-gram models in the WFST framework. *Nat. Lang. Eng.*, 22(6):907–938.

Stefan Ortmanns and Hermann Ney. 2000. Look-ahead techniques for fast beam search. *Computer Speech & Language*, 14(1):15–32.

Stefan Ortmanns, Hermann Ney, and Xavier Aubert. 1997. A word graph algorithm for large vocabulary continuous speech recognition. *Computer Speech & Language*, 11(1):43–72.

Youssef Oualil, Dietrich Klakow, Gyorgy Szaszák, Ajay Srinivasamurthy, Hartmut Helmke, and Petr Motlicek. 2017. A context-aware speech recognition and understanding system for air traffic control domain. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 404–408. IEEE.

Youssef Oualil, Marc Schulder, Hartmut Helmke, Anna Schmidt, and Dietrich Klakow. 2015. Real-time integration of dynamic context information for improving automatic speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: an ASR corpus based on public domain audio books. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE.

Sankaran Panchapagesan, Daniel S Park, Chung-Cheng Chiu, Yuan Shangguan, Qiao Liang, and Alexander Gruenstein. 2021. Efficient knowledge distillation for

rnn-transducer models. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5639–5643. IEEE.

Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. 2019. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. In *Proc. Interspeech 2019*, pages 2613–2617.

Daniel S Park, Yu Zhang, Ye Jia, Wei Han, Chung-Cheng Chiu, Bo Li, Yonghui Wu, and Quoc V Le. 2020. Improved noisy student training for automatic speech recognition. *arXiv preprint arXiv:2005.09629*.

Barbara BH Partee, Alice G ter Meulen, and Robert Wall. 2012. *Mathematical methods in linguistics*, volume 30. Springer Science & Business Media.

Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. A time delay neural network architecture for efficient modeling of long temporal contexts. In *Interspeech*, volume 2015, pages 3214–3218.

Stephane Pigeon, Wade Shen, Aaron Lawson, and David A van Leeuwen. 2007. Design and characterization of the non-native military air traffic communications database (nnmatc). In *Eighth Annual Conference of the International Speech Communication Association*.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, CONF. IEEE Signal Processing Society.

Daniel Povey, Mirko Hannemann, Gilles Boulianne, Lukáš Burget, Arnab Ghoshal, Miloš Janda, Martin Karafiát, Stefan Kombrink, Petr Motlíček, Yanmin Qian, et al. 2012. Generating exact lattices in the wfst framework. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4213–4216. IEEE.

Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur. 2016. Purely sequence-trained neural networks for asr based on lattice-free mmi. In *Interspeech*, pages 2751–2755.

Rohit Prabhavalkar, Takaaki Hori, Tara N Sainath, Ralf Schlüter, and Shinji Watanabe. 2023. End-to-end speech recognition: A survey. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

Golan Pundak, Tara N Sainath, Rohit Prabhavalkar, Anjuli Kannan, and Ding Zhao. 2018. Deep context: end-to-end contextual speech recognition. In *2018 IEEE spoken language technology workshop (SLT)*, pages 418–425. IEEE.

Janne Pylkkönen, Antti Ukkonen, Juho Kilpikoski, Samu Tamminen, and Hannes Heikinheimo. 2021. Fast text-only domain adaptation of rnn-transducer prediction network. *arXiv preprint arXiv:2104.11127*.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pages 28492–28518. PMLR.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Mosur K Ravishankar. 1996. Efficient algorithms for speech recognition. Technical report, Carnegie Mellon University, Pittsburgh.

Dima Rekesh, Nithin Rao Koluguri, Samuel Kriman, Somshubra Majumdar, Vahid Noroozi, He Huang, Oleksii Hrinchuk, Krishna Puvvada, Ankur Kumar, Jagadeesh Balam, et al. 2023. Fast conformer with linearly scalable attention for efficient speech recognition. In *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 1–8. IEEE.

MD Rio, Peter Ha, Quinten McNamara, Corey Miller, and Shipra Chandra. 2022. Earnings-22: A practical benchmark for accents in the wild.

Miguel Del Rio, Natalie Delworth, Ryan Westerman, Michelle Huang, Nishchal Bhandari, Joseph Palakapilly, Quinten McNamara, Joshua Dong, Piotr Zelasko, and Miguel Jette. 2021. Earnings-21: A practical benchmark for asr in the wild. In *Interspeech*.

David Rybach, Hermann Ney, and Ralf Schlüter. 2013. Lexical prefix tree and wfst: A comparison of two dynamic search concepts for lvcsr. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(6):1295–1307.

David Rybach, Ralf Schlüter, and Hermann Ney. 2011. A comparative analysis of dynamic network decoding. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5184–5187. IEEE.

Tara N Sainath, Yanzhang He, Bo Li, Arun Narayanan, Ruoming Pang, Antoine Bruguier, Shuo-yiin Chang, Wei Li, Raziel Alvarez, Zhifeng Chen, et al. 2020. A streaming on-device end-to-end model surpassing server-side conventional model quality and latency. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6059–6063. IEEE.

Anna Schmidt, Youssef Oualil, Oliver Ohneiser, Matthias Kleinert, Marc Schulder, Arif Khan, Hartmut Helmke, and Dietrich Klakow. 2014. Context-based recognition network adaptation for improving on-line asr in air traffic control. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 13–18. IEEE.

JC Segura, T Ehrette, A Potamianos, D Fohr, I Illina, PA Breton, V Clot, R Gemello, M Matassoni, and P Maragos. 2007. The hiwire database, a noisy and non-native english speech corpus for cockpit communication. *Online. http://www. hiwire. org*.

Jack Serrino, Leonid Velikovich, Petar S Aleksic, and Cyril Allauzen. 2019. Contextual recovery of out-of-lattice named entities in automatic speech recognition. In *Interspeech*, pages 3830–3834.

Jennifer R Shelton and Randi C Martin. 1992. How semantic is automatic semantic priming? *Journal of Experimental Psychology: Learning, memory, and cognition*, 18(6):1191.

Todd Shore, Friedrich Faubel, Hartmut Helmke, and Dietrich Klakow. 2012. Knowledge-based word lattice rescoring in a dynamic context. In *Thirteenth Annual Conference of the International Speech Communication Association*.

Yujing Si, Qingqing Zhang, Ta Li, Jielin Pan, and Yonghong Yan. 2013. Prefix tree based n-best list re-scoring for recurrent neural network language model used in speech recognition system. In *Interspeech*, pages 3419–3423.

K. C. Sim, F. Beaufays, A. Benard, D. Guliani, A. Kabel, N. Khare, T. Lucassen, P. Zadrazil, H. Zhang, L. Johnson, G. Motta, and L. Zhou. 2019. Personalization of End-to-End Speech Recognition on Mobile Devices for Named Entities. In *Proc. Autom. Speech Recog. and Understanding Workshop*, pages 23–30.

Luboš Šmídl, Jan Švec, Daniel Tihelka, Jindřich Matoušek, Jan Romportl, and Pavel Ircing. 2019. Air traffic control communication (atcc) speech corpora and

their use for asr and tts development. *Language Resources and Evaluation*, 53:449–464.

Ajay Srinivasamurthy, Petr Motlicek, Ivan Himawan, Gyorgy Szaszak, Youssef Oualil, and Hartmut Helmke. 2017. Semi-supervised learning with semantic knowledge extraction for improved speech recognition in air traffic control. In *Proc. of the 18th Annual Conference of the International Speech Communication Association*.

Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. 2017. Cold fusion: Training seq2seq models together with language models. *arXiv preprint arXiv:1708.06426*.

Andreas Stolcke. 2002. SRILM-an extensible language modeling toolkit. In *Seventh international conference on spoken language processing*.

Anej Svete, Benjamin Dayan, Ryan Cotterell, Tim Vieira, and Jason Eisner. 2022. Algorithms for acyclic weighted finite-state automata with failure arcs. In *EMNLP*, pages 8289–8305, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Pawel Swietojanski, Stefan Braun, et al. 2023. Variable attention masking for configurable transformer transducer speech recognition. In *ICASSP*, pages 1–5. IEEE.

Ryoichi Takashima, Sheng Li, and Hisashi Kawai. 2018. An investigation of a knowledge distillation method for ctc acoustic models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5809–5813. IEEE.

Simone Tedeschi, Valentino Maiorca, Niccolò Campolungo, Francesco Cecconi, and Roberto Navigli. 2021. WikiNEuRal: Combined neural and knowledge-based silver data creation for multilingual NER. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2521–2533, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Samuel Thomas, Brian Kingsbury, George Saon, and Hong-Kwang J Kuo. 2022. Integrating text inputs for training and adapting rnn transducer asr models. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8127–8131. IEEE.

Iuliia Thorbecke, Esaú Villatoro-Tello, Juan Zuluaga-Gomez, Shashi Kumar, Sergio Burdisso, Pradeep Rangappa, Andrés Carofilis, Srikanth Madikeri, Petr

Motlicek, Karthik Pandia, Kadri Hacioğlu, and Andreas Stolcke. 2025. Unifying global and near-context biasing in a single trie pass. In *The 28th International Conference of Text, Speech and Dialogue (TSD2025)*. Springer.

Iuliia Thorbecke, Juan Zuluaga-Gomez, Esaú Villatoro-Tello, Shashi Kumar, Pradeep Rangappa, Sergio Burdisso, Petr Motlicek, Karthik Pandia, and Aravind Ganapathiraju. 2024. Fast streaming transducer asr prototyping via knowledge distillation with whisper. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16747—-16762.

Jinchuan Tian, Jianwei Yu, Chao Weng, Yuexian Zou, and Dong Yu. 2022. Improving mandarin end-to-end speech recognition with word n-gram language model. *IEEE Signal Processing Letters*, 29:812–816.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ehsan Variani, David Rybach, Cyril Allauzen, and Michael Riley. 2020. Hybrid autoregressive transducer (hat). In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6139–6143. IEEE.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Leonid Velikovich, Ian Williams, Justin Scheiner, Petar S Aleksic, Pedro J Moreno, and Michael Riley. 2018. Semantic lattice processing in contextual automatic speech recognition for google assistant. In *Interspeech*, pages 2222–2226.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.

A. Vyas, S. Madikeri, and H. Bourlard. 2021. Lattice-Free Mmi Adaptation of Self-Supervised Pretrained Acoustic Models. In *Proc. Int. Conf. on Acoust., Speech and Signal Process.*, pages 6219–6223.

Dong Wang, Xiaodong Wang, and Shaohe Lv. 2019. An overview of end-to-end automatic speech recognition. *Symmetry*, 11(8):1018.

Jiannan Wang, Jianhua Feng, and Guoliang Li. 2010. Trie-join: Efficient trie-based string similarity joins with edit-distance constraints. *Proceedings of the VLDB Endowment*, 3(1-2):1219–1230.

Peidong Wang, Tara N Sainath, and Ron J Weiss. 2020. Multitask training with text data for end-to-end speech recognition. *arXiv preprint arXiv:2010.14318*.

Weiran Wang, Zelin Wu, Diamantino Caseiro, Tsendsuren Munkhdalai, Khe Chai Sim, Pat Rondon, Golan Pundak, Gan Song, Rohit Prabhavalkar, Zhong Meng, et al. 2023. Contextual biasing with the knuth-morris-pratt matching algorithm. *arXiv preprint arXiv:2310.00178*.

Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R Hershey, and Tomoki Hayashi. 2017a. Hybrid ctc/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1240–1253.

Shinji Watanabe, Takaaki Hori, Jonathan Le Roux, and John R Hershey. 2017b. Student-teacher network learning with enhanced features. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5275–5279. IEEE.

Ian Williams, Anjuli Kannan, Petar S Aleksic, David Rybach, and Tara N Sainath. 2018. Contextual speech recognition in end-to-end neural network systems using beam search. In *Interspeech*, pages 2227–2231.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, and Morgan Funtowicz et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45. Association for Computational Linguistics.

Wayne Xiong, Lingfeng Wu, Fil Alleva, Jasha Droppo, Xuedong Huang, and Andreas Stolcke. 2018. The microsoft 2017 conversational speech recognition system. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5934–5938. IEEE.

Guanrou Yang, Ziyang Ma, Zhifu Gao, Shiliang Zhang, and Xie Chen. 2024a. Ctc-assisted llm-based contextual asr. In *2024 IEEE Spoken Language Technology Workshop (SLT)*, pages 126–131. IEEE.

Haichuan Yang, Yuan Shangguan, Dilin Wang, Meng Li, Pierce Chuang, Xiaohui Zhang, Ganesh Venkatesh, Ozlem Kalinli, and Vikas Chandra. 2022a. Omni-sparsity dnn: Fast sparsity optimization for on-device streaming e2e asr via supernet. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8197–8201. IEEE.

Xiaoyu Yang, Wei Kang, Zengwei Yao, Yifan Yang, Liyong Guo, Fangjun Kuang, Long Lin, and Daniel Povey. 2024b. Promptasr for contextualized asr with controllable style. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 10536–10540. IEEE.

Xiaoyu Yang, Qiujia Li, and Philip C Woodland. 2022b. Knowledge distillation for neural transducers from large self-supervised pre-trained models. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8527–8531. IEEE.

Zijian Yang, Minh-Nghia Phan, Ralf Schlüter, and Hermann Ney. 2025. Label-context-dependent internal language model estimation for ctc. *arXiv preprint arXiv:2506.06096*.

Zengwei Yao, Liyong Guo, Xiaoyu Yang, Wei Kang, Fangjun Kuang, Yifan Yang, Zengrui Jin, Long Lin, and Daniel Povey. 2024. Zipformer: A faster and better encoder for automatic speech recognition. In *ICLR*.

Ching-Feng Yeh, Jay Mahadeokar, Kaustubh Kalgaonkar, Yongqiang Wang, Duc Le, Mahaveer Jain, Kjell Schubert, Christian Fuegen, and Michael L Seltzer. 2019. Transformer-transducer: End-to-end speech recognition with self-attention. *arXiv preprint arXiv:1910.12977*.

Steve J Young and Sj Young. 1993. The htk hidden markov model toolkit: Design and philosophy.

George Zavaliagkos and Thomas Colthurst. 1998. Utilizing untranscribed training data to improve perfomance. In *LREC*, pages 317–322. Citeseer.

Mohammad Zeineldeen, Aleksandr Glushko, Wilfried Michel, Albert Zeyer, Ralf Schlüter, and Hermann Ney. 2021. Investigating methods to improve language model integration for attention-based encoder-decoder asr models. In *Interspeech*, pages 2856–2860.

Piotr Żelasko, Daniel Povey, Jan Trmal, Sanjeev Khudanpur, et al. 2021. Lhotse: a speech data representation library for the modern deep learning ecosystem. *arXiv preprint arXiv:2110.12561*.

Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik McDermott, Stephen Koo, and Shankar Kumar. 2020a. Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7829–7833. IEEE.

Yu Zhang, James Qin, Daniel S Park, Wei Han, Chung-Cheng Chiu, Ruoming Pang, Quoc V Le, and Yonghui Wu. 2020b. Pushing the limits of semi-supervised learning for automatic speech recognition. *arXiv preprint arXiv:2010.10504*.

Ding Zhao, Tara N Sainath, David Rybach, Pat Rondon, Deepti Bhatia, Bo Li, and Ruoming Pang. 2019. Shallow-fusion end-to-end contextual biasing. In *Interspeech*, pages 1418–1422.

Rui Zhao, Jian Xue, Jinyu Li, Wenning Wei, Lei He, and Yifan Gong. 2021. On addressing practical challenges for rnn-transducer. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 526–533. IEEE.

Rui Zhao, Jian Xue, Partha Parthasarathy, Veljko Miljanic, and Jinyu Li. 2023. Fast and accurate factorized neural transducer for text adaption of end-to-end speech recognition models. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Huahuan Zheng, Keyu An, Zhijian Ou, Chen Huang, Ke Ding, and Guanglu Wan. 2022. An empirical study of language model integration for transducer based speech recognition. In *Interspeech 2022*, pages 3904–3908.

Xianrui Zheng, Yulan Liu, Deniz Gunceler, and Daniel Willett. 2021. Using synthetic audio to improve the recognition of out-of-vocabulary words in end-to-end asr systems. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5674–5678. IEEE.

Wei Zhou, Eugen Beck, Simon Berger, Ralf Schlüter, and Hermann Ney. 2023. Rasr2: The rwth asr toolkit for generic sequence-to-sequence speech recognition. In *Interspeech*, pages 4094–4098.

Wei Zhou, Zuoyun Zheng, Ralf Schlüter, and Hermann Ney. 2022. On language model integration for rnn transducer based speech recognition. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8407–8411. IEEE.

Han Zhu, Dongji Gao, Gaofeng Cheng, Daniel Povey, Pengyuan Zhang, and Yonghong Yan. 2023. Alternative pseudo-labeling for semi-supervised automatic speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

J. Zuluaga-Gomez, K. Veselỳ, A. Blatt, P. Motlicek, D. Klakow, A. Tart, I. Szöke, A. Prasad, S. Sarfjoo, P. Kolčárek, et al. 2020. Automatic call sign detection: Matching air surveillance data with air traffic spoken communications. In *Proc. of 8th OpenSky Symp.*, volume 59, page 14. Multidisciplinary Digital Publishing Institute Proceedings.

Juan Zuluaga-Gomez, Petr Motlícek, Qingran Zhan, Karel Veselý, and Rudolf Braun. 2020. Automatic speech recognition benchmark for air-traffic communications. In *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, pages 2297–2301. ISCA.

Juan Zuluaga-Gomez, Iuliia Nigmatulina, Amrutha Prasad, Petr Motlicek, Driss Khalil, Srikanth Madikeri, Allan Tart, Igor Szoke, Vincent Lenders, Mickael Rigault, et al. 2023a. Lessons learned in transcribing 5000 h of air traffic control communications for robust automatic speech understanding. *Aerospace*, 10(10):898.

Juan Zuluaga-Gomez, Iuliia Nigmatulina, Amrutha Prasad, Petr Motlicek, Karel Veselỳ, Martin Kocour, and Igor Szöke. 2021. Contextual semi-supervised learning: An approach to leverage air-surveillance and untranscribed atc data in asr systems. In *Interspeech 2021*, pages 3296–3300.

Juan Zuluaga-Gomez, Amrutha Prasad, Iuliia Nigmatulina, Petr Motlicek, and Matthias Kleinert. 2023b. A virtual simulation-pilot agent for training of air traffic controllers. *Aerospace*, 10(5):490.

Juan Zuluaga-Gomez, Amrutha Prasad, Iuliia Nigmatulina, Seyyed Saeed Sarfjoo, Petr Motlicek, Matthias Kleinert, Hartmut Helmke, Oliver Ohneiser, and Qingran Zhan. 2023c. How does pre-trained wav2vec 2.0 perform on domain-shifted asr? an extensive benchmark on air traffic control communications. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pages 205–212. IEEE.

Juan Zuluaga-Gomez, Seyyed Saeed Sarfjoo, Amrutha Prasad, Iuliia Nigmatulina, Petr Motlicek, Karel Ondrej, Oliver Ohneiser, and Hartmut Helmke. 2023d. Bertraffic: Bert-based joint speaker role and speaker change detection for air

traffic control communications. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pages 633–640. IEEE.

Juan Zuluaga-Gomez, Karel Veselỳ, Igor Szöke, Alexander Blatt, Petr Motlicek, Martin Kocour, Mickael Rigault, Khalid Choukri, Amrutha Prasad, Seyyed Saeed Sarfjoo, et al. 2022. Atco2 corpus: A large-scale dataset for research on automatic speech recognition and natural language understanding of air traffic control communications. *arXiv preprint arXiv:2211.04054*.

A. Additional Tables

A.1. Streaming decoding configurations

We perform a swipe of streaming decoding evaluations under multiple low-latency settings. We evaluate the following configurations:

- Decode chunk size = 320ms with left context of 2560ms, 5120ms and full;
- decode chunk size = 640ms with left context of 2560ms, 5120ms and full;
- decode chunk size = 1280ms with left context of 2560ms, 5120ms and full;
- decode chunk size = 2560ms with left context of 2560ms, 5120ms and full.

The overall results are reported in Figure 6.7 for each of the proposed languages.

A.2. Extended results for models trained on PL data

Offline models evaluation Table A.1 shows the WERs for Zipformer offline models trained on six CommonVoice languages with either solely PLs or a mix of PLs and a small amount of supervised data (100h). These extended results correspond to those depicted in Figure 6.7 in the main paper.

Streaming models evaluation Table A.2 shows the WERs for Zipformer streaming models trained on four CommonVoice languages with solely PLs and evaluated on two different streaming configurations.

Table A.1.: WERs for six CommonVoice languages. The Zipformer offline models are trained with pseudo-labelled data from different Whisper models. We also report WERs when a small amount of supervised data is added during training, denoted as "`sup. [100h]`". Note that the transducer models are trained from scratch in ∼1 day GPU time.

Experiment	**Language [hours]**					
	CA	EN	DE	FR	ES	IT
	1200	1000	600	600	317	200
Whisper-tiny (39M)						
Whisper	51.0	28.8	34.5	49.7	30.3	44.5
Zipformer (70M)	41.1	21.5	25.7	33.8	20.1	32.2
+sup. [100h]	36.8	20.9	22.6	29.7	16.1	19.8
+n-gram LM	32.4	21.0	22.1	31.3	15.9	18.7
+n-gram LM+(BL)	32.0	20.7	21.5	30.8	15.6	18.0
Whisper-base (74M)						
Whisper	39.9	21.9	24.5	37.3	19.6	30.5
Zipformer (70M)	30.5	19.2	19.4	24.7	14.8	22.7
+sup. [100h]	27.9	19.1	17.5	21.8	12.6	16.3
+n-gram LM	24.0	19.1	17.0	22.2	12.2	15.5
+n-gram LM+(BL)	23.7	18.8	16.4	21.7	11.8	14.8
Whisper-small (244M)						
Whisper	23.8	14.5	13.0	22.7	10.3	16.0
Zipformer (70M)	18.6	17.1	13.4	16.5	10.7	14.8
+sup. [100h]	17.4	16.9	12.8	15.8	10.2	12.9
+n-gram LM	15.1	16.7	12.4	15.8	10.0	12.4
+n-gram LM+(BL)	14.9	16.4	11.9	15.4	9.7	11.8
Whisper-medium (769M)						
Whisper	16.4	11.2	8.5	16.0	6.9	9.4
Zipformer (70M)	14.0	16.7	11.3	13.7	9.5	12.1
+sup. [100h]	13.7	16.4	11.3	13.5	9.5	12.0
+n-gram LM	12.1	16.2	10.9	13.2	9.3	11.5
+n-gram LM+(BL)	11.9	15.9	10.4	12.9	9.0	11.1
Whisper-large-v3 (1.5B)						
Whisper	14.1	9.4	6.4	13.9	5.6	7.1
Zipformer (70M)	12.8	16.2	10.5	12.4	8.9	11.1
+sup. [100h]	13.6	16.3	10.7	12.4	9.0	11.6
+n-gram LM	12.1	16.0	10.4	12.0	8.9	11.3
+n-gram LM+(BL)	11.8	15.6	10.0	11.6	8.6	10.8
Baseline offline Zipformer (only supervised data)						
Zipformer (70M)	4.9	14.5	8.5	10.7	8.1	10.2

Table A.2.: WERs for streaming evaluation with n-gram LM and bias-lists (BL). Listed on four CommonVoice languages and two decoding configurations. The Zipformer models are trained with only pseudo-labelled data from different Whisper models. All experiments show additive WERs improvement when adding either (or both) n-gram LM or biasing lists.

	cs=320ms;lf=2.5s				**cs=320ms;lf=∞**			
Experiment	CA	DE	ES	IT	CA	DE	ES	IT
Whisper-tiny (39M)								
Zipformer (70M)	46.2	29.5	24.5	37.3	46.0	28.9	23.8	36.7
+n-gram LM	46.0	29.0	24.0	36.5	45.8	28.4	23.2	35.9
+bias-list	43.7	28.9	23.7	36.0	43.6	28.4	23.0	35.3
+n-gram LM+(BL)	43.6	28.4	23.3	35.3	43.5	28.0	22.6	34.6
Whisper-base (74M)								
Zipformer (70M)	39.7	23.9	20.2	28.4	39.4	23.3	19.5	27.6
+n-gram LM	39.1	23.2	19.8	27.5	38.7	22.5	19.0	26.7
+bias-list	36.2	23.3	19.6	27.2	36.0	22.7	18.9	26.4
+n-gram LM+(BL)	36.0	22.7	19.3	26.5	35.7	22.2	18.5	25.7
Whisper-small (244M)								
Zipformer (70M)	21.1	22.4	16.2	21.5	20.8	21.3	15.4	20.6
+n-gram LM	20.8	21.8	15.8	20.7	20.5	20.8	15.0	19.8
+bias-list	20.0	21.7	15.6	20.5	19.7	20.6	14.8	19.7
+n-gram LM+(BL)	19.8	21.3	15.2	20.0	19.6	20.2	14.5	19.1
Whisper-medium (769M)								
Zipformer (70M)	16.7	16.5	17.6	18.6	16.4	15.8	16.7	17.8
+n-gram LM	16.4	15.8	17.4	17.8	16.1	15.1	16.4	17.0
+bias-list	16.1	16.0	17.0	17.8	15.9	15.3	16.0	17.1
+n-gram LM+(BL)	15.9	15.6	16.8	17.3	15.7	14.9	15.8	16.5
Whisper-large-v3 (1.5B)								
Zipformer (70M)	22.5	16.1	15.9	17.5	21.8	15.3	15.1	16.7
+n-gram LM	22.4	15.4	15.6	16.8	21.7	14.7	14.9	16.0
+bias-list	21.9	15.6	15.5	16.9	21.1	14.9	14.8	16.1
+n-gram LM+(BL)	22.1	15.2	15.3	16.4	21.3	14.5	14.6	15.6
Baseline streaming Zipformer (only supervised data)								
Zipformer (70M)	7.8	13.8	13.5	17.5	7.6	13.1	12.8	16.6

A.3. Domain adaptation with pseudo-audio features

Table A.3.: Domain adaptation evaluated with speech and text modalities on the GigaSpeech test set and the DefinedAI test set. Evaluation is done in two separated ways: (1) when the input is speech and features are extracted from audio (**Giga-S** and **DefAI-S**), (2) when the input is text and pseudo-audio features are extracted text (**Giga-T** and **DefAI-T**). Adaptation data: 48h (health, insurance, banking) + 299h (insurance) + 173h (banking). Predictor (P), joiner (J), decoder linear projector (LP-LM); LinearProjector for feature projection.

Exp	**Fine-tuning on DefinedAI:**			**WER**			
	audio features	**layers**	**data,h**	**Giga-S**	**Giga-T**	**DefAI-S**	**DefAI-T**
(0)	-	-	-	18.6	17.7	25.0	15.0
(1)	Audio (Fbank)	all layers	48	24.0	22.3	*11.7*	*13.4*
(2)	Text (SpT5-Yf)	all layers	48	27.1	13.4	28.7	8.1
	With LinearProjection:						
(3)	Text (SpT5-Yf)	P, J, LP-LM	48	26.3	20.5	22.6	11.1