KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

From Speech to Semantics

Adapting Pretrained Transformers for Low Resource Spoken Language Understanding

Quentin Meeus

Supervisors:
Hugo Van hamme (ESAT-PSI)
Marie-Francine Moens (CW-HCI)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

June 2024

From Speech to Semantics

Adapting Pretrained Transformers for Low Resource Spoken Language Understanding

Quentin MEEUS

Examination committee:
Omer Van der Biest, chair
Hugo Van hamme, supervisor
Marie-Francine Moens, supervisor
Tinne Tuytelaars
Patrick Wambacq
Véronique Hoste (UGent)
Wenpeng Yin (Temple University)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

June 2024

Preface

This doctoral thesis is about enabling computers to understand spoken language. It goes beyond literal transcription, into the real meaning of the words and phrases. Human languages are extremely rich and complex. They evolve dynamically, reflecting societal changes and bridging cultures. Speaking, not writing, is one of the first thing that a newborn will learn, or rather, emitting sounds for the purpose of communicating. This richness and evolution pose a fascinating challenge for artificial intelligence, but also a crucial one. Spoken language understanding is not merely an academic pursuit; it's the key to unlocking new frontiers in human-computer interaction. Imagine systems that can truly listen, respond intelligently, and even anticipate our needs, bridging the gap between humans and machines. By delving into the nuances of spoken language, this thesis aims to contribute to the ongoing quest to create AI systems that are not just intelligent, but truly conversational.

Spoken language is constantly evolving over time and space, shaped by the societal context. Words come and go, new ones are invented, and old ones change meaning or disappear, forgotten even by our grandparents. And looking at this evolution can truly give us a sense of major societal events. For example, every year, Oxford Languages captures in a public report[1] "how English around the world expressed its own view, sometimes sharing the collective expressions for the phenomena endured globally this year, and at other times using regionally specific words and usages". The first year of my PhD, in 2020, the reports cites 16 words that have been highly influential to describe the year. Among them, "Covid-19", "Lockdown" and "Social Distancing" shaped the first two years of my PhD. In this report, we can also find the word "Impeachment". Indeed, this is the third time in History that an American president got impeached (in the next year, after his second impeachment trial, Donald Trump will become the first U.S. president ever to be impeached twice). We can also cite "Black Lives Matter", a scream coming from the U.S. that echoed all around the world,

[1] https://languages.oup.com/word-of-the-year/2020/

showing that discrimination still poses problems in many countries including Belgium. Finally, “Bushfire”, “Climate” and “Net Zero” describe how our view of Climate Change has evolved since, with an ever-increasing public awareness but also more and more voices speaking out against political inaction.

The development of intelligent systems is not different from any other large-scale industry: the training and usage of deep neural networks has a tremendous cost on the planet. Researchers at HuggingFace calculated that training GPT3 took 1,287 MWh, resulting in 552 tons of emitted CO_2[2], that is, the carbon footprint of flying almost 150 times from Paris to New York and back. As the need for sustainable solutions grows, so does the demand for energy-efficient technologies. This thesis explores how spoken language understanding can be achieved with low computational resources, ensuring that advancements in this field are not only intelligent but also environmentally conscious.

I could not have completed my PhD without the help and support of many people. First and foremost, I would like to express my deepest gratitude to my promoters, Hugo and Sien, for their unwavering support, guidance, and expertise throughout my PhD journey. Their insightful feedback and encouragements were invaluable in shaping this research, and I will be forever grateful to them. I am also thankful to the members of my PhD committee for spending the time to read my thesis and giving me constructive feedback. To my colleagues at LIIR and Spraak, thank you for the discussions. Your intellectual curiosity and insightful discussions enriched my research experience immeasurably. Special thanks to Jakob and Pu, my office buddies in the last four years. Finally, I am deeply grateful for the funding provided by Flanders AI Research, which made this research possible.

Enfin, à mes parents, à mes sœurs, à mes amis: merci de m’avoir accompagné pendant toutes ces années. Merci pour les nombreux verres, restos, cinés, courses de trail, séances d’escalade et de badminton, toutes ces activités qui m’ont permis de prendre de la distance et d’arriver au bout de cette aventure avec toute ma tête. Merci tout specialement à Romane qui, sans même en avoir conscience, m’a donné la motivation pour venir à bout de cette thèse.

[2] A. S. Luccioni, S. Viguier, and A.-L. Ligozat. “Estimating the Carbon Footprint of BLOOM, a 176B Parameter Language Model”. In: (2022). arXiv: 2211.02001 [cs.LG].

Abstract

This thesis investigates Spoken Language Understanding (SLU) within low-resource scenarios, focusing on enhancing efficiency and adaptability. We introduce a novel approach for learning expressive speech representations using Bidirectional Transformers with masked language modeling. This approach allows us to train lightweight downstream SLU models effectively, even with limited labeled data. We also explore multitask learning, combining SLU tasks with speech recognition to further boost performance in low-resource settings. Our experiments with intent recognition and sentiment analysis demonstrate the effectiveness of these methods on Dutch and English datasets.

Building upon these findings, we adapt a pretrained speech-to-text model (Whisper) for more complex SLU tasks like slot filling and Spoken Named Entity Recognition (Spoken NER). We explore techniques such as task-specific decoder modules, parameter freezing, and embedding pruning to maintain low data and computational requirements. Our multilingual Spoken NER model, trained on the MSNER dataset we created, showcases the benefits of shared representations across languages and tackles the issue of catastrophic forgetting through strategies like embedding reset and experience replay.

To address the scarcity of labeled data for multilingual SLU, we investigate the use of text-based NLU models to generate silver-quality annotations. We analyze the MSNER dataset, highlighting the potential of this approach while cautioning against potential biases. Finally, we explore the advantages of multilingual Spoken NER models over monolingual counterparts, demonstrating improved performance and generalization, particularly in low-resource scenarios.

Overall, this thesis contributes to the development of efficient and adaptable SLU systems for low-resource contexts. Our work underscores the potential of bidirectional attention, multitask learning, pretrained model adaptation, and multilingual approaches in advancing the field of SLU. We also highlight the challenges and potential solutions for catastrophic forgetting in multilingual settings.

Beknopte Samenvatting

Dit proefschrift onderzoekt Gesproken Taalbegrip (SLU) in situaties met beperkte gegevens, met een focus op het verbeteren van efficiëntie en aanpassingsvermogen. We introduceren een nieuwe aanpak voor het leren van expressieve spraakrepresentaties met behulp van Bidirectionele Transformers met gemaskeerde taalmodellering. Deze aanpak stelt ons in staat om lichte downstream SLU-modellen effectief te trainen, zelfs met beperkte gelabelde data. We verkennen ook *multitask learning*, waarbij SLU-taken worden gecombineerd met spraakherkenning om de prestaties in situaties met beperkte gegevens verder te verbeteren. Onze experimenten met intentieherkenning en sentimentanalyse tonen de effectiviteit van deze methoden aan op Nederlandse en Engelse datasets.

Voortbouwend op deze bevindingen passen we een vooraf getraind spraak-naar-tekst model (Whisper) aan voor complexere SLU-taken zoals *slot filling* en *Spoken Named Entity Recognition* (Spoken NER). We onderzoeken technieken zoals taakspecifieke decodermodules, parameterbevriezing en *embedding pruning* om lage data- en computationele vereisten te behouden. Ons meertalige Spoken NER-model, getraind op de door ons gemaakte MSNER-dataset, toont de voordelen van gemeenschappelijke representaties tussen talen en pakt het probleem van catastrofaal vergeten aan door middel van strategieën zoals *embedding reset* en *experience replay*.

Om het gebrek aan gelabelde data voor meertalig SLU aan te pakken, onderzoeken we het gebruik van tekstgebaseerde NLU-modellen om zilveren annotaties te genereren. We analyseren de MSNER-dataset en benadrukken het potentieel van deze aanpak, terwijl we waarschuwen voor mogelijke biases. Ten slotte onderzoeken we de voordelen van meertalige Spoken NER-modellen ten opzichte van eentalige tegenhangers, waarbij we verbeterde prestaties en generalisatie aantonen, vooral in situaties met beperkte gegevens.

Over het algemeen draagt dit proefschrift bij aan de ontwikkeling van efficiënte en aanpasbare SLU-systemen voor contexten met beperkte gegevens. Ons werk onderstreept het potentieel van bidirectionele aandacht, *multitask learning*, aanpassing van vooraf getrainde modellen en meertalige benaderingen bij het bevorderen van het SLU vakgebied. We benadrukken ook de uitdagingen en mogelijke oplossingen voor catastrofaal vergeten in meertalige omgevingen.

List of Abbreviations

AI Artificial Intelligence
ASR Automatic Speech Recognition
BERT Bidirectional Representations from Transformers
BIO Beginning, inside, outside
CER Character Error Rate
CGN Corpus Gesproken Nederland
CLM Causal Language Model
CNN Convolutional Neural Network
CRF Conditional Random Field
CTC Connectionist Temporal Classification
ER Experience Replay
FSC Fluent Speech Commands
GPT Generative Pretrained Transformer
IR Intent Recognition
LLM Large Language Model
LM Language Model
MLM Masked Language Model
MSNER Multilingual Spoken Named Entity Recognition
MTL Multitask Learning
NER Named Entity Recognition
NLP Natural Language Processing
NLU Natural Language Understanding
NMF Non-negative Matrix Factorization
RNN Recurrent Neural Network
RoBERTa Robustly Optimized BERT
SC Sentiment Classification
SF Slot Filling
SLU Spoken Language Understanding
t-SNE t-distributed Stochastic Neighbor Embedding
WER Word Error Rate
XLM Cross-lingual Language Model

Contents

List of Figures

List of Tables

Chapter 1

General Introduction

The increasing demand for sophisticated spoken language understanding (SLU) systems remains a compelling research area, ever more so with the recent rise in popularity of large language models (LLM). While LLMs have demonstrated impressive capabilities, they are essentially trained on written language and their effectiveness in understanding spoken language, particularly in real-world, diverse contexts, remains a challenge. Modern users, accustomed to voice-activated virtual assistants, expect not only accuracy but also fairness, inclusivity, and environmental responsibility from these applications.

Commercial systems predominantly rely on cascading automatic speech recognition (ASR) and natural language understanding (NLU) modules. However, this approach suffers from limitations such as error propagation, loss of valuable acoustic information, and computational heaviness. Moreover, these systems often struggle to understand nuances in spoken language, particularly accents and dialects that deviate from mainstream datasets, perpetuating biases against minority groups. Additionally, the environmental cost of training and deploying these computationally intensive models raises concerns about their long-term sustainability. There is a strong demand for lightweight, adaptable personal assistants that understand spoken language nuances without extensive user input.

Recent breakthroughs with Transformer-based architectures have revolutionized natural language processing (NLP) and spurred advancements in speech processing applications. Transformers excel at processing long sequences, capturing relevant information, and generating robust representations. However, applying these models to resource-constrained SLU scenarios, where data and computational power may be limited, demands further investigation.

This thesis investigates end-to-end SLU approaches, with a specific emphasis on extracting information from spoken language within low-resource contexts. Leveraging

the power of Transformer models, the research investigates methods for pretraining speech representations, multitask learning, and extending pretrained models to specific SLU tasks like named entity recognition (NER), intent recognition (IR), and slot filling (SF). By addressing these challenges, we aim to develop efficient SLU models that can generalize well, especially with limited training data, while minimizing the computational and environmental impacts.

1.1 Research Questions

1.1.1 How can Transformer architectures be effectively pretrained to capture robust speech representations suitable for downstream SLU tasks?

The premise for this research question lies on the realization that speech input is very noisy and raw. Semantic information pertaining to the meaning of the words and phrases is deeply entangled and requires large and powerful models to be extracted. However, commercial applications such as command-and-control interfaces might have severe restrictions and require lightweight models, able to effectively extract semantic information from speech while functioning on devices with limited computing resources. These systems are expected to be plug & play, without needing to adapt to a specific voice or scenario. To be robust to speaker and language variations, a strong speech encoder is needed that can effectively abstract away irrelevant information while extracting semantic knowledge in the form of compact linguistic representations. A smaller model can then be trained to make use of these representations for specific scenarios.

Speech representation models exist [6, 42, 84], but they often use a self-supervised training strategy, a technique that trains a model to predict its own input from a corrupted or incomplete version of it. This paradigm primarily encourages the encoding of acoustic information because linguistic information is not directly useful to solve the training task.

In Chapter 3, we propose to investigate pretraining strategies using masked language modeling on a surrogate ASR task. The model learns to reconstruct masked portions of the speech input, forcing it to capture essential linguistic information. These pretrained representations are then used as input to train small models for specific SLU tasks. We use few training examples to effectively demonstrate that the information contained in the learned representations is sufficiently well organized that it can be extracted by a downstream model very efficiently.

1.1.2 To what extent does multitask learning, combining spoken language understanding with speech recognition, improve performance in low-resource scenarios?

Spoken language understanding applications cannot rely on large datasets because generating training data is impractical and expensive. Instead, the underlying models need to be able to learn from as few examples as possible, something that is intrinsically difficult with artificial neural networks, which are notoriously data-hungry.

Already in the early days of machine learning, it has been proposed that training a model on multiple tasks at the same time reduces the number of training examples required for good generalization, while improving the performance in single tasks compared to models trained on one unique task [8].

In Chapter 4, we propose to apply this to the domain of spoken language understanding and investigate whether a model can learn more efficiently when it is trained on both speech recognition and a SLU task (e.g., intent recognition) simultaneously, than when it is trained for the SLU task alone. We make the following two hypotheses: firstly, by sharing parameters across tasks, the model learns to represent the input data more efficiently, which improves the performance on the low resource SLU tasks; and secondly, by benefitting from the additional ASR data, fewer annotated SLU examples are necessary to train the model.

1.1.3 How can pretrained speech-to-text Transformers be successfully adapted to SLU tasks while keeping low data requirements and computational footprint for the downstream tasks, and while retaining previously acquired knowledge?

The release of pretrained speech-to-text Transformers, trained on massive datasets, has revolutionized ASR, but they cannot be directly applied to SLU tasks and using them for low-resource SLU scenarios remains a challenge. The computational and data requirements of these models are often incompatible with the constraints of real-world SLU applications, particularly those deployed on resource-limited devices.

Furthermore, fine-tuning pretrained models can lead to a degraded generalization due to the catastrophic forgetting phenomenon, which is when a model overwrites its own knowledge when observing new training examples.

With this research question, we aim to bridge the gap between the pretrained speech-to-text Transformers and their potential in SLU. By developing effective methods to adapt these models to various SLU tasks, we can unlock their power for a broader range of applications. This is crucial in scenarios where labeled SLU data is scarce or where computational resources are limited, such as in multilingual settings or on-device applications.

This research question aligns with the growing interest in transfer learning and model adaptation in the SLU community. In Chapters 5 and 7, we explore techniques like adding task-specific decoder modules, freezing parameters and pruning embedding layers to efficiently adapt pretrained models to specific tasks. By investigating these and other approaches, we aim to contribute to the development of more accessible and effective SLU systems that can function effectively in real-world environments. In particular, we investigate approaches that do not require large training datasets and maintain a relatively small SLU model footprint.

In Chapter 7, we explore different strategies for training and adapting Transformer models to handle the complexities of speech across multiple languages, while also investigating techniques to mitigate performance drops due to catastrophic forgetting. We use the task of Spoken NER to illustrate our research in this domain.

1.1.4 How can we use text-based NLU models to build training datasets for multilingual SLU research?

The abundance of text data and the rapid progress in natural language processing present a unique opportunity to address the scarcity of labeled SLU datasets. The high cost and complexity of annotating speech data, particularly for diverse languages and domains, pose a significant bottleneck for SLU research.

This challenge is further amplified in the context of multilingual SLU, where the lack of annotated data for low-resource languages can severely hinder progress. This research question aims to explore how text-based NLU models, trained on vast amounts of textual data, can be leveraged to generate labeled speech data for SLU tasks across multiple languages.

Answering this research question is important to democratize access to SLU resources and accelerate research in this field. By automating the annotation of speech data using existing text-based models, we could drastically reduce the reliance on manual annotation, enabling faster development and broader accessibility of multilingual SLU systems. Recent advancements in multilingual NLU models, such as mBERT [23] and XLM-R [18], have shown promising results in transferring knowledge across languages. This opens up the possibility of using these models to generate annotations for speech data in multiple languages, even for those with limited labeled resources. However, challenges remain in ensuring the quality and accuracy of these automatically generated annotations, as well as in adapting them to the specific nuances of spoken language.

In Chapter 6, we propose to build a dataset for multilingual Spoken NER by using a text-based model to generate annotations. We also build an evaluation dataset entirely validated by human hand to make sure that the labels are reliable. Finally, we explore the potential of transfer learning from NLU to SLU and from English-only to multilingual NER.

1.1.5 What are the benefits of multilingual Spoken NER systems compared to monolingual Spoken NER systems?

The growing demand for SLU systems that can operate across multiple languages necessitates a deeper understanding of the advantages and challenges of multilingual approaches. Multilingual Spoken Named Entity Recognition (Spoken NER) models, in particular, offer the potential to streamline information extraction across multiple languages.

This research question seeks to quantify the benefits of multilingual Spoken NER models over their monolingual counterparts. By exploring how shared representations and transfer learning across languages can impact performance, generalization, and efficiency, we aim to identify the scenarios where multilingual models offer significant advantages.

The current landscape of SLU research is increasingly shifting towards multilingual approaches, driven by the need for inclusivity and the desire to leverage linguistic diversity for improved model robustness. However, the challenges of varying data availability and linguistic complexities across languages remain significant hurdles. This research seeks to shed light on these challenges, identify strategies for mitigating them, and ultimately contribute to the development of more effective and equitable multilingual Spoken NER systems.

Chapter 7 investigates the potential benefits of multilingual Spoken NER models, such as improved generalizability and performance. It also explores the challenges associated with multilingual SLU, such as performance differences across languages due to varying data availability or language complexity. This research aims to identify scenarios where multilingual Spoken NER offers significant advantages and areas where monolingual approaches might still be preferable.

1.2 Outline

The remainder of this thesis is divided into seven chapters. Chapter 2 introduces background concepts necessary to understand the techniques developed in the subsequent chapters. We lay out the landscape of current research in the topics covered in this thesis, their challenges and the datasets employed to train the architectures developed in thee following chapters.

In Chapters 3 to 5, we explore strategies for efficiently training spoken language understanding models on low resource tasks such as intent recognition, slot filling and spoken named entity recognition. Chapter 3 specifically looks at how to build semantic representations for speech, by exploring strategies for training a robust speech representation model. After training, the learned representations generated by the model are used to train lightweight downstream models for intent recognition with minimal data and computation requirements. In Chapter 4, we investigate the

potential of multitask learning, where SLU and ASR tasks are combined in a single training objective. Learning two tasks at the same time encourages the model to identify synergies and build robust representations while using more data than what is available for the SLU task alone. Consequently, training the model requires fewer SLU examples than without the multitask learning objective. Finally, Chapter 5 proposes techniques for adapting large pretrained speech-to-text models for specific SLU tasks, and more particularly which modifications to apply to the pretrained model to make it compatible with the new task with minimal training efforts. We also look at how we can integrate domain knowledge as architectural constraints to facilitate the learning process.

In Chapters 6 and 7, we extend our research focus to multilingual SLU. Chapter 6 introduces MSNER, a multilingual dataset for Spoken NER, built to facilitate cross-lingual SLU research. The construction of this dataset was done in a semi-annotated manner, with evaluation datasets in four languages re-annotated completely by hand to ensure gold standard labeling. Extensive analyses of the annotations before and after human intervention assess the quality of the subsets and emits a number of considerations for its future uses in research. Chapter 7 delves into multilingual Spoken NER using this new dataset, applying techniques explored in the previous chapters to this particular setting. We also investigate techniques to mitigate catastrophic forgetting during multilingual fine-tuning. We propose three techniques in particular. The first is directly derived from observations in Chapter 5 and proposes, after fine-tuning, to reset the embeddings to their original value. The second builds on the first technique by bringing a more formal approach, effectively freezing partially the embeddings during fine-tuning to protect the pretrained embeddings from undesirable updates. At last, we introduce experience replay with synthetic data, a method that leverages the pretrained model to generate pretraining data that can be used during training to mitigate catastrophic forgetting.

Finally, Chapter 8 concludes this thesis, summarizing our contributions to the research community. We specifically come back to the research questions from Section 1.1 and attempt to answer them using what we have learned in Chapters 3 to 7. We end up with a discussion on promising directions for future research in this field.

Chapter 2

Background

2.1 Artificial Neural Networks and Deep Learning

Artificial neural networks are function approximation machines loosely inspired by the biological neurons. In their most basic form, they consist of interconnected nodes (artificial neurons) arranged in layers. Each connection between neurons has an associated weight, which determines the signal strength transmitted between them. These weights are the **model's parameters** that need to be optimized with regard to an objective function. The optimization of a machine learning model is called training, a process during which the parameters defining the model are learned.

Roughly, a model takes one or multiple inputs, and after a series of internal computations, generates one or more outputs. The transformation of the inputs into outputs can be viewed as the model solving a task. For example, given an image, the machine learning task could be to predict whether the image contains a cat. The ultimate goal is to train a model by optimizing its parameters, so that it can generalize well to new observations, *i.e.* it can accurately predict an output for input data absent from the training set.

Deep learning assumes that the learning problem can be decomposed in several simpler functions [31]. Neural networks are thus organized in different layers, with an input layer that typically maps the input data to the model's dimensional space, followed by a number of hidden layers and finally, an output layer that maps the data to the desired output dimension. It has been shown that deep neural networks learn hierarchical concepts and representations of the input data [44, 87], and that deeper networks empirically result in better generalization [31].

Training a neural network involves an iterative process of exposing it to a large dataset of examples paired with their desired output. During each iteration, the following

steps are performed:

1. Forward Pass: The input data x is propagated through the network layer by layer to compute the model's prediction $h(x)$ given the current parameters' value.
2. Loss Calculation: The network's output is compared to the expected outputs y using a loss function, $\mathcal{L}$, that quantifies the distance between the predicted and actual values.
3. Back propagation: The gradient of the loss with regard to the model's parameters is computed. We talk of back propagation because the chain rule of calculus is used recursively to compute the derivatives at each layer, from the last to the first [31].
4. Parameters Update: The gradient descent algorithm uses the gradients computed at the previous step to update the parameters in the direction of the negative gradient [10]. A hyperparameter called the learning rate controls how much the parameters are updated.

Through repeated training steps, the network progressively adjusts its weights, allowing it to learn the underlying patterns and relationships within the data. In practice, since the computational cost of computing the loss function and its gradient increase with the number of examples in the training set, a training step is performed on a minibatch sampled from the training set, with a typical size ranging from a few to a few hundreds examples [31]. This algorithm is known as **stochastic gradient descent**.

2.1.1 Overfitting and Regularization

As we have mentioned, the central challenge in machine learning is to train a model to generalize well, *i.e.* to perform well on unobserved inputs [31]. This is ensured by monitoring both the **training error**, computed on the training set, and a **generalization error**, measured on a held-out validation set. If the training error does not decrease over time, the model is not learning. However, if the training error is getting lower but not the generalization error, the model is **overfitting**.

To understand why this is happening, we have to consider the **capacity** of the model, compared to the size and diversity of the training set. The model capacity is roughly defined by the number of functions that it can represent, or put simply, its number of parameters and the complexity of the inner operations. A model with a large capacity will be able to memorize the training set. Instead, we want the model to identify patterns and learn to extract high level features and representations from the data, much like one would summarize a text by listing its relevant arguments. Overfitting can also occur when the optimization algorithm is not properly configured, for example by setting a learning rate too large or too many training steps. This last example is easily controlled with **early stopping**, a technique which interrupts the training process as soon as the generalization error increases for more than a predetermined

number of steps. Finally, significant discrepancies between the training and testing data distributions might lead to a model that performs well on the training set but poorly on the testing set, highlighting the importance of carefully choosing appropriate training and testing sets, and monitoring both the training and the generalization errors.

Aside from choosing the right model architecture and the appropriate datasets, numerous methods have been proposed to mitigate overfitting, that can be divided into **regularization methods**, **noise-injection techniques** and **data augmentation**. Regularization methods, such as L1 and L2 regularization, add a penalty term to the loss function, discouraging the model from assigning overly large weights to specific features, thereby reducing model complexity. Noise-injection techniques, such as Dropout [41] and LayerDrop [26], add noise to the training process to make it more robust to variations in the input data and improve its generalization ability. Dropout [41] randomly sets to zero the output of a neuron based on a predefined probability, while LayerDrop randomly skips layers. Finally, data augmentation involves creating artificial variations of the training data through input transformations. This increases the diversity of the training set and helps the model learn more robust features.

2.1.2 Objective Functions

As we have seen, gradient-based learning requires the definition of a differentiable loss function, which quantifies the error of the network by comparing its prediction to an expected output. The loss family type is defined by the task that the model must solve. For regression problems, where the objective is to predict a real value, such as stock price prediction, an error function that measures the average distance between two vectors will be preferred, like the mean squared error, which is the average Euclidean distance between the model's predictions and the expected outputs. For classification tasks, where the model must assign a label to a dataset example, a typical loss function is the **cross-entropy**. In the binary case, where the label y can take one of two values, cross-entropy is defined as follows:

$$\mathrm{H}(y,p) = \begin{cases} -\log p & \text{if } y = 1 \\ -\log 1-p & \text{otherwise} \end{cases} \tag{2.1}$$

where p is the model's predicted probability that the example belongs to the positive class. For convenience, this can be rewritten as $\mathrm{H}(y,p) = -\log p_k$, with $p_k = p$ when $y = 1$ and $p_k = 1-p$ otherwise. When the classification problem extends to more than two classes, i.e. y can take one of K values, the model then generates a probability distribution $p = \mathrm{P}(y|x)$ over the K possible classes, that is, a k dimensional vector whose elements p_k represent the model's predicted probability that input x belongs to class $k \in \{1, ..., K\}$. We can generalize Equation 2.1 in the following way:

$$\mathrm{H}(y,p) = -\sum_{k=1}^{K} t_k \cdot \log p_k \tag{2.2}$$

where t_k is a binary indicator equal to one when $y = k^*$ and zero otherwise, k^* being the ground-truth label. This equation can be simplified to

$$\mathrm{H}(y, p) = -\log p_{k^*} \tag{2.3}$$

where $p_{k^*} = \mathrm{P}(y = k^*|x)$ is the model's predicted probability that input x belongs to class k^*.

By comparing the model's prediction to a one-hot vector, using cross-entropy will often lead to overconfident predictions, especially when the class distribution is unbalanced or the number of classes K is large. **Label smoothing** [89] acts as a regularization method to soften the ground truth labels. Instead of using a one-hot encoded vector (where only the correct element is set to 1), a smoothed distribution is used that assigns a small probability mass to all possible tokens, including incorrect ones. The amount of smoothing applied is proportional to a hyperparameter λ called the smoothing factor. This encourages the model to consider more diverse predictions. Formally, the binary indicator t_k in Equation 2.2 is replaced with $1 - \lambda$ if $y = k$ (or equivalently $t_k = 1$) and $\frac{\lambda}{K-1}$ otherwise. This technique encourages the model to consider a broader range of possibilities during training, leading to improved generalization and robustness to unseen data.

Another technique to mitigate issues from class imbalances is to artificially increase the weight of rare classes in the loss. To do this, we use the balanced cross-entropy, $\tilde{\mathrm{H}}(y, p)$, which introduces a parameter α_k generally set as the inverse class frequency of class k in the training dataset. Equation 2.1 is rewritten as:

$$\tilde{\mathrm{H}}(y, p) = -\alpha_k \cdot \log p_k \tag{2.4}$$

Until now, we have considered that all the examples in a minibatch should have an equal weight in the total loss, by taking either the sum or the average. In most cases, this is a good decision, but it can sometimes lead to a model consistently predicting easy or frequent examples correctly, while systematically failing on rare or challenging examples. To avoid this, the **focal loss** [55] was proposed as a method to improve object detection in computer vision systems. In practice, it reweighs the contributions of each example in a minibatch with a factor $(1-p)^f$. The value $1-p$ is the model's uncertainty that a certain example belongs to the correct class k^*, and f is the focus, a hyperparameter that smoothly adjusts the rate at which easy examples are down-weighted [55]. An example misclassified with high uncertainty ($1 - p \to 0$) will not affect the loss. However, as the uncertainty decreases, the relative weight of this particular example in the total loss decreases. By down-weighting well-classified examples predicted with a high probability (*i.e.* "easy" examples), we are assigning more weight to difficult examples. Further extending Equation 2.4:

$$\mathrm{FL}(y, p) = -\alpha_t (1 - p_k)^f \log p_k \tag{2.5}$$

Note that Equation 2.5 corresponds to Equation 2.3 when $\alpha = 1$ and $f = 0$, and to Equation 2.4 when $f = 0$. Both Equations 2.4 and 2.5 can be trivially extended to K classes by defining $\alpha = (\alpha_1, ..., \alpha_K)$. Equation 2.3 then becomes:

$$\tilde{\mathrm{H}}(y, p) = -\alpha_{k^*} (1 - p_{k^*})^f \log p_{k^*} \tag{2.6}$$

2.1.3 Pretraining, Fine-tuning and Forgetting

Large neural networks trained on massive, generic datasets, have become a cornerstone of modern machine learning. These **pretrained models**, sometimes named foundation models, are often trained using **self-supervised methods**, which leverage the data itself for learning without the need for a labeled dataset. The training objective is encouraging the model to learn generic representations of the input data that can then be leveraged for a wide range of downstream tasks. This is called pretraining because it is considered a preliminary step, as it lays the groundwork for subsequent supervised training, called **fine-tuning**. In the context of a supervised learning task, pretraining can act as a regularizer and a form of parameter initialization [31].

Fine-tuning offers several advantages:

- Reduced Training Time: By leveraging the pretrained weights, fine-tuning requires significantly less training data and computational resources compared to training a model from scratch.
- Improved Performance: The pretrained model provides a strong foundation for learning, often leading to better performance on the target task compared to training a smaller model from scratch.

An important assumption when fine-tuning a pretrained model is that the downstream task shares enough common ground with the pretraining task, so that the knowledge and representations learned during pretraining can be transferred to the downstream task. This is why pretraining is sometimes called **representation learning** and fine-tuning, **transfer learning**.

If the pretrain/fine-tune paradigm has been widely adopted in the whole research community, it can however lead to **catastrophic forgetting**. This is when the model's performance on the original pretraining tasks degrades as it learns the new task. During fine-tuning, the loss is back-propagated and the parameters updated, potentially overwriting previously acquired knowledge. Although this is not always undesirable, in some cases, we want the model to keep its previous capabilities.

Multiple ideas have been proposed to mitigate this issue, broadly: regularization techniques, architectural methods and rehearsal approaches [98]. The first acts on the optimization for example by adding regularization terms to the loss, or consolidating the weights during training [49]. The second increases the capacity of the network to accommodate for new data [82]. Finally, in rehearsal methods, a memory of past examples is stored and mixed with new examples during training [81]. If the topic of catastrophic forgetting has received some attention, specific research in speech processing remains sparse (see for example [98]).

2.1.4 Multitask Learning

Unlike the pretrain/fine-tune paradigm, with multitask learning, the model learns to solve multiple tasks in parallel. The assumption is that learning these multiple tasks leads to a more robust model, generalizing better than if it was optimized for one and only one task. Baxter [8] makes two interesting conclusions: "Learning multiple related tasks reduces the sampling burden required for good generalization, at least on a number-of-examples-required-per-task basis" and "Bias that is learnt on sufficiently many training tasks is likely to be good for learning novel tasks drawn from the same environment.". We will take advantage of this later, in Chapter 4.

Multitask learning (MTL) is most often realized by building a hybrid loss that is a sum of the losses $\mathcal{L}_i$, relative to each of the T tasks, weighted by a factor λ_i.

$$\mathcal{L}_{\text{MTL}} = \sum_{i=1}^{T} \lambda_i \cdot \mathcal{L}_i \tag{2.7}$$

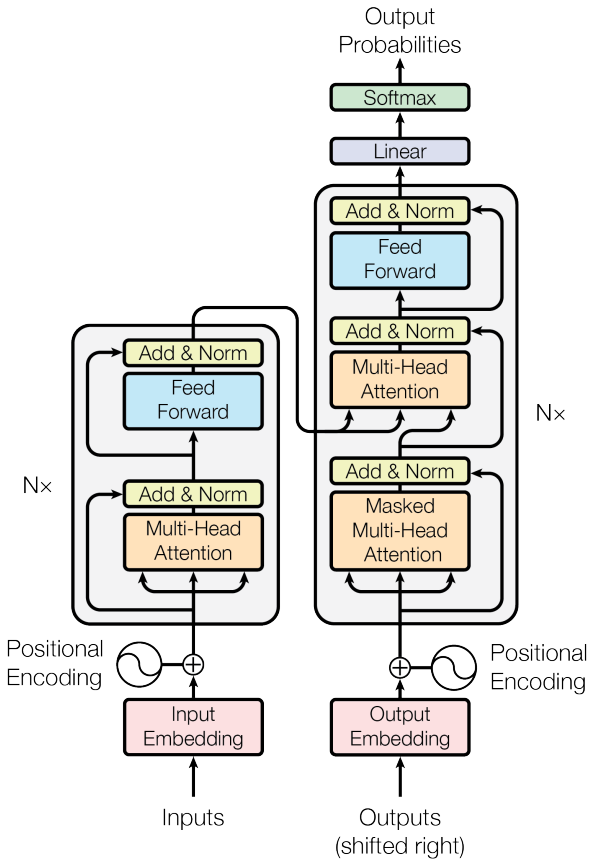
In this case, a trade-off occurs, caused by competing optimization objectives. The same assumptions as in Section 2.1.3 should hold that the multiple objectives share a common ground and that the ability to solve one task helps on solving others. It is easy to imagine many examples for which this assumption might hold or not, depending on the tasks and datasets considered.

2.2 Transformers

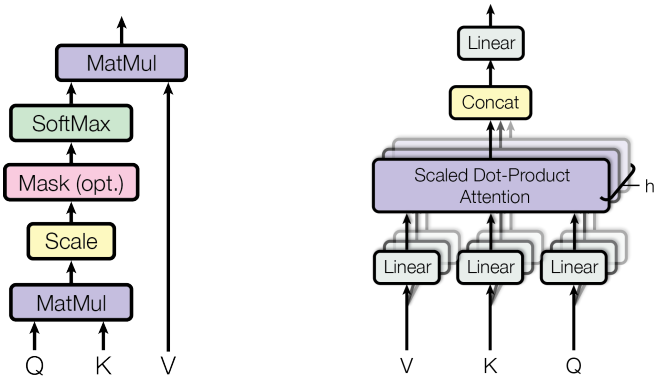
A Transformer [99] is a specific neural network architecture that has become dominant in practically all areas of artificial intelligence, leading to major advancements in NLP, computer vision, speech processing. It shines in its ability to efficiently process large amounts of sequential data and learn short and long range dependencies. In language models, it processes input sequences of tokens to predict the next token and generate contextual embeddings representing the meaning of each token in context. Each Transformer block builds richer contextual representations by combining information from the previous layer's representations of neighboring tokens. Originally, it was proposed as an encoder-decoder architecture (Figure 2.1a), ideal for sequence-to-sequence tasks like machine translation, image captioning or speech recognition. The encoder (left part in Figure 2.1a) maps the input sequence $\text{x} = (x_1, ..., x_n)$ to $\text{h} = (h_1, ..., h_n)$. The decoder (right part in Figure 2.1a) then uses this encoded representation h to generate the output sequence $\text{y} = (y_1, ..., y_m)$, one item at a time. At each step, the model generates a new output y_i given input sequence x and the sequence of previously generated elements $(y_1, ..., y_{i-1})$.

$$\text{h} = \text{Encoder}(\text{x}) \tag{2.8}$$

$$\text{y} = \text{Decoder}(\text{h}) \tag{2.9}$$



(a) Transformer architecture



(b) Scaled dot-product attention

(c) Multihead attention

Figure 2.1: The Transformer Architecture. Figures from Vaswani et al. [99]

2.2.1 Attention

Self-attention [99] is a key mechanism in Transformers, enabling the model to consider information from distant tokens within the context. It can be thought of as a way to build contextual representations of an object (e.g. a word, a subword token, a speech frame) that integrate information from surrounding objects within the context. In backward-looking or **causal attention**, the context is restricted to prior items in the input sequence. When the attention is unrestricted, like in a Transformer encoder, it is said to be **bidirectional**, because attention scores are computed on both left and right contexts. The self-attention computation involves comparing vectors using dot products, resulting in scores (Figure 2.1b). These scores are normalized using softmax, producing weights that indicate the proportional relevance of each input to the current focus element. A weighted sum of the input vectors, based on these attention weights, generates the output representation for each token. To distinguish between the different roles each input embedding plays (queries, keys, and values), Transformers use weight matrices (W_Q, W_K, W_V) to project each input vector into corresponding representations of dimension d_k. The score between the current focus and a context element is calculated as the dot product of the query and key vectors, scaled by the square root of the key vector dimension to prevent numerical issues. In Transformer decoder blocks, an additional attention layer, named **cross-attention** or encoder attention, computes the relevance of the encoder hidden states h with regard to the partial sequence y.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.10}$$

In the case of self-attention, where the keys, queries and values originate from the same source X:

$$\begin{cases} Q = W_q \cdot X \\ K = W_k \cdot X \\ V = W_v \cdot X \end{cases} \tag{2.11}$$

In encoder-decoder attention, K and V are linear projections of h, and Q from shifted outputs. The entire self-attention process for a sequence can be parallelized using matrix multiplication, improving efficiency. However, to maintain the autoregressive property (predicting the next item given previous ones), future items are masked out in the self-attention computation.

Multi-head attention [99] enhances self-attention by allowing the model to jointly attend to information from different representation subspaces. It involves multiple parallel attention functions operating on different projections of queries, keys, and values, each capturing different aspects of the input relationships. The outputs are then concatenated and projected again to yield the final states (Figure 2.1c).

2.2.2 Transformer Block

A Transformer block comprises a self-attention layer, potentially a cross-attention layer, a feedforward layer, residual connections, and layer normalization. The feedforward layer consists of position-wise networks that apply identical transformations to each position, while residual connections and layer normalization aid in training deeper networks and stabilizing the training process. Note that in Figure 2.1a, the layer normalization comes after the other operations (post-layer normalization). Recent implementations advocate for pre-layer normalization, which is suggested to further improve training stability [109].

Transformer blocks can be viewed as operating on individual token vectors (residual streams), where the multi-head attention mechanism moves information from neighboring tokens into the current token's stream. The residual connections continuously pass information up from earlier embeddings, while other components add new views of this representation.

2.2.3 Embeddings and Output Layers

In NLP Transformers, the input is represented as a sequence of tokens, each associated with a learned embedding vector. This embedding is obtained by adding two components: a **token embedding**, which captures the semantic meaning of the token, and a **position embedding**, which encodes the token's position in the sequence. Token embeddings are typically obtained from a pretrained embedding matrix, where each row corresponds to a specific token in the vocabulary. Alternatively, one-hot vectors can be used to select the appropriate row from the embedding matrix.

Position embeddings are necessary because token embeddings alone do not convey information about sequence order, and the Transformer is permutation invariant and would be otherwise unable to make use of the order of the sequence [99]. These can be either learned or fixed functions. A common approach is to use sinusoidal functions with varying frequencies to represent the relative positions of tokens, allowing the model to easily learn to attend to tokens based on their distance from each other. Other approaches use relative or learned position embeddings [20, 29]. The final input representation for a Transformer is a sequence of vectors where each individual vector is the sum of the token embedding and the corresponding position embedding for that position. This matrix serves as the input to the Transformer's layers, enabling it to process both the meaning and the order of tokens in the input sequence.

Finally, an output layer maps the hidden states of the model to the desired output dimension (e.g. the number of classes in a classification task). In most implementations (including Vaswani et al. [99]), the output layer's weights are tied to the decoder's embedding weights, meaning that the values for both matrices are forced to be the same. This simple trick reduces the effective number of parameters in the model and helps prevent overfitting.

2.2.4 Transformers Taxonomy

Transformer models can be classified into three categories based on their components. First, **Transformer encoder-decoder** models combine both the encoder and decoder components [53, 99, 105]. The encoder processes the input sequence and generates a representation, which is then used by the decoder to generate the output sequence. This architecture is well-suited for tasks that require understanding the input and generating a corresponding output, such as machine translation and text summarization. They are also particularly well suited for tasks dealing with multiple modalities such as image captioning (image-to-text) and speech recognition (speech-to-text).

Transformer encoder models are a type of model that focuses on understanding and representing the input sequence [23, 36, 56]. They consist solely of the encoder part of the Transformer architecture, which processes the input sequence and generates a contextualized representation for each item in the sequence. This representation captures the meaning of the item in relation to the surrounding elements, making it useful for tasks like text classification, sentiment analysis, and named entity recognition.

Transformer decoder models are primarily designed for text generation [2, 11, 95]. They use the decoder part of the Transformer architecture, which takes the previously generated outputs as input and predicts the next element in the sequence. This autoregressive generation process allows for the creation of fluent and coherent text, making decoder-only models suitable for developing conversational agents like chatbots.

2.3 Natural Language Processing

Natural Language Processing (NLP) is a field of artificial intelligence that focuses on enabling computers to understand, interpret, and generate human language in a way that is both meaningful and contextually relevant. In simpler terms, NLP aims to bridge the gap between human communication and computer understanding, allowing machines to process and analyze vast amounts of written text to perform various tasks such as machine translation, sentiment analysis, conversational agents, or text summarization. In the remaining of this section, we will discuss common NLP tasks.

2.3.1 Text Tokenization

In practice, neural networks operate on numerical representations, not raw words or characters. To solve this, a **subword tokenization** algorithm like Byte-Pair Encoding [28] and SentencePiece [50] is used to split the text into subword units. By learning a vocabulary of subword tokens, the tokenizer strikes a balance between character-level granularity and word-level semantics. The decision to split a word or subword into smaller units is guided by frequency analysis and algorithmic rules. Common words might remain intact, while rarer or morphologically complex words

are often segmented into smaller units. This strategy allows the model to handle out-of-vocabulary words and morphological variations effectively.

An important parameter of subword tokenization is the vocabulary size, which determines the maximum number of distinct tokens the model encounters. A larger vocabulary offers greater expressiveness and potentially captures finer-grained linguistic nuances. However, it also increases model complexity and demands more extensive training data to ensure sufficient representation for each token. Conversely, a smaller vocabulary limits model capacity but requires fewer examples for adequate training.

Depending on the prediction task, special tokens might be added to the vocabulary. In language modeling, it is often convenient to add a token that marks the beginning of the text input, and one that marks the end. Finally, as we will see in Section 2.4.2 (page 20), language and task tokens can be used to indicate the input language or the prediction task respectively.

2.3.2 Language Modeling

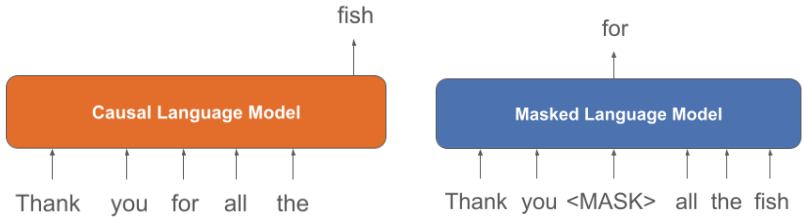
At its core, a Language Model (LM) tries to predict the next word in a sentence given the previous words. By training large language models on massive datasets of text, such as Wikipedia or news articles, we end up with very powerful engines able to generate meaningful text very effectively. Autoregressive or **Causal Language Models** (CLM) are trained to predict the next token in a sequence. First, an initial input, or prompt, is given. From there, the model will generate, one token at a time, what it considers to be the most probable sequence of token given this initial input. When the model outputs a special token `</s>`, the generation stops. Formally, given a sequence of tokens $\mathrm{t} = (t_1, ..., t_i)$ and a vocabulary of size V, at each step, the model prediction is

$$t_{i+1} = \underset{j}{\operatorname{argmax}}\, p(\hat{t}_{i+1} = j | t_1, ..., t_i) \tag{2.12}$$

where $p(\hat{t}_{i+1} = j | t_1, ..., t_i)$ is the model's predicted probability that the next token in the sequence is the j^{th} token in the vocabulary. Therefore, the model must learn to estimate this probability, which is done following the steps in Section 2.1.

CLMs used to rely almost exclusively on Recurrent Neural Networks (RNN) [45], until the Transformer [99] was introduced. Nowadays, most CLMs, like GPT and Llama, are decoder-only Transformers [2, 11, 95].

Masked Language Models (MLM) use a different approach. Taking advantage of the bidirectional properties of encoder-only Transformers, this method works by randomly masking (hiding) a portion of the input tokens and training the model to predict the masked tokens based on the surrounding context [23]. This approach has proven highly effective in pretraining language representation models such as BERT and RoBERTa, as it forces the model to learn robust features that capture the meaning of words in context. By learning to fill in the blanks, the model develops a deep understanding of language structure, semantics, and relationships between words



(a) Causal Language Modeling task **(b)** Masked Language Modeling task

Figure 2.2: Causal and Masked Language Models

[23, 44, 64]. Unlike CLMs, which only have access to previous tokens, MLMs can benefit from both left and right context. Additionally, they can fill multiple masked tokens in parallel, instead of one at a time [64].

A trained MLM can be seen as a powerful feature extractor that can be combined with additional modules to perform NLU tasks like text classification, sentiment analysis, and named entity recognition. One well-known example is BERT [23], a bidirectional Transformer encoder that is widely used for various NLP tasks. Later on, RoBERTa [56] was released, an improved BERT model pretrained with additional data and carefully tuned hyperparameters.

2.3.3 Named Entity Recognition

Named Entity Recognition (NER)[1] is a fundamental task in NLP that focuses on identifying and classifying named entities within text. Named entities are specific categories such as person names, organizations, locations, dates, quantities, and more. By automatically extracting these entities, NER enables deeper analysis and understanding of unstructured text. To illustrate this, consider the sentence: "I'd like to book a flight to Paris on Tuesday evening." NER would identify "Paris" as a location, "Tuesday" as a date, and "evening" as a time. This structured information is crucial for various applications, including information retrieval, question answering, sentiment analysis, and language understanding.

In NER, there is not one unified label set. Both generic and specialized datasets exist with their own label sets defined. Specialized datasets might cover large amounts of topics with specific vocabulary and entities. For example, a NER system for doctors would include medications, dosages, medical reasons, etc. [97], and biomedical entities include names of proteins, chemical, disease, or species [19]. Other datasets provide more generic entities that cover broader landscapes. One of the most widely used is

[1] Some parts of this section were adapted from Q. Meeus, M.-F. Moens, and H. Van hamme. "MSNER: A Multilingual Speech Dataset for Named Entity Recognition". In: *LREC-COLING ISA-20 Workshop*. 2024. arXiv: 2405.11519, Section 2.

CoNLL-2003 [93], although it comes with only four entity types (LOC, ORG, PER and MISC). OntoNotes v5 enriches this set to 18 classes (Table 6.2 on page 69), to include things such as numbers, dates, and laws. Its high quality makes it one of the most widely used NER datasets, although it only covers three languages: English, Arabic and Chinese. Another notable mention is Tedeschi et al. [90], which adds a few more generic classes to OntoNotes definitions to cover things such as animal names, diseases, food, and plants, and released a dataset derived from Wikipedia where named entities were annotated automatically with an annotation pipeline that effectively combined pretrained language models and knowledge-based approaches. A follow-up dataset was published covering more languages [91].

This task is often framed as a token-level classification problem where each word or subword must be assigned an entity tag. The widely used BIO (Beginning-Inside-Outside) tagging scheme [76] assigns each token a label indicating its role within a named entity. "B" marks the beginning of an entity, "I" represents an inner token, and "O" denotes a token that is not part of an entity. This marker, together with the entity type, makes the target for the classification task. Other variations of the BIO framework exist, offering different trade-offs in complexity and representation. While BIO efficiently captures entity boundaries, it cannot represent nested entities, where one entity is contained within another [65].

Modern NER approaches use pretrained language models, such as BERT [23], with additional layers fine-tuned on specific NER datasets. Sometimes, Conditional Random Fields (CRFs) [51], can be incorporated to learn the transition probabilities between the label classes [96].

2.4 Speech Processing

Speech Processing[2] is the study of speech signals and their processing methods by computers. Although many speech processing tasks exist, we will focus here on the tasks relevant to this thesis, namely speech features extraction, automatic speech recognition and spoken language understanding.

2.4.1 Speech Features Extraction

The first step in the speech processing pipeline generally involves transforming raw audio waveforms into informative acoustic feature vectors (Figure 2.3). These vectors, representing short segments of the signal, capture essential speech characteristics. A commonly used feature representation is the log Mel spectrum, which mimics human auditory perception.

[2] Some parts of this section were summarized from D. Jurafsky and J. H. Martin. *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Upper Saddle River, N.J.: Pearson Prentice Hall, 2009. Online book, Chap. 16.

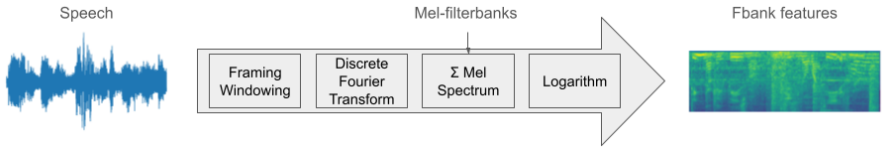


Figure 2.3: Speech Preprocessing Pipeline

1. Digitization: Analog sound waves are first converted into digital signals through sampling (measuring amplitude at regular intervals) and quantization (representing amplitudes as integers).
2. Windowing: The digitized signal is divided into short, overlapping frames using a window function (e.g., Hamming window). This helps analyze the signal as a series of quasi-stationary segments.
3. Discrete Fourier Transform (DFT): Each frame undergoes a DFT, usually implemented with the Fast Fourier Transform (FFT), to extract spectral information. This decomposes the signal into its constituent frequencies, revealing the energy distribution across different frequency bands.
4. Mel Filter Bank: The FFT output is then passed through a Mel filter bank. This bank consists of triangular filters spaced along the Mel scale, a perceptual frequency scale that approximates human hearing. The filters collect energy from specific frequency ranges, with finer resolution at lower frequencies (where human hearing is more sensitive).
5. Logarithm: Finally, we take the log of each of the Mel spectrum values. This step reflects the logarithmic nature of human auditory perception, emphasizing differences at lower amplitudes while compressing differences at higher amplitudes. It also makes the features more robust to variations in input levels.

The resulting log Mel spectrum represents a sequence of feature vectors, each capturing the energy distribution across Mel frequency bands within a short time window. These feature vectors serve as the input to subsequent speech processing models, enabling them to learn patterns and relationships between acoustic features and linguistic units (e.g., phonemes, words) [46].

2.4.2 Automatic Speech Recognition

Automatic speech recognition (ASR) is one of the earliest goals of computer language processing, even older than computers. Today, ASR aims to accurately convert any spoken utterance into its corresponding text representation. While achieving perfect transcription in any environment remains an ongoing challenge, significant progress has been made, opening doors to numerous practical applications.

One key dimension of variation in ASR tasks is vocabulary size. Systems designed for limited vocabularies, like digit recognition or simple voice commands, achieve impressive accuracy. However, open-ended tasks like transcribing videos or conversations, with vast vocabularies, present a greater challenge. The nature of the interaction also influences ASR performance. Speech directed towards machines (e.g., dictation or voice commands) is typically easier to recognize than spontaneous human-to-human conversations. This is because people tend to modify their speech patterns when interacting with machines, speaking more clearly and deliberately. Additionally, the acoustic environment and channel characteristics play a crucial role. Speech recorded in quiet rooms with high-quality microphones is significantly easier to recognize than speech captured in noisy environments or through distant microphones. Another factor impacting ASR performance is speaker variability. Accents, dialects, and individual speech patterns can pose challenges for systems trained on a limited set of voices. For instance, recognizing speech from children or speakers with regional accents can be difficult for a system primarily trained on adult voices speaking a standard dialect. To tackle these diverse challenges, researchers leverage a wide range of publicly available corpora with human-generated transcripts. These corpora, such as LibriSpeech [70], CGN [69], VoxPopuli [102], and many others, encompass various scenarios, including audiobooks, telephone conversations, unscripted dialogues, and even recordings of European Parliament sessions. These datasets help researchers train and evaluate ASR systems on different types of speech, leading to improved performance across the board.

Despite these advancements, ASR still faces significant hurdles. While word error rates (WER) for read speech are relatively low (around 2%), conversational speech still presents a considerable challenge, with WERs ranging from 5.8% to 11% or even higher for specific dialects and difficult acoustic conditions [78, 105]. Nevertheless, ASR has found its way into numerous applications, including smart home devices, personal assistants, call routing systems, and tools for automatic transcription. The technology also holds promise for augmentative communication, assisting individuals with speech or hearing impairments. As research progresses and models become more sophisticated, we can expect ASR systems to continue improving in their ability to understand and transcribe human speech in all its diverse forms. This will undoubtedly unlock even more exciting possibilities for communication and interaction between humans and machines.

Error Rates

The Word Error Rate (WER) is used to assess the performance of ASR systems. It measures the dissimilarity between the reference (ground truth) text transcript and the system's generated transcript. Lower WER indicates better performance, meaning the ASR system produced a transcript closer to the actual spoken words. WER is calculated by finding the edit distance between the reference (the ground-truth) and the hypothesis (the model's prediction). The edit distance refers to the minimum number of edits (insertions, deletions, substitutions) required to transform

the hypothesis into the reference. Formally, if the number of insertions, deletions and substitutions are denoted I, D and S respectively, and the reference contains N words, the WER is given by

$$\text{WER} = \frac{S + I + D}{N} \tag{2.13}$$

Similarly, the Character Error Rate (CER) and the Token Error Rate (TER) compute the same ratio for characters and language tokens respectively.

Connectionist Temporal Classification

Connectionist Temporal Classification (CTC) [32] is a loss function that provides an alternative to the encoder-decoder architecture for automatic speech recognition (ASR). It addresses the challenge of aligning long acoustic input sequences (X) with shorter text outputs (Y). CTC accomplishes this by outputting a character for each input frame and then using a collapsing function to merge repeating characters and eliminate blanks (Figure 2.4 b and c). Blank characters (ϵ) are special symbols added to the vocabulary to handle repeated symbols, like the letter "l" in "Hello".

To make predictions, one approach consists in choosing the most probable character for each frame independently. However, this might not result in the most likely output string due to the many-to-one nature of the collapsing function. To find the most probable output, CTC sums over the probabilities of all possible alignments that collapse to the same output.

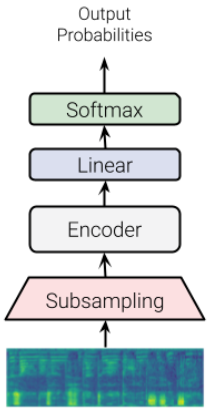
$$P_{\text{CTC}}(Y|X) = \sum_{A \in B^{-1}(Y)} P(A|X) = \sum_{A \in B^{-1}(Y)} \prod_{t=1}^{T} p(a_t|h_t) \tag{2.14}$$

where X is the sequence of speech features, Y is the output sequence, $A = (a_1, ..., a_T)$ is an alignment such as the first line in Figure 2.4b, B is the collapsing function in Figure 2.4b, and $p(a_t|h_t)$ is the probability that symbol a_t is aligned with encoded speech input h_t (output probabilities in Figure 2.4a, h_t is the output of the encoder). This can be efficiently approximated using a modified beam search algorithm (See Hannun [35] for a detailed description). To train this classifier, the conditional probabilities $p(a_t|h_t)$ can be efficiently calculated using the CTC Forward-Backward algorithm or Maximum Likelihood training [32].

CTC assumes conditional independence between output labels given the input, meaning it doesn't inherently learn a language model. Therefore, it is important to consider the incorporation of an external language model and length factor to the loss and scoring function.

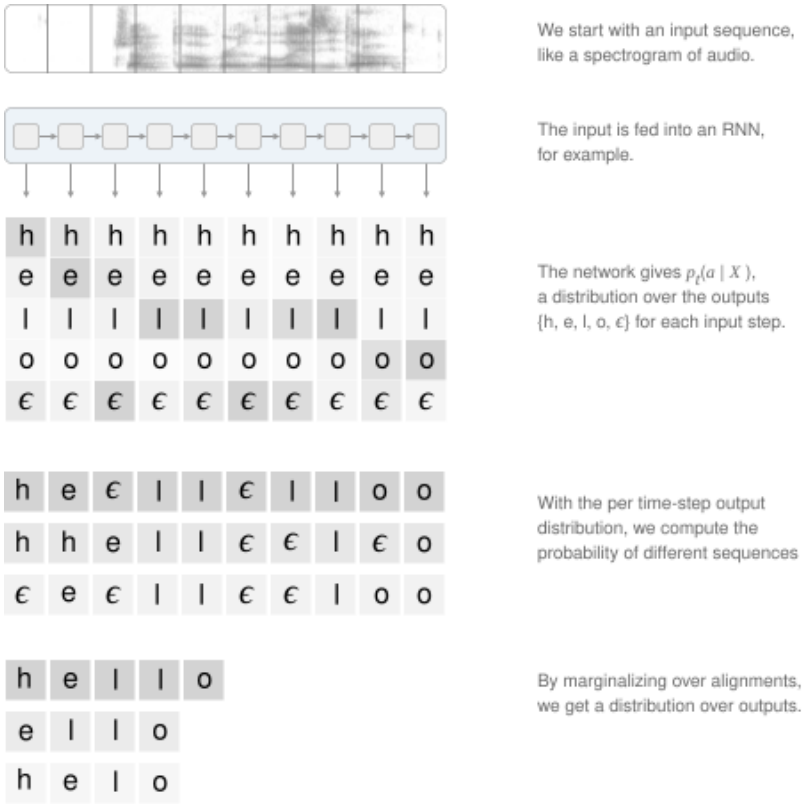
Encoder-Decoder Architectures for ASR

ASR systems often leverage encoder-decoder architectures (Figure 2.1a). This structure is well-suited for speech recognition due to the significant length difference between



(a) CTC model for ASR

(b) CTC collapsing function



(c) CTC alignments

Figure 2.4: From Speech Features to CTC predictions. Figures b and c from Hannun [35]

input audio features and output text representations. In ASR, a convolutional frontend is often used to process the input sequence of acoustic features, derived from the log Mel spectrum, and maps them to a shorter sequence of continuous representations that captures the local dependencies in the audio [25, 68]. The Transformer encoder further processes the output to generate the hidden representations. The decoder then generates the output sequence of characters or subword tokens, typically using an autoregressive approach where each element is predicted based on the previous ones. In this setting, the encoder can be considered as an acoustic model, extracting acoustic features from speech, and the decoder as a language model, generating linguistic representations from the acoustic features. The encoder-decoder can be implemented using either Recurrent Neural Networks (RNNs) [12] or Transformers [75, 105]. During inference, the model estimates the probability of the output string $\mathrm{y} = (y_1, ..., y_n)$ by considering the conditional probabilities of each token given the previous tokens and the encoded input x:

$$\mathrm{p(y)} = \prod_{i=1}^{n} p(y_i | y_1, ..., y_{i-1}, x) \tag{2.15}$$

Greedy decoding or beam search can be used to generate the output text. Beam search is often preferred, especially when incorporating a language model to rescore the hypotheses and improve overall accuracy. The scoring function typically combines the language model score with the encoder-decoder score, along with a length normalization factor.

During training, the model is optimized using the cross-entropy loss, which measures the negative log probability of the correct token at each decoding step. Label smoothing [89] can be used to avoid overconfident predictions and encourage diversity [17]. Teacher forcing, where the decoder receives the correct previous outputs, is often used to guide the training process. Alternatively, a mixture of the correct and predicted outputs can be used to introduce some variability.

The encoder-decoder architecture can be combined with CTC by using a weighted combination of the cross-entropy loss from the encoder-decoder, $\mathcal{L}_{\text{att}}$ and the CTC loss, $\mathcal{L}_{\text{ctc}}$ [106] (Figure 2.5):

$$\mathcal{L}_{\text{hybrid}} = \lambda \cdot \mathcal{L}_{\text{att}} + (1 - \lambda) \cdot \mathcal{L}_{\text{ctc}} \tag{2.16}$$

λ is a real number that balances the weight of each loss in the total loss. Similarly, during inference, the scores from both architectures, along with the language model, can be combined to find the most likely output. In the encoder-decoder architecture, the alignments between speech and text can only rely on the cross-attention between the encoder outputs and the language tokens. In contrast, the CTC objective in the hybrid transformer enforces a monotonic alignment between speech and text, resulting in a more robust model and faster convergence [64].

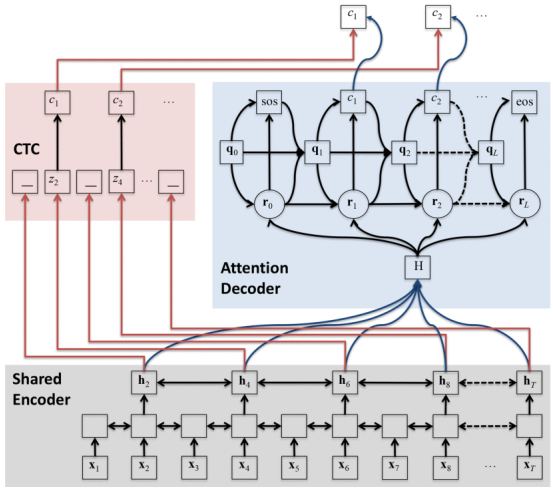


Figure 2.5: Hybrid CTC/Attention Architecture. Source: Watanabe et al. [106].

OpenAI's Whisper

OpenAI's Whisper[3] [75] is a speech-to-text foundation model that was trained on massive amounts of curated multilingual speech to translate speech to English and do multilingual automatic speech recognition (ASR). Pretrained models of different sizes have been released, ranging from tiny (39 million parameters) and small (244 million) to large (1.5 billion). Whisper is a Transformer encoder-decoder that takes audio input sampled at 16 kHz and represented as 80-dimensional Mel-filterbanks features. The Transformer encoder is preceded by a convolutional frontend, which has the effect of reducing the time dimension of the input by a factor of two. Convolutional frontends have been widely used in speech processing, as it has been suggested that they help the model learn local relationships in the input signal [68]. The encoder also uses sinusoidal position embeddings, followed by a number of Transformer encoder layers. The decoder uses learned position embeddings and tied input-output token embeddings [74]. The entire model is trained end-to-end in a multitask setting to produce sequences conditioned on both the speech input and previous tokens predicted by the decoder. The training tasks are language classification, speech transcription and translation, and timestamp prediction. The decoder input is composed of past context, if any, a `<start-of-text>` token, a language token, then either `<translate>` or `<transcribe>` and finally, `<notimestamps>`, a flag that prevents the prediction of timestamps. The prediction ends with a `<end-of-text>` token (Figure 2.6).

[3] This section was adapted from Q. Meeus, M.-F. Moens, and H. Van hamme. "Multilingual Spoken Named Entity Recognition with Transformers (under review)". In: *Natural Language Processing* (2024), Section 2.2.

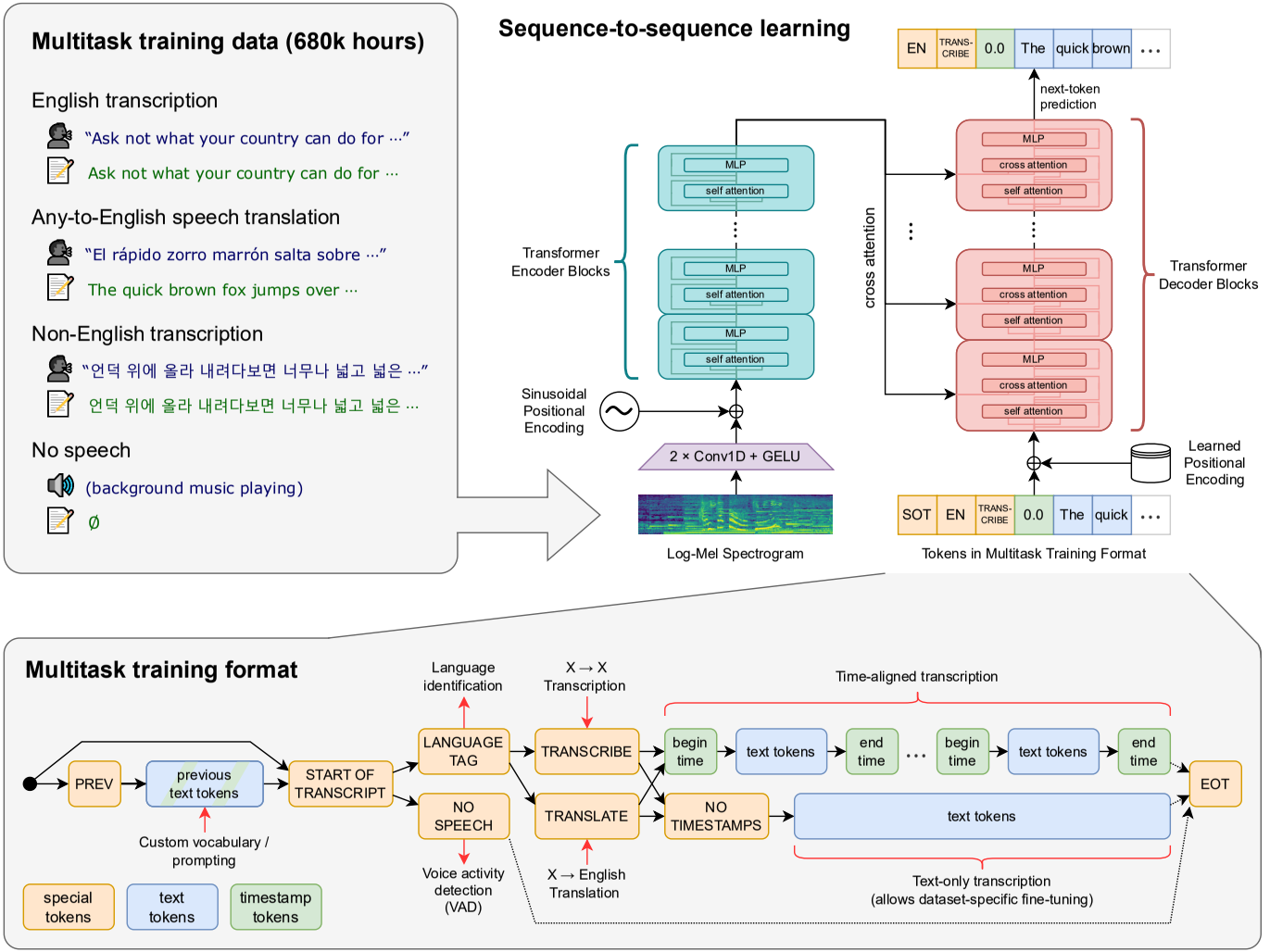


Figure 2.6: Whisper Approach. Source: Radford et al. [75].

2.4.3 Spoken Language Understanding

Spoken Language Understanding (SLU) tackles the complex challenge of interpreting spoken language [21]. It goes beyond simply recognizing the words spoken, aiming to grasp the intent and meaning behind the utterance, or to extract semantic information from speech. SLU presents significant challenges due to the complexity of the speech signal. Meaning is encoded alongside information about the speaker's identity and the acoustic environment, making the extraction of semantic content difficult. Furthermore, spoken language often contains ungrammatical constructions and disfluencies, such as hesitations, interruptions, and self-corrections, adding another layer of complexity.

Designing effective SLU systems requires a sophisticated interpretation strategy that incorporates computational models for various types of knowledge. This strategy must account for the inherent imperfection of these models and the potential errors introduced by the Automatic Speech Recognition (ASR) component, which transcribes the speech into text.

In this regard, it is similar to NLU, with the substantial difference that the inputs are speech signals and not texts. There are two main approaches to achieving SLU:

- **Pipeline systems**: Early SLU systems primarily relied on text-based NLU methods to process transcriptions generated using ASR models. A substantial advantage is that both ASR and NLU modules are independent and can easily be optimized on their own, with their own set of data. However, this also has three drawbacks. First, errors can accumulate as information passes through each stage, a phenomenon called **error cascading**. Second, a lot of acoustic information cannot be captured in text, like pitch, or intonation, and ends up being discarded, although being potentially useful for the downstream task. Finally, the NLU model is oftentimes trained standalone on written text, as opposed to ASR transcripts. Since written language is almost always different from spoken language, this introduces significant language discrepancies between training and real-world evaluation.
- **End-to-End systems**: This approach aims to train a model that performs speech-to-semantics in one go, directly optimizing for the downstream task, rather than decomposing the task in different subtasks [85, 59]. Although this can use acoustic and linguistic information more efficiently, avoids error cascading and solves the spoken-written language discrepancies, it requires more complex models and larger amounts of training data. For these reasons, end-to-end approaches trained only for SLU tasks struggle to match the performance of pipeline systems. Indeed, these specific datasets are considerably smaller and neural networks notably require large amounts of data to perform well.

In this thesis, we focus on four specific tasks: sentiment classification, spoken named entity recognition, intent recognition and slot filling.

Sentiment Classification

Sentiment Classification (SC) is the identification of the overall sentiment in an utterance into categories like "positive", "neutral", or "negative". This task is particularly useful to analyze product reviews videos, or as an intermediate step to discourse analysis. SC is typically an excellent example where speech cues such as intonation, pitch, and other information that cannot be expressed in pure text, are essential to solve the task. Indeed, to detect some expressions of language like sarcasm and irony, having access to the acoustic content might be fundamental. The metric used to evaluate sentiment classification is the macro-averaged F1-score.

Spoken NER

This task is the auditory counterpart of text-based NER, extracting entities directly from speech recordings. It encompasses not only the identification and classification of named entities but also the transcription of the spoken words, introducing additional complexities. Variations in pronunciation, accents, and dialects can significantly impact both the accurate detection of entities and their correct spelling. However, spoken language provides unique cues, such as prosody, intonation, and emphasis, which can enhance the model's understanding but are absent in written text [66]. We evaluate this task using the unordered micro-averaged F1-score and the label-F1. The former takes into account both the tag and the entity transcription, while the latter only takes the tag into account. The spread between both scores gives an indication of the ability of the model to spell the entities correctly.

Intent Recognition

Intent recognition (IR) is the task that, given a spoken command, tries to identify the overall goal behind the speaker's statement [64]. For example, the sentence "turn the lights on in the kitchen" would result in the following intent:

```
{"action": "activate", "object": "lights", "location": "kitchen"}
```

As we can see from this example, this task sometimes also includes the identification of simple arguments to complete the user's request. To evaluate intent recognition, we use the accuracy, where an example is considered as correct when the intent and all the arguments are predicted correctly.

Slot Filling

The slot filling task (SF) can be seen as an extension to IR. The goal is to identify the intent together with any number of arguments (slots) relevant to answer a query. In the context of airplane booking, these might be the Departure Airport, Destination Airport, Date and Time. Although IR also sometimes involves arguments, the difference rests in the number of possible values that each argument can take. Typically, for SF, this

number is large, sometimes unlimited, like in the context of smart speaker control, where it would be inconvenient to keep a list of all the possible artists and song titles that can be played. Although similar in nature to Spoken NER, SF is limited to a specific application such as airlines, and consists in short commands, whereas Spoken NER covers much broader and unrestricted domains. We use the same evaluation metric as intent recognition to evaluate slot filling.

Low Resource SLU

A core challenge in SLU is data scarcity. Annotated speech data is expensive and time-consuming to produce. Consequently, models able to learn from few examples are particularly attractive, especially since they need fewer computational resources and can be trained in no-time. This presents considerable advantages from both an economical and environmental perspective. Another aspect in which such models might offer a competitive advantage is in commercial applications requiring some degree of adaptation to individual users. As a manufacturer, proposing a model that works out-of-the-box is often necessary. This is where low-resource techniques come into play. These techniques focus on achieving good performance with limited data, making SLU systems more practical and accessible for real-world applications.

Architectures for SLU

For Spoken NER, Peng et al. [72] takes a sequence-to-sequence approach, where special symbols are issued to mark the beginning and the end of an entity phrase. They use a Conformer [33] encoder and a Transformer decoder [99]. Mdhaffar et al. [62] proposed to replace the last layers of a pretrained ASR model with a text-based NER model. Unfortunately, they use a non freely available dataset and did not release the source code, making any comparison difficult.

For IR, Lugosch et al. [59] propose an end-to-end model in which three modules are stacked together: a phoneme module, a word module and an intent module. Each module can be pretrained independently by following an unfreezing schedule. The IR task is reformulated as a multilabel classification task. Haghani et al. [34] define the task as a sequence-to-sequence problem with an encoder-decoder architecture and introduces a multitask learning approach where the model learns both the transcript and the semantics. Unlike our work presented in (Chapter 5), they opt for sharing the encoder and use a separate decoder for each task. Finally, multi-modal approaches that use both text and speech encoders have been proposed. ST-BERT [47] enhance their model by pretraining it with cross-modal language modelling on both speech and text data using two different approaches. In the first approach, the objective is to predict one modality based on the other. In the second approach, they combined both modalities in one sequence and mask parts of it. The training objective is then to predict the masked parts given the rest. To enhance the pretraining process, ST-BERT uses large text corpora and domain adaptation pretraining with SLU

transcripts. J. Dong et al. [24] implement cross-modal contrastive learning to learn multi-modal representations from text and speech in parallel. They use a Conformer speech encoder [33] and BERT text encoder [23] and use contrastive learning to align the resulting embeddings in the same latent space. Multitask learning (Chapter 4) and multimodal approaches [24, 47] allow the model to learn shared representations from both modalities, which can be useful in a low resource scenario. In almost all mentioned research, pretrained models are shown to improve the performance considerably [3, 24, 47, 52, 72].

2.5 Datasets

In this section, we describe the datasets that will be used in the following chapters of the thesis.

2.5.1 Datasets for Automatic Speech Recognition

Librispeech [70] contains about a thousand hours of read English speech derived from audiobooks. It is the most well-known ASR dataset and is a reference for training and evaluating ASR systems. The authors provide official splits for training, validating and testing the models and this dataset serves as reference in ASR. We use all 960 hours for training.

Corpus Gesproken Nederlands (CGN) [69] is a collection of recordings in Dutch and Flemish collected from various sources such as readings, lectures, news reports, conferences, telephone conversations, etc. totaling more than 900 hours. They are divided into 15 components (a to o) based on their nature. After removing short and overlapping utterances, we divide each component into three subsets that will serve as training, validation and test sets. We leave out three components (a, c and d) because the quality of the transcriptions differ considerably from the other ones. The subsets from the remaining components, totaling 415 hours of speech, are concatenated together to form the training, validation and test sets.

2.5.2 Datasets for Spoken Language Understanding

Sentiment Classification

SLUE-VoxCeleb [86] is a benchmark dataset for sentiment classification from speech. The task is to classify each utterance as being positive, negative or neutral.

Intent Recognition and Slot Filling

Grabo [80] is composed of spoken commands, mostly in Dutch (one speaker speaks in English) intended to control a robot. There are 36 commands that were repeated 15 times by each of the 11 speakers. More precisely, the robot can perform eight actions (i.e., approach, move (relative), move (absolute), turn (relative), turn (absolute), grab, point, lift), each being further defined by some attributes (e.g., the robot can move forward or backward, rapidly or slowly). We follow the methodology applied in [80]: For each speaker, we create 9 datasets of different sizes and divide each dataset in 5 folds for cross-validation.

Patience [92] is derived from a card game where the player is giving vocal instructions to move cards between different tiles on the table. The dataset contains recordings from 8 different Flemish speakers. The intent of the player is represented as a binary vector. Again, we use the same splitting methodology as Renkens and Van Hamme [80], keeping only the 31 most represented classes. It should be noted that without visual input, this task is quite difficult even for a human operator.

Fluent Speech Commands [59] contains 30,043 spoken commands from 97 English speakers to control smart-home appliances or virtual assistants. Each intent can have up to 2 arguments, with a total of 31 commands. In its original form, the train, validation a test splits have respectively 23,000, 3,000 and 3,800 examples, totaling 14h40, 2h and 2h35 respectively.

FSC Challenge [5, 59]: The challenge splits of Fluent Speech Challenge, proposed in Arora et al. [5], contain 19,262 training examples (12h), 2,597 validation examples (1h40), 3,349 test examples with unseen speakers (2h12) and 4,204 examples with unknown utterances (2h40).

Smartlights [83] contains instructions to control smart lights. They can be turned on and off, the brightness can be set, or the light color changed. Slots include the rooms of the house, color and brightness (a number from 0 to 100). In its original form, the dataset (1,660 examples, 1h13) does not contain splits. Following Agrawal et al. [1], we use random splits (90/10/10) in our experiments.

SL Challenge [5, 83]: New splits of the Smartlights dataset have been proposed in Arora et al. [5], with two challenging test sets. One test set (referred to as Spk.) is composed of recordings from unseen speakers and the other (referred to as Utt.) contains utterances for which the wording was not observed for a particular target. The dataset has 1,251 training examples (1h), 156 validation examples (7 min), and each test set has 126 examples (6min).

Spoken Named Entity Recognition

SLUE-VoxPopuli [86] is a subset of the VoxPopuli dataset [102] from the European Parliament sessions, together with named entities. The SLUE benchmark [86] provides limited training (5,000 examples, 14 h) and development sets (1,753 examples, 5h)

and a blind test set (1,842 examples, 5h). They primarily use a derived version of the OntoNotes label set that combines the original 18 classes to a smaller 7 classes label set, merging some categories together and removing the rarest ones. Refer to Chapter 6 for a detailed table of the classes (Table 6.2 on page 69).

Chapter 3

Bidirectional Representations for Low Resource Spoken Language Understanding

3.1 Introduction

There is a strong need for lightweight robust personal assistants able to understand the nuances and intricacies of spoken language, and it is essential that those systems work out-of-the-box, without requiring much effort from the user. To do this, the underlying models need to be able to generate representations of speech that capture essential information such as meaning and intent, and store it efficiently so that it is retrieved easily, regardless of background noise or speech variations.

In this chapter, we address the question raised in Section 1.1.1 (page 2), and attempt to build expressive representations of speech that can be used in low-resource SLU scenarios, when only a handful of examples are available to train and validate the models. We propose an architecture inspired from Higuchi et al. [39] for pretraining bidirectional speech transformers on a surrogate ASR task with the goal of learning qualitative representations that prove useful for solving language understanding tasks. The masked language modeling objective is used instead of the more common

This chapter was originally published as a journal paper with the following reference: Q. Meeus, M.-F. Moens, and H. Van hamme. "Bidirectional Representations for Low-Resource Spoken Language Understanding". In: *Applied Sciences* 13.20 (2023). DOI: 10.3390/app132011291. arXiv: 2211.14320.

autoregressive language model (Section 2.3.2), to ensure representations leveraging both left and right contexts. To generate transcription, we build a template of the same length as the greedy CTC output computed from the encoder output. This template is completely masked and iteratively filled and refined using the bidirectional decoder. Contrarily to Higuchi et al. [39], where the focus is on building fast ASR models, we exploit the learned representations for downstream SLU tasks. Non-autoregressive decoding has been applied to speech recognition in recent papers. In one line of research [15, 16], the CTC output is iteratively refined before collapse, therefore processing sequences that are much longer than our proposed model. Other approaches look at how to integrate acoustic models such as Wav2Vec2 and language models like BERT together [7, 47].

Additionally, since the original publication of this chapter as a journal article, considerable progress has been made in non-autoregressive speech processing, fueled for the most part by its promise for faster decoding speeds [27, 38, 88, 113]. Various architectural improvements have been proposed: replacing the Transformer encoder with a Conformer [38] and with a pretrained Wav2Vec2 model [22], replacing the decoder with BERT pretrained model [37, 40], with a Transducer and with contextual block streaming ASR [113]. One key line of research is the use of non-autoregressive decoder for streaming ASR [27, 88, 113]. In Someki et al. [88], beam search decoding is used only on low-confidence CTC segments to minimize the expensive decoder computations. Although these studies focus on ASR and do not consider the model's representations for other tasks, the representations obtained with these methods are likely to improve what we propose in this chapter [54].

In this work, we pretrain our representation model from scratch and leave large pretrained models to the following chapters. The representation model is evaluated offline by generating representations and using them to train downstream models for a SLU task, intent recognition. Additionally, since the original publication of this chapter as a journal article, considerable progress has been made in non-autoregressive speech processing, fueled for the most part by its promise for faster decoding speeds [27, 113, 88, 38]. Various architectural improvements have been proposed: replacing the Transformer encoder with a Conformer [38] and with a pretrained Wav2Vec2 model [22], replacing the decoder with BERT pretrained model [40, 37], with a Transducer and with contextual block streaming ASR [113]. One key line of research is the use of non-autoregressive decoder for streaming ASR [113, 88, 27]. In Someki et al. [88], beam search decoding is used only on low-confidence CTC segments to minimize the expensive decoder computations. Although these studies focus on ASR and do not consider the model's representations for other tasks, the representations obtained with these methods are likely to improve what we propose in this chapter [54].

In this work, we pretrain our representation model from scratch and leave large pretrained models to the following chapters. The representation model is evaluated offline by generating representations and using them to train downstream models for a SLU task, intent recognition. We will be particularly interested in low resource scenarios, where we control the number of examples used for training and validation to study the limits of what can be learned from the representations. When information

is stored efficiently, few examples are necessary to learn how to extract it. In contrast, noisy embeddings require many examples to extract relevant information. For the SLU module, we propose class attention as a drop-in replacement of LSTMs. Class attention assigns to each position a score that relates to the importance of the corresponding token to predict the intent and arguments. Aside from the advantages brought by its small footprint, it gives us a method to understand why a model makes predictions. In the rest of the chapter, we first detail the architecture of the different components of the model. Then, we lay out the experimental setup and methodology. Finally, we evaluate the models and analyze the representations in details.

3.2 Methods

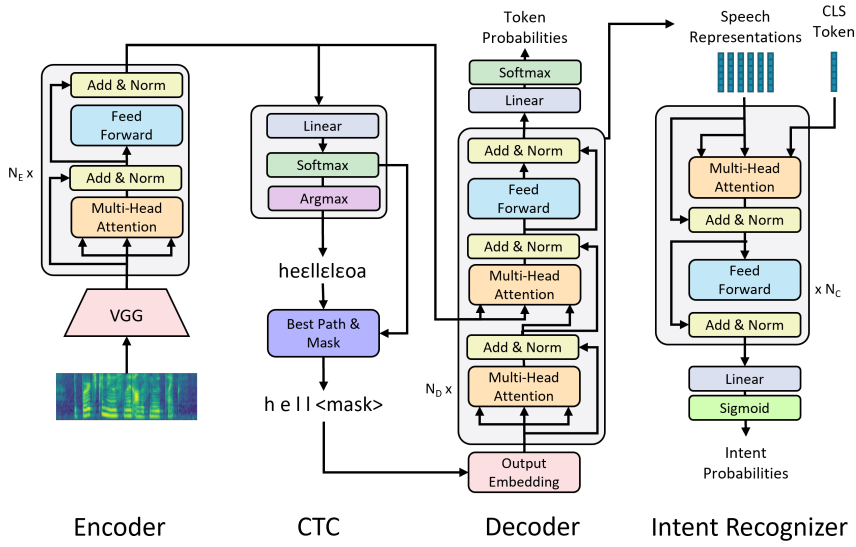


Figure 3.1: The encoder processes speech features. The CTC module makes a rough prediction of the text, where low-probability tokens are masked. The decoder attends to the masked transcription and to the encoder's output and predicts missing symbols. The generated representations are used as input to the downstream model to predict the intent.

Building on Higuchi et al. [39], we propose an encoder-decoder architecture with a multi-objective to learn bidirectional representations of speech (Figure 3.1). We evaluate the features learned in a downstream SLU task: intent prediction. The encoder, presented in Section 3.2.1, learns acoustic representations of speech. The sequence of acoustic unit representations are used in two modules in parallel: they are mapped to output symbols with a classification layer and optimized with CTC

(Section 3.2.2), and they are processed by a bidirectional transformer decoder to learn linguistic features (Section 3.2.3). These features learned by the decoder are processed by an intent recognizer to get the final output (Section 3.2.4).

3.2.1 Encoder

The encoder processes Mel-scaled filter banks and transforms them into a sequence of acoustic embeddings, h_{enc}. The module is composed of a convolutional frontend followed by a transformer encoder. The role of the CNN is two-fold: decrease the dimension along the frequency and time dimensions and learn the local relationships in the sequence of speech features. This replaces the positional encoding traditionally required by transformers to keep track of the sequence order [68]. The CNN frontend compresses a sequence of speech features of length T to a smaller sequence of length $T' \approx T/4$. The transformer encoder is composed of multiple blocks of multi-head self-attentions and position-wise fully-connected feedforward networks. Each block has a residual connection around both operations, followed by layer normalization. In this setting, the encoder cannot be trained standalone.

3.2.2 CTC

Connectionist Temporal Classification [32] is a method for optimizing sequence-to-sequence models. For each unit in the input sequence, CTC predicts a probability distribution over the vocabulary, consisting of the set of output symbols and a blank token. The predictions are assumed to be conditionally independent. A simple method to reduce a sequence of CTC tokens into a sentence consists in first removing repeated symbols, then removing blank tokens. The CTC loss is computed by summing the negative log-likelihood of all the alignments that give the correct sentence. For decoding, we opt for a greedy algorithm, which takes the token predicted with the highest probability at each time step. Although more advanced techniques such as beam search or weighted finite state transducers [111] can improve the CTC prediction considerably, Zenkel et al. [111] found that the largest gains were observed on substitution errors, with substantially lower gains on insertions and deletions. For this reason, we prefer a fast decoding technique and let the decoder fix the substitution errors in the rough prediction.

3.2.3 Decoder

Although the encoder equipped with CTC is theoretically able to transcribe speech, it does so by assigning a token to each position independently. In Watanabe et al. [106], the addition of a decoder integrates language modeling in the learning process. Indeed, the decoder combines the acoustic cues of the encoder with the previous tokens to predict the next one. Here, we opt for a bidirectional transformer decoder instead of a left-to-right model. Left-to-right decoders predict one token at a time

and stop after producing a special end-of-sequence token. This method is slower because a prediction depends on the previously predicted tokens. Additionally, the network only has access to previous tokens, thus limiting the amount of information available to make a prediction. In modern NLP, bidirectional architectures have become popular to learn powerful representations that make use of both left and right contexts [23, 73]. For instance, BERT [23] is optimized with a masked language modelling objective by randomly masking some tokens in the input sequence. In ASR, a similar approach was introduced by Higuchi et al. [39], where a rough text prediction obtained with CTC is refined by masking low-probability tokens and letting the decoder predict the missing items in the sequence. During inference, the CTC module is used to initialize a partially filled sequence that is completed by the decoder. This method underperforms on ASR because the decoder is only able to handle substitution errors in the initial CTC template, and not insertions and deletions. Ghazvininejad et al. [30] proposed to predict the length as part of the prediction task of the encoder. During inference, the decoder predicts multiple candidates of different lengths corresponding to the lengths predicted with the largest probability. However, this requires considerably more processing because the decoder must iteratively predict a sentence for all candidates. In this work, we focus on learning rich representations for spoken language understanding. We work around this issue by assuming that the length is not essential for our purpose. We follow Higuchi et al. [39] for the architecture of the decoder, but we adapt the training strategy. In addition to the masked tokens, we randomly swap unmasked tokens with incorrect tokens to simulate errors in the initial template. The output of the penultimate layer is used as the linguistic representations of speech.

3.2.4 Intent Recognizer

In the SLU datasets, intent is represented as a certain action (e.g. move, grab, turn, etc.) to which are attached a number of arguments (forward, fast, left, etc.). We encode the full intent string as a multi-hot vector, where each bit corresponds to either one of the possible actions or an argument value. The dimension of the vectors is thus given by the sum of the number of values that each argument can take. The representations obtained from the decoder are summarized with a stack of class attention layers [94]. Class attention combines a sequence of representations into a class embedding, x_{CLS}, that is, a learned vector of the same dimensions as the representations. During training, the module learns to identify patterns in the input features that are predictive of the class labels. Similarly to other transformers, a class attention layer is composed of two sub-layers: an attention layer and a point-wise feed-forward layer. The input to the layer is normalized before processing to stabilize training, as prescribed in Xiong et al. [109]. We define the queries Q in Equation 2.11 (page 14) as the class embedding. The keys K and values V from Equation 2.11 are

linear projections of the elements in the input sequence of features X:

$$\begin{cases} Q = x_{\text{CLS}} \\ K = W_k \cdot X \\ V = W_v \cdot X \end{cases} \tag{3.1}$$

Contrarily to Touvron et al. [94], we do not include the CLS token in the keys and values. We compute the attention weights by applying the softmax function to the inner product of the keys and queries, scaled by the square root of the dimension of each attention head:

$$A = \sigma(Q \cdot K^T / \sqrt{d/h}) \tag{3.2}$$

where d is the total dimension of x_{CLS} and h is the number of attention heads. The output corresponds to

$$y = W_o AV + b_o \tag{3.3}$$

where W_k, W_v, W_o and b_o are learned parameters. We train the model by minimizing a balanced binary cross-entropy loss (2.4). We found that a weight equal to the inverse class frequency measured on the training set was optimal for our dataset. Additionally, since not all output sequences are valid, we enforce the output structure with a simple algorithm that finds the sequence from the set of legal sequences that minimizes the cross-entropy with the predicted vector. If $\mathbb{Y}$ is the set of legal outputs, p the model's output, the decoded prediction, y^*, is given by:

$$y^* = \underset{y \in \mathbb{Y}}{\text{argmax}}\, H(y, p) \tag{3.4}$$

3.3 Experiments

3.3.1 Pretraining

We pretrain our encoder-decoder model on an ASR task with a hybrid objective that is the weighted sum of the CTC loss, L_{ctc}, and the label smoothing cross-entropy loss, L_{dec}:

$$L = \rho \cdot L_{\text{ctc}} + (1 - \rho) \cdot L_{\text{dec}} \tag{3.5}$$

We use CGN (220h) to pretrain the Dutch model and Librispeech (960h) for the English model. Our objective is to train a representation model of speech that captures elements of language and semantics while remaining robust to irrelevant aspects such as speaker identity or background noise. Additionally, the information stored in the representations should be readily accessible for processing by a downstream model. We choose ASR as a proxy task because of the availability of large datasets and because it is easier to generate rich representations containing elements of language than with self-supervised approaches, which will typically require much more resources to achieve the same goal. Yet, the supervised objective is likely to throw away information not directly useful for solving the task. Solving this neglect will be the focus of future work. During pretraining, we partially mask tokens in the target sequence, and we randomly swap some unmasked tokens to simulate errors in the initial sequence.

3.3.2 Generating Speech Embeddings

Once the model is trained, speech embeddings can be generated. The CTC module predicts a rough prediction of the final transcription. We use Algorithm 1 to mask the tokens with a probability inferior to 90%. The *collapse* and *filterblanks* functions are similar to Figure 2.4c. The probabilities are aggregated by taking the average value for each subsequence of consecutive tokens. We have found that this method slightly balances the overconfidence of the CTC module. We iteratively refine the input for a maximum of 10 steps.

Algorithm 1 Best Path & Mask

Require: tokens Y, probabilities P, threshold π^*
  $Y, P \leftarrow \text{collapse}(Y, P)$ ▷ Remove repeated symbols
  $Y, P \leftarrow \text{filterblanks}(Y, P)$ ▷ Remove blank symbols
Ensure: $Y.\text{length} = P.\text{length}$
  $i \leftarrow 0$
  while $i < Y.\text{length}$ **do**
    if $P[i] < \pi^*$ **then**
      $Y[i] \leftarrow$ <mask>
    end if
    $i \leftarrow i + 1$
  end while
  return Y

3.3.3 Training

We use the pretrained model to generate embeddings of the downstream datasets. This is equivalent to freezing both the encoder and the decoder of the pretraining model and attaching the SLU module on top, but saves us from the extra-processing. This method gives us the opportunity to evaluate the predictive power of the representations. Indeed, the pretrained encoder-decoder from Section 3.3.1 has not been exposed to any training example from the SLU datasets, nor does it know about the output structure of the underlying task (number of classes, etc.). During the forward pass, the embeddings sequences are propagated in the network, and we compute the binary cross-entropy between the output and the multi-hot encoded targets.

3.3.4 Fine-tuning

Our main objective with this research is to learn representations that perform well without fine-tuning. However, to provide a good basis for comparison with previous research, and to quantify how much can be gained by updating the main model, we perform a few fine-tuning steps at a low learning rate while unfreezing all or some layers of the representation model. This operation can be seen as specializing the representation model for the specific aspects of the downstream task, often at the

expense of generality. During fine-tuning, the decoder needs to receive an input. As discussed in Section 3.2.3, we use the CTC module to produce the initial sequence where low-probability tokens are masked. We do only one pass with the decoder to generate the representations used by the SLU module.

3.3.5 Hyperparameters

The transformer has 18 layers (12 encoder layers and 6 decoder layers) each with 4 attention heads, a hidden size of 256 and 2048 hidden units in the linear layer. We use dropout with a probability of 0.1 after each transformer layer. The frontend has two 2D convolutional layers with a kernel size of 3 by 3 and a stride of 2 by 2, the input dimension is thus divided by a factor of 4 along the time and frequency dimensions. The hyperparameters related to the architecture of the encoder-decoder were chosen following Higuchi et al. [39]. The encoder-decoder model has 30.9 million parameters. The weights for scaling the different losses in the final loss function were chosen by experimentation on the pretraining data. We pretrain our models for 200 epochs with a batch size of 256. The models are trained with the Adam optimizer [48]. The learning rate linearly increases until reaching a maximum value of 0.4 after 25,000 steps, then decreases according to the Noam schedule [99]. We experimentally set ρ in (3.5) to 0.3 by measuring the ASR accuracy on a held-out validation set. The downstream datasets were not used in any way to determine the value of the above hyperparameters.

The downstream models are composed of two class attention layers with 4 attention heads of dimension 32. The fully connected layer has 1024 units. The models are trained with batches of 512 examples for 100 epochs, with a learning rate equal to 0.005 except for the Smartlights dataset, for which we found that a slower learning rate of 0.001 gave better results. The intent classification module has 890 thousand parameters. We train the models with the Adam optimizer [48] for a maximum of 200 epochs with early stopping.

3.4 Results

We compare our models on basis of the accuracy score to remain consistent with previous research. The two baselines considered are end-to-end SLU [59] and ST-BERT [47]. The former is a model composed of a phoneme module, a word module and an intent module that can all be trained or fine-tuned independently. They have experimented with four settings: with or without pretraining, and fine-tuning the word module only or all modules together. The two best models (pretrained on ASR and fine-tune word module only) are displayed in Table 3.1 and Table 3.2. ST-BERT [47] also uses an MLM objective. However, they pretrain their model with cross-modal language modelling on speech and text data in two different ways: using MLM or with CLM where the goal is to predict one modality given the other. Additionally, they

Table 3.1: Test accuracies on Fluent Speech Commands.

Stage	% train	MLM (Ours)	E2E SLU [59]	ST-BERT [47]
train	100	99.45	98.80	99.39
	10	99.00	97.96	99.04
	1	91.48	82.78	89.81
fine-tune	100	99.79	99.10	99.50
	10	99.52	97.90	99.13
	1	95.70	-	95.64

use large text corpora for pretraining and use domain adaptation pretraining with the SLU transcripts to further improve their results. However, using text transcripts from the downstream task is not compatible with our use case as we assume that only the speech is available for the SLU task. Since domain adaptation gives an unfair advantage to their model, we did not include here the results related to this experiment. The pretrained model corresponds thus to the model that was pretrained on speech-only data, and the fine-tuned model corresponds to the model pretrained on text with CLM and fine-tuned on SLU. We show results on Fluent Speech Commands

Table 3.2: Test accuracies on FSC Challenge, SmartLights and SL Challenge.

Model	Stage	FSC Challenge		SmartLights	SL Challenge	
		Spk.	Utt.	Random	Spk.	Utt.
MLM (Ours)	pretrained ASR	95.19	86.30	84.03	79.53	69.05
	fine-tune decoder	98.80	87.99	85.84	81.89	74.60
E2E SLU [59]	pretrained ASR	90.90	73.40	83.20	73.20	67.40
	fine-tune word layer	92.30	78.30	88.00	82.60	78.50
ST-BERT [47]	pretrained ASR	-	-	81.22	-	-
	pretrain text+fine-tune	-	-	84.65	-	-

and Smartlights both with the old splits and new challenge splits [5]. For Smartlights, no previous splits were available, so we follow the same approach as in Arora et al. [5] and use a random splitting strategy. In Table 3.1, we observe significant improvements compared to Lugosch et al. [59], especially when few training examples are used, both before and after fine-tuning. Lugosch et al. [59] noted that unfreezing the whole model for fine-tuning does not always improve performance due to catastrophic forgetting. We make here the same observation and find that unfreezing the encoder leads to its degradation. The encoder was built to be robust to variations in the input features. It was pretrained on large amounts of data and when it is fine-tuned with a different objective on a handful of examples, the parameter updates following the new setting inevitably lead to overfitting and loss of generality. Additionally, updating the encoder also deteriorates its ability to generate a good template for the decoder, which leads to poorer performance. Consequently, we decided to freeze the encoder but update the decoder during the fine-tuning stage. With this setting, the performance improves

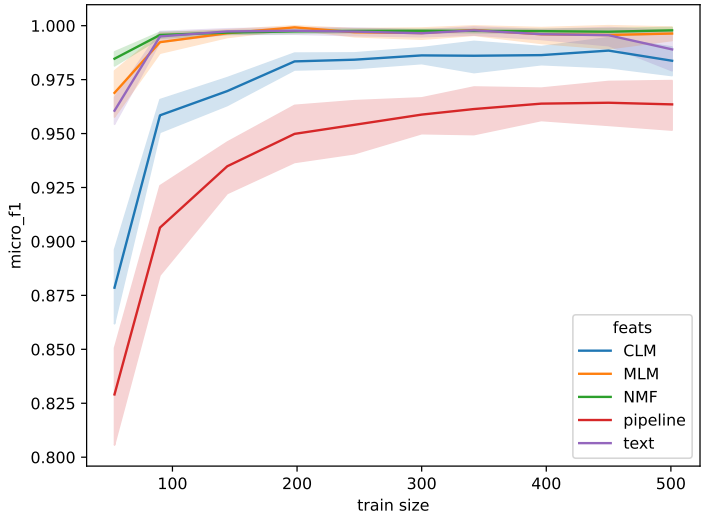
slightly, although consistently compared to the model before fine-tuning. Our model shows a similar pretraining accuracy as Kim et al. [47], but small improvements are observed after fine-tuning. In Table 3.2, the challenge splits [5] correspond to improved splits where specific speakers or specific utterances have been partitioned. For Fluent Speech Commands, we show considerably better performance, in particular for unknown utterances where we improve over Lugosch et al. [59] by 17 % and 12 % relative before and after fine-tuning. However, our model does not improve on Lugosch et al. [59] on the SmartLights dataset, although we show better results than Kim et al. [47]. This dataset's target differentiates between arguments that belong to the same semantic class (e.g. rooms or colors). The few examples in the training set do not suffice to generalize to unknown phrasings for a specific target (i.e. when the same target is expressed differently).

3.5 Analysis

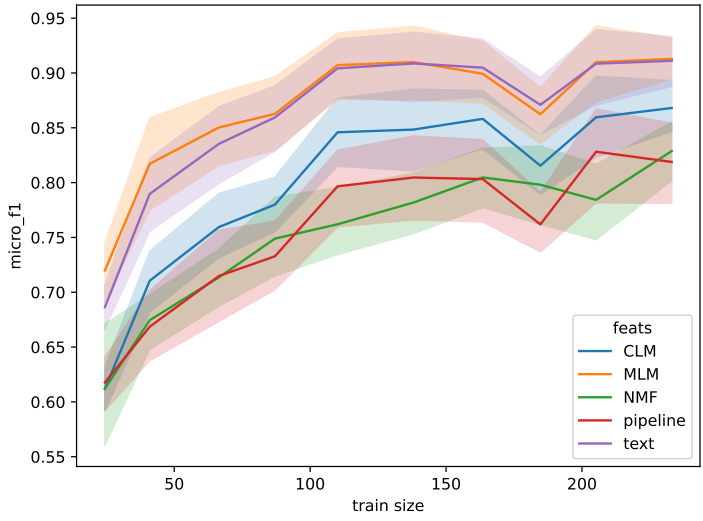
3.5.1 Low Resource Scenario

To explore low-resource scenarios, we split Grabo and Patience per speaker. For each speaker, we randomly select a fixed number of examples per class for the training set of the SLU model. By gradually increasing the training set size, we are able to measure the learning curve. For each speaker, this operation is performed three times, with different splits each time. We compute the micro-averaged F1 score for each experiment and report the average F1 score with its standard deviation in Figure 3.2. We compare our representations (*MLM* in Figure 3.2) with three types of features: *NLP* features are generated by encoding the gold transcriptions with *bert-base-dutch-cased* [101], *pipeline* features result from encoding ASR transcripts predicted by ESPnet's hybrid ASR model [106] trained on CGN with *bert-base-dutch-cased* and *CLM* features correspond to the output of the penultimate layer of the ASR model. To provide a fair comparison, neither the NLP, ASR nor MLM models have encountered training examples from the SLU dataset. In other words, we do not fine-tune any of the representation models on the downstream datasets at this stage. Both the ASR and MLM models are trained on CGN, and the NLP model was trained on a collection of five text corpora for a total of about 2.4B tokens.

Considering the amount of training data, it is no surprise to see that the gold transcript encoded with the NLP model performs very well in all data regimes. As expected, the ASR transcripts show the weakest performance. Converting features into discrete symbols forces the model to make decisions and potential errors that cannot be recovered from. By contrast, the CLM and MLM models use the hidden representations produced by the model, thus avoiding this problem. Nonetheless, we see that the CLM features perform worse than our model. The masked language modelling objective allows the MLM model to look at both right and left contexts, where the CLM model only has access to previous predictions. Because each predicted unit is conditioned on the whole sentence, the model is able to derive better representations than by only looking at the past.



(a) Grabo



(b) Patience

Figure 3.2: Comparison of different models trained with increasing train set sizes on Grabo (left) and Patience (right). Each curve represents the F1 score on the test set as a function of the size of the training set (utterance count). The colored areas represent a 68% confidence interval. We use the F1 score to compare with the NMF baseline [104].

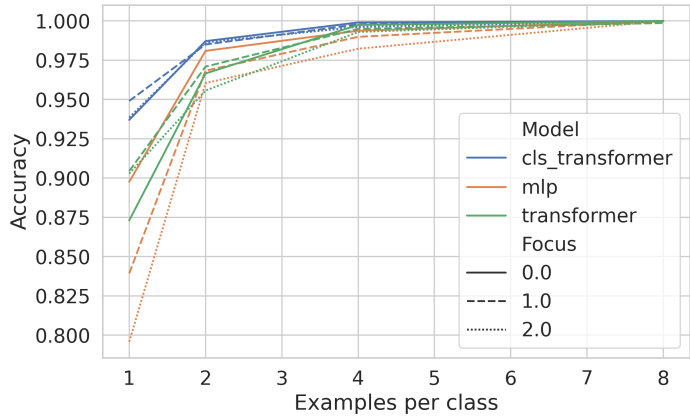


Figure 3.3: Comparison of Class Transformer, Transformer encoder and Multilayer Perceptron for different focus values.

We also compare our results with the state-of-the-art on both datasets [104]. This model combines ASR pretraining and an NMF decoder for intent recognition [104, 103]. The NMF decoder uses a bag-of-words approach with multi-hot intent representation, thus ignoring the order of the words in the sentence. Although this model was presented for dysarthric speech, results are available for both Grabo and Patience datasets [104]. The main advantage of the NMF decoder is its low computational requirements, which makes it particularly well-suited when very few training data are available, as can be observed on Figure 3.2a. This advantage disappears as the size of the training set increases. For Patience corpus however, where the order of words is necessary to make a prediction, our approach shows a better performance (Figure 3.2b).

3.5.2 Class Attention Transformer and Focal Loss

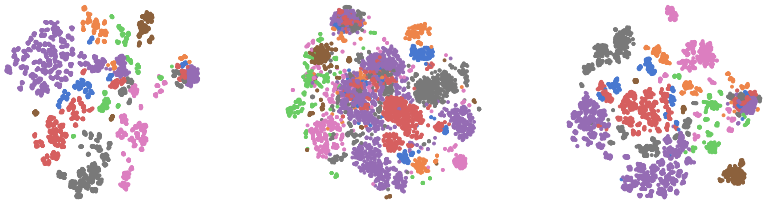
This experiment attempts to evaluate the advantages of class attention in low resource scenarios. We also study which loss is more appropriate for our training task and data (Section 2.1.2). We compare three models, the Class Transformer, presented in Section 3.2.4, a Transformer encoder (Section 2.2.4), and a feedforward network (MLP in Figure 3.5.2). All three models were chosen to have approximately the same number of parameters per layer. We compare these model architectures trained with focal loss (Equation 2.6 on page 10) for various values of f and α.

We performed a hyperparameter search with training sets containing utterances from one speaker and 1, 2, 4 or 8 examples per class. The search parameter space included for each architecture the number of layers (1 or 2), the batch size (powers of 2 from 4

to 128), dropout (0.1, 0.3, 0.5), learning rate $(10^{-4}, 10^{-3}, 10^{-2})$, and parameters for the loss α (0.25, 0.5, 0.75, 1) and f (0, 1, 2). Figure 3.3 shows the best configuration found for each model, train size and focus. Due to the poor results of the unbalanced binary cross-entropy, models using this configuration were not displayed in the graph. The results are cross-validated accuracy score, averaged over multiple speakers and training set samples, following the same methodology as in the previous experiment (Section 3.5.1). From the grid search, we found that $\alpha = 0.75$ was almost always the better choice (91.6% of the cases), smaller batch size was more appropriate in lower resource scenarios, and the best MLPs generally preferred lower dropout and fewer layers than the transformers.

In Figure 3.3, we see that the class transformer stands ahead in every low resource scenario, and for all values of focus f, although for larger training sizes, all the models seem to perform equally well. The focus seems to have a positive effect in low resource scenarios, but this advantage disappears as more training examples become available. Our results show that $f = 1$ is the most appropriate value of the three for all three architectures in the setting with the least number of examples. This suggests that in low resource scenarios, strategies that tend to put more importance on fixing incorrect predictions pay more than weighing all examples equally.

3.5.3 Content Representation



(a) Encoder **(b)** Decoder.2 **(c)** Decoder.5

Figure 3.4: t-SNE representation of the representations at different layers. Intents: **approach** (forward/backward), **grab** (open/close grabber), **lift** (lift object up/down), **move_abs** (move in absolute direction), **move_rel** (move in relative directions), **pointer** (laser on/off), **turn_abs** (turn to absolute position), **turn_rel** (turn to relative position). Best viewed in color.

To better understand the characteristics of the representations, we average the sequences to a unique representation per utterance and visualize them in two dimensions using the t-SNE algorithm [60]. For this experiment, we focus on Grabo, and in particular, on the eight actions that the robot can do. We compare the representations produced by different layers, namely the last encoder layer, the third

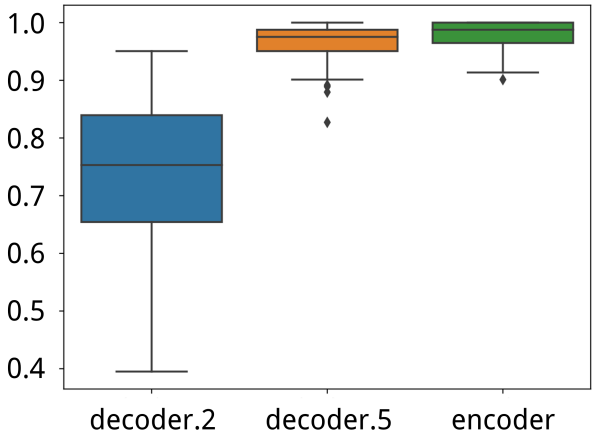


Figure 3.5: SLU accuracy of the models trained with different layer representations. Best viewed in color.

decoder layer and the last decoder layer (resp. encoder, decoder.2 and decoder.5 in Figure 3.4). Both encoder and decoder.5 form meaningful clusters, while decoder.2 is less defined. This is also observed in the classification accuracy of those representations on SLU where we observe a clear difference between decoder.2 and the other features. We expected that decoder.5 would generate better features than encoder, although they seem to lead to similar performance with a small advantage for the encoder's output. We conjecture that the CTC component helps to define the acoustic units at the encoder's output, which translates into well-defined representation sequences, albeit much longer ones.
We also observe that *approach* and *move_abs* end up in the same region, which means that a classifier might often confuse them. This makes sense from a language perspective, and the fact that this is more the case in decoder.5 than in encoder leads us to the hypothesis that the implicit language model from the decoder generates similar embeddings for these two commands, although they do not sound similar.

3.5.4 Explain the Predictions

Finally, we wanted to explore an aspect that is often overlooked in deep learning, namely trying to explain what elements in the input features lead to the model making a certain prediction. One key aspect of our class attention layer is that it relates the final output prediction with the input sequence by learning attention weights. Those can be seen as the probability that each input unit is relevant for assigning the right label to the utterance. An example is displayed in Figure 3.6, where we observe that a limited number of tokens are responsible for making the prediction (turn, the, light,

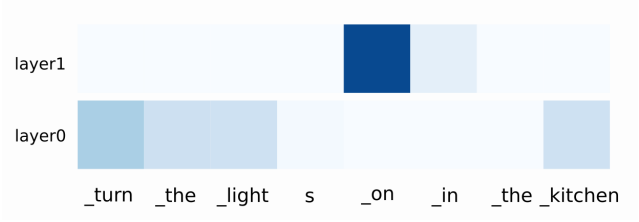


Figure 3.6: Class Attention weight visualization of the two layers of the downstream class attention model. Although the model receives sequences of embeddings, we labeled the graph with the corresponding tokens for demonstration purposes. The two layers focus on different positions to predict the intent and arguments.

on, kitchen). Would the model make an incorrect assumption, these weights can serve as a basis to investigate why.

3.6 Conclusion

SLU research is increasingly moving from pipelines towards end-to-end approaches, but often struggles to match their counterparts' performance. In this chapter, we proposed a bidirectional representation model pretrained on an ASR task and showed that the learned features transfer very well to intent recognition even without fine-tuning. During pretraining, we used CTC in conjunction with an MLM objective. We compared the representations obtained at different layers of the model and in particular how a new, unrelated dataset was encoded. We showed that the representations could be used to train a SLU model successfully. The representations performed as good as or better than state-of-the-art models on the SLU task, especially before fine-tuning. We proposed to use class attention to summarize the representation sequences. This method provides two main advantages: it is efficient due to the transformer-like architecture and the small number of parameters, and it generates attention weights that are helpful to understand a model's inner workings and identify which patterns in the input influence the prediction. The combination of class attention and ASR pretraining leads to considerable savings in the amount of data that is necessary for reaching good performance, which we demonstrated in a data shortage scenario by arbitrarily limiting the number of training examples for the SLU task. As models become bigger and data-hungrier, deep learning research should head towards systems that are more resource-friendly and more efficient. We have also validated experimentally the class transformers applied to low resource SLU, compared to other similar models, and, although we used balanced cross-entropy loss to train the models presented in this chapter and the original paper, we have also shown that the focal loss has real potential in low resource SLU, as we will see further in Chapter 5 onwards.

One limitation of the presented model is when the SLU dataset makes distinctions

between terms that are not or rarely observed in the pretraining dataset. For example, let us assume that two words are used interchangeably in all but a few examples in the pretraining set. Their representations will be very similar, if not indistinguishable. Without fine-tuning, our proposed model will likely not be able to discriminate between those two classes. In the next chapter, we will explore how we can improve the representation model with fine-tuning in low resource scenarios.

Chapter 4

Multitask Learning for Low Resource Spoken Language Understanding

4.1 Introduction

In this chapter, we address the question introduced in Section 1.1.2 (page 3). This continues our research for building good speech representations for low resource SLU. As we have seen, SLU datasets are notoriously small. In the previous chapter, we have trained a representation model and evaluated the learned representations offline. However, we have seen that representing spoken words with similar meaning, this model fell short. To overcome this limitation while keeping reasonable training data requirements, we look closer at multitask learning, a method that combines multiple objectives in a unique loss function (2.1.4 on page 12).

In this work, we investigate synergies between spoken language understanding, and in particular intent recognition and sentiment classification, and speech recognition. Having a system able to transcribe speech to text is likely to be helpful for such tasks and requires fewer data and fewer parameters to reach a good performance [34, 77].

This chapter was originally published as a conference paper with the following reference: Q. Meeus, M.-F. Moens, and H. Van Hamme. "Multitask Learning for Low Resource Spoken Language Understanding". In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. 2022, pp. 4073–4077. DOI: 10.21437/Interspeech.2022-11401. arXiv: 2211.13703. It was presented on September 20th, 2022 in Incheon, South Korea.

These are two aspects that we will focus on in this work: We compare a multitask model optimized for both speech transcription and intent recognition or sentiment classification with a model pretrained on ASR then fine-tuned on SLU. We also look into edge cases in IR by simulating training sets with few examples. Further, we limit our model sizes considerably for the advantages in terms of speed and energy consumption that it brings. Experiments are performed in English and in Dutch.

4.2 Architecture

The model is composed of a generic transformer encoder on which we stack different modules, each specialized for one task. We experiment with different arrangements in order to find the optimal architecture that allows each task to benefit from - rather than compete with - each other. We have selected four tasks of interest: ASR, IR and SC (see Section 2.4.3 (page 27) for the task definitions). In this work, we explore the synergies between each task and ASR. Additionally, we explicitly want our models to be relatively small. Smaller models require less energy both during training and inference, which is good from a cost and environmental perspective.

4.2.1 Encoder

The encoder is the only component that is entirely shared with all the tasks. It is composed of a convolutional feature extractor followed by transformer encoder layers. The role of the convolutional frontend is two-fold: decrease the dimension along the frequency and time axes and learn the local relationships in the sequence. This has been successfully used in previous works to replace the positional encoding traditionally required by transformers [68]. A sequence of speech features of length T is reduced to a length $T' < T$ by the CNN, where $T' \approx T/4$. The resulting features are further processed by a series of 12 self-attention and feedforward layers. The output of the encoder is a sequence of features of 256 units. It is used as input to the other modules and is never optimized in a standalone fashion. The encoder has under 18 million parameters. Wav2Vec 2.0 [6] has more than 300 million.

4.2.2 Automatic Speech Recognition

For the ASR module, we use a hybrid setting with a transformer decoder coupled with a CTC classification layer [105]. The prediction units are 5,000 subword units. The CTC layer assigns to each acoustic unit a symbol that is either a token from the vocabulary or the blank symbol. A sentence can be derived either using a greedy algorithm that first removes duplicated symbols then blank tokens, or with a more fine-grained approach such as beam search. The transformer decoder generates one token at a time by comparing previously generated tokens with the acoustic evidence brought by the encoder. Following Watanabe et al. [105], the loss corresponding to this

task combines a CTC loss [32] and a cross-entropy loss. We also use label smoothing to avoid overconfident predictions. The total number of parameters for the ASR head is 13.3 million.

4.2.3 Intent Recognition

Intent is composed of an action (e.g. move) and a number of arguments (e.g. where: forward, how: slowly). We encode the intent as a multi-hot vector where the first k bits correspond to the k actions, the following l bits to the l possible values for the first argument and so on. Inspired by Touvron et al. [94], we use multihead class attention to summarize the sequence of embeddings into one prediction. We introduce a special learned token, x_{CLS}, that is attending to each element in the sequence. In the original notation from Vaswani et al. [99], we substitute the embedded queries, Q, with x_{CLS}. However, contrarily to Touvron et al. [94], we exclude the CLS token from the keys and values. Class attention compares the class embeddings, x_{CLS}, to the sequence of representations, X, to identify and extract relevant information from the embedded sequence. The task loss is binary cross-entropy. The IR module has only 825 thousand parameters.

4.2.4 Sentiment Classification

For sentiment classification, we use the same architecture as in Section 4.2.3. The loss for this task is the balanced cross-entropy (2.4).

4.3 Methodology

In this section, we describe how we train and evaluate the models. For each language, we have selected one ASR dataset to pretrain the encoder and ASR decoder and two datasets to perform multitask learning (MTL). Pretraining has shown its benefits in numerous applications and serves here as a shortcut to save considerable computing resources by initializing parts of our model. We pretrain our model with ESPnet's recipe for ASR. At this stage, we freeze the encoder's parameters, as our experiments did not show improvements when the encoder was updated. The decoder is trained with teacher forcing, meaning that the true tokens are used as input rather than the decoder's own prediction. Although this introduces a difference between training and inference, it also makes the training procedure faster by allowing parallel training, and more stable by avoiding introducing errors while the encoder is still learning.

4.3.1 Training

All downstream datasets have at least two targets: transcriptions for ASR and labels for IR or SC. Given a speech utterance, the model predicts both targets. Then, the weighted sum of the errors relative to each task is backpropagated through the network and the parameters are updated accordingly. The decoder is thus trained jointly on ASR and either IR or SC. We set the loss weights experimentally by measuring the performance on the validation set.

4.3.2 Evaluation

We evaluate the models by measuring the task performance on a left-out subset. For intent classification, we report the accuracy. Note that for calculating the accuracy, we consider that a prediction is correct if all the elements are correct – intent and arguments. For sentiment classification, we report the macro averaged f1-score.

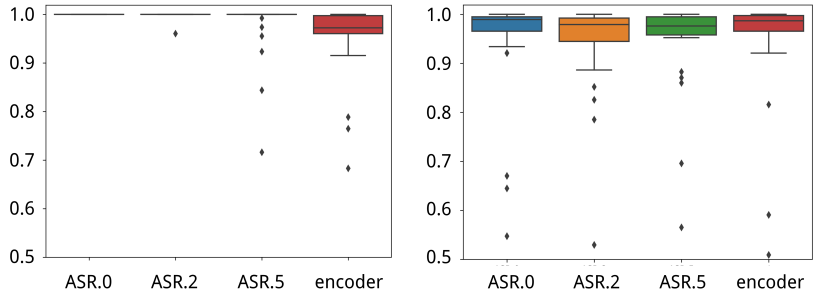
In addition, we explore the ability of our model to learn in data shortage scenarios, using as few training examples as possible. We follow the directions in Renkens et al. [79] and train our multitask models with varying amounts of training data for one speaker. In practice, for each speaker dataset, we define several training sets of increasing sizes to train and evaluate the models. This allows us to build learning curves that show the average performance of the same model on increasing training sizes. We compare these results with two types of baseline features: one is obtained by encoding the gold transcription with a pretrained BERT (NLP in Figure 4.2), and the second is obtained by transcribing speech utterances with an ASR model and encoding them with a pretrained BERT (Pipeline in Figure 4.2).

4.4 Experiments

We explore datasets in both English and Dutch. We use CGN [69] and Librispeech [70] for pretraining, Grabo and Patience for IR and SLUE-VoxCeleb for SC. The 80-dimensional Mel-filterbanks are pre-computed for all the datasets. We modified ESPnet [105] to fit our specific requirements. For the actual training of the multitask models, we use 2 datasets in Dutch and 2 datasets in English.

4.4.1 Model Selection

In this experiment, we compare the model performance on the SLU task depending on the features it receives, namely intermediary representations from the first, third or last layer of the ASR decoder or from the encoder's output. We perform the experiment first on the Grabo dataset where we build small training sets with 2 examples per class to simulate a data shortage scenario. All the models are initialized with the



(a) *Multitask Learning* **(b)** *Single task (SLU)*

Figure 4.1: Comparison of accuracy obtained from different settings on Grabo dataset for our model. The SLU module is connected to the output of the encoder (encoder), or the first (ASR.0), third (ASR.2) or last layer (ASR.5) of the ASR decoder. Each box represents the distribution of the score across speakers and folds where each model was trained on a dataset with 2 examples per class. Performance of a multitask model trained on ASR+SLU (a) and a model trained solely on SLU (b) is shown.

same pretrained weights to ensure comparable results. We perform this operation three times and report the accuracy in Figure 4.1. The model where the SLU module connects to the encoder's output shows the largest variability and the lowest average performance. The encoder's output corresponds to acoustic units that have little to do with meaning in language. The decoder, however, produces a sequence of subword units given the acoustic evidence and the previous predicted tokens. When the SLU module is connected to the encoder's output, it must summarize a longer sequence where each element conveys acoustic rather than linguistic information. In contrast, aside from the obvious advantage that the SLU head must cope with shorter sentences, the implicit language model in the decoder gives another meaning to the tokens that is dictated by the masked language modelling objective. These representations consider the acoustics through the cross-attention but also linguistic elements with the self-attention. The model trained on features from the last layer of the decoder shows an increased number of outliers, depicted by the diamonds in Figure 4.1. This indicates that there might be a trade-off between using better-defined or lower-level features as we change the capacity of the ASR module dedicated to only one task. In other words, when the SLU head is connected to the last decoder layer, the decoder has fewer parameters to learn elements necessary to ASR but detrimental, or unnecessary to SLU. Consequently, competition arises between both objectives.
In Figure 4.1, the performance of the models trained with a unique objective is lower than when we optimize for both ASR and SLU. This confirms that performing both tasks helps the performance on SLU. The difference between the models' performance in Figure 4.1b is not statistically significant, even when the encoder's output is used. This indicates that the improved performance between the models is due to the dual objective rather than the increased capacity given by the additional layers.

4.4.2 Data Shortage Scenario

When data is scarce, multitask learning can do a lot, as shown in Figure 4.2. We compare our multitask model (blue) with two baselines: we either use ASR transcripts or gold transcription to generate features with BERTje [101]. These sequences of representations are used as input to train a SLU module. As the Grabo dataset contains the exact same set of sentences repeated multiple times, we do not fine-tune BERT in order not to make the task trivial. We observe remarkable results on both datasets, Grabo and Patience, especially in the low resource scenario where our model shows impressive performance compared to the baselines. The Patience Corpus is considerably more complicated than Grabo because of the diversity of training examples and the larger number of possible outputs. Nonetheless, our model shows a better performance for small training sizes although for larger training sets, the NLP model's accuracy score rises above the results of the multitask model.

4.4.3 Results

Table 4.1: Performance comparison of our model compared to baselines [3, 86].

(a) Results on Intent Recognition

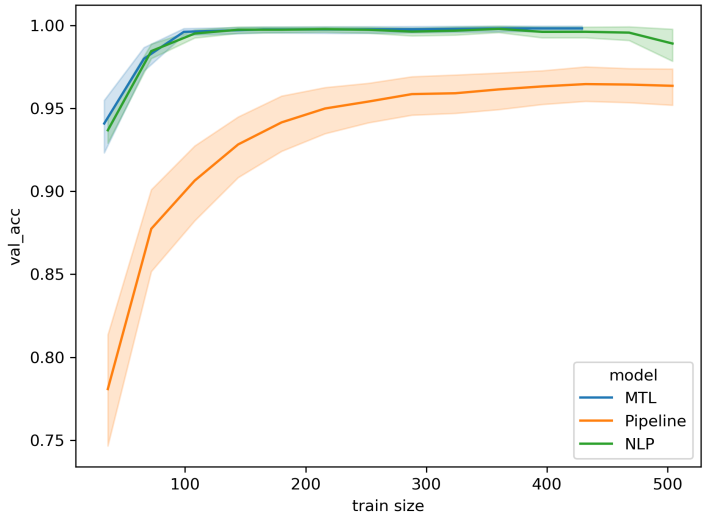
Model	Grabo	FSC Challenge	
		Spk.	Utt.
MTL	100	98.2	88.1
ESPnet-SLU [3]	97.2	97.5	78.5

(b) Results on Sentiment Classification

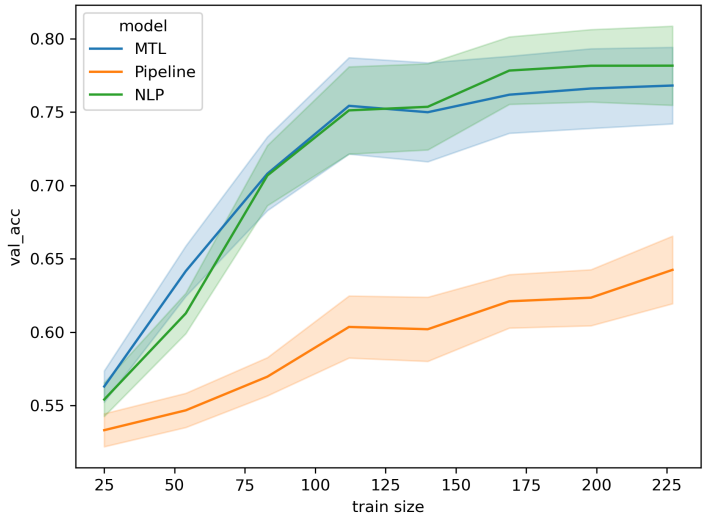
Model	# params	macro-f1
MTL (Ours)	31 M	49.82
NLP [86]	390 M	64.3
Pipeline [86]	707 M	63.3
E2E [86]	317 M	48.5

Table 4.1 compares our results with baseline models proposed in Arora et al. [3] and Shon et al. [86], on Fluent and Grabo for the intent recognition task (Table 4.1a) and SLUE-VoxCeleb for the sentiment classification task (Table 4.1b). ESPnet-SLU [3] is an autoregressive transformer encoder-decoder model with a conformer encoder and hybrid CTC-attention architecture, using the pretrained ASR model HuBERT as a feature extractor for Fluent, and no pretraining for Grabo. Although the encoder-decoder is comparable to ours in number of parameters, the feature extractor makes it very large. In Table 4.1b, the model labelled *NLP* corresponds to the performance of DeBERTa large [36] on the gold transcriptions. *Pipeline* is composed of Wav2Vec 2.0 large [6] followed by DeBERTa large. *E2E* corresponds to Wav2Vec 2.0 without a dedicated language model for decoding. All these models have many parameters and were pretrained on considerable amounts of pretraining data (more than 53.2k hours of audio for Wav2Vec 2.0 and HuBERT, 78 GB of texts for DeBERTa). These elements contribute to making the training and deployment of these models particularly expensive and resource intensive.

In Table 4.1a, the comparison of our model with ESPnet-SLU on Grabo is unfair to our advantage, as we have pretrained our representation model on Dutch data,



(a) Grabo



(b) Patience Corpus

Figure 4.2: Comparison of different models trained with increasing training set sizes on Grabo (a) and Patience Corpus (b). Each curve represents the accuracy on the test set as a function of the size of the training set. The bands around the curves represent a 68% confidence interval.

whereas ESPnet-SLU reported results on Grabo were produced without pretraining. However, the comparison with Fluent on the challenge splits show a clear benefit of our bidirectional representations, even compared to the much larger and complex conformer encoder-decoder with pretrained HuBERT feature extractor. The combination of pretraining and multitask learning helps the model to generalize to unseen output patterns, as evidenced by the strong score on FSC-U. Our model also proves to be robust against unknown speakers, something that we instigated by making sure of the diversity of the pretraining dataset. At this stage, we did not observe strong evidence that SLU positively impacts ASR scores. In Table 4.1b, we compare the performance of our model on the sentiment classification task with results reported in Shon et al. [86]. We observe a small yet consistent improvement over the end-to-end baseline, even though our model contains ten times fewer parameters, but there is still a large performance gap with the pipeline approach.

4.5 Conclusion

In this work, we proposed a multitask-learning model to do both automatic speech recognition and either intent classification or sentiment classification in parallel. We see an improvement over doing IR independently, even more so in a low-resource scenario where it is beneficial to access knowledge from written as well as spoken language. Further, we reach a better performance than the end-to-end baseline on sentiment classification, although we are using a model that is ten times smaller. We also show that representations generated by the decoder are more expressive and suggest that the implicit language model leads to better representations, especially due to the masked language modeling objective and the synergies between SLU and ASR.

Overall, this research demonstrates the feasibility and efficacy of pretraining bidirectional speech representations for low-resource spoken language understanding. By combining multitask learning with carefully designed model architectures, we have achieved significant improvements in intent recognition performance, particularly in challenging low-resource environments. Future chapters will focus on extending this approach to more complex SLU tasks and multilingual datasets.

Chapter 5

Whisper-SLU: Extending a Pretrained Speech-to-Text Transformer for Low Resource Spoken Language Understanding

5.1 Introduction

In the previous chapters, we have built a representation model for bidirectional context speech encoding, and we have fine-tuned this model using multitask learning. In this chapter, we look more specifically at pretraining and fine-tuning to answer the question in Section 1.1.3 (page 3), with three low resource SLU tasks: NER, IR and SF. The field of NLP overflows with examples of fine-tuned transformers performing well on tasks that differ considerably from what they were trained on [108]. However, the transferability of speech-to-text transformer models to other speech processing

This chapter was originally published as a conference paper with the following reference: Q. Meeus, M.-F. Moens, and H. Van hamme. "Whisper-SLU: Extending a Pretrained Speech-to-Text Transformer for Low Resource Spoken Language Understanding". In: *2023 IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2023*. 2023, pp. 189–194. DOI: 10.1109/ASRU57964.2023.10389786. IEEE: 10389786 . It was presented on December 18th, 2023 in Taipei, Taiwan.

tasks has not been extensively explored yet, even more so in SLU where data is often not abundant.

These tasks are particularly important in the development of conversational interfaces, such as those used for controlling robots, where accurate interpretation of spoken language input is crucial. For these systems to be useful, they must meet several key requirements: robustness to speaker and accent variations, ability to learn from few examples, and a small footprint to work on small appliances without sacrificing accuracy or performance.

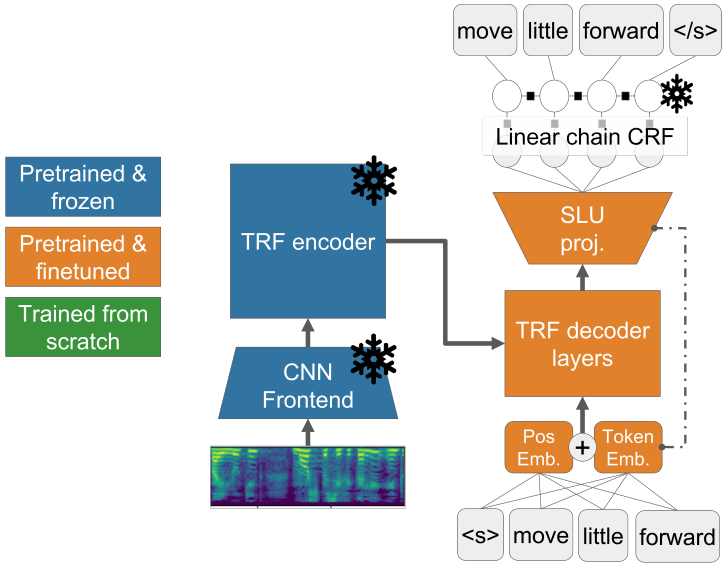


Figure 5.1: Whisper-SLU Architecture for Intent Recognition: For intent recognition, the transformer decoder is adapted for the task-specific vocabulary by replacing the embeddings and performing selective fine-tuning. The embeddings are constructed by using the original token embeddings when available or taking the average of multiple embeddings.

In this chapter, we propose to extend OpenAI's Whisper with dedicated modules for each considered task. Although other candidate models exist, like Wav2Vec2 [6] and HuBERT [42], those are encoder-only models that we think mostly encode acoustic information rather than linguistic information, as it is the case for Whisper's decoder.

For NER, we propose to add a transformer encoder that processes and tags the hidden states produced by Whisper's decoder. We take a similar approach for SF, with the difference that an additional token is predicted for the intent label, before predicting the slots. Finally, for IR, we replace Whisper's embedding and carefully select which parameters to fine-tune and which remain frozen.

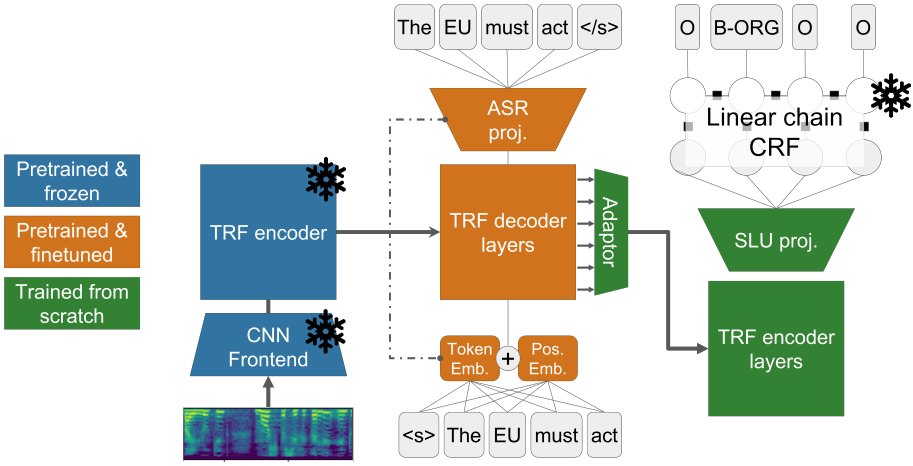


Figure 5.2: Whisper-SLU Architecture for Slot Filling and Named Entity Recognition: A transformer encoder-decoder transcribes speech and generates hidden representations. The NER module predicts entity types from the decoder hidden states following the BIO framework. For SF, we predict an additional token corresponding the intent label class.

For all tasks, we enforce the specific output structure with a Markov network. We also use test sets with withdrawn speakers and unseen phrasing to evaluate the ability of our models to generalize on challenging data.

Our main contributions are as follows: We propose an end-to-end model for low resource Spoken NER and SF that outperforms previous models without requiring an external language model. We also show how a transformer decoder can be modified for a new task by replacing the token embeddings and gradually unfreezing the parameters. This method allows transferring and completely adapting a model to a new task, even when we dispose of only a few training samples. The small footprint for both fine-tuning and inference makes it attractive for many applications.

5.2 Methodology

The models presented in this work build upon OpenAI's Whisper model. We are exploring three tasks defined in Section 2.4.3 (page 27): Spoken NER, SF and IR. In this section, we provide for each of them the specific architecture and the objective function.

5.2.1 Shared Architecture

All three proposed models share a speech encoder in the form of a transformer encoder with convolutional frontend, that takes Mel-spectogram inputs and outputs speech representations (Figure 5.1 and 5.2). The subsequent modules are specific to the SLU tasks and are described below. All three tasks have a very specific output structure that we enforce during inference with a Markov network. In low-resource scenarios, the amount of training data is insufficient to learn the transition probabilities of a dynamic CRF [51]. Therefore, we set the transition probability from i to j as $p_{ij} = 1/n_i$, where n_i is the number of legal transitions from i. Illegal transitions receive a null probability. We use the Viterbi algorithm [100] to decode the prediction given the SLU output.

5.2.2 Intent Recognition

For this task, the number of intents / arguments combination is relatively small, and the structure is well-defined. The applications of intent recognition are often specific to one use case, such as a smart home device or an automated guided vehicle. These can only perform a predefined set of actions. Moreover, data gathering is expensive, and we want to train our model with as few examples as possible. Consequently, we chose a different approach for this model than for SF and Spoken NER, and how we tackled this task in the previous chapters. We define a vocabulary as the set of possible actions (intent) together with the set of all possible argument values for each argument. For example, if a robot can only turn left or right and move forward or backward, the vocabulary size is 8 (2 intents, 4 arguments, start- and end-of-sequence tokens). We then reformulate the task as a sequence prediction problem. In the spirit of recycling as much as we can from the pretrained decoder, we keep all layers but the token embeddings, which we initialize by selecting relevant vectors in the original embeddings (Figure 5.3 left and center). We also experiment with random initial embeddings that we train with a large learning rate while keeping the rest of the parameters. We then partially unfreeze the pretrained layers and fine-tune the model with a low learning rate to allow the layers to adapt to the new task. The objective of this model is to minimize the cross-entropy loss between the predicted tokens and the true intent sequence. The evaluation metric is the accuracy, where an example is considered correct when both the intent and all slots are predicted correctly. We use Fluent Speech Commands [59] and FSC Challenge [5] for this task.

5.2.3 Spoken Named Entity Recognition

For Spoken NER, the model predicts two outputs: the verbatim transcription of the speech and a sequence of the same length containing the BIO entity tags. For the transcription task, we keep Whisper's decoder as is. An adaptor transforms the decoder's hidden states, which are then processed and tagged by a transformer encoder (Figure 5.2). The adaptor has two roles: it learns to weigh the contributions of each

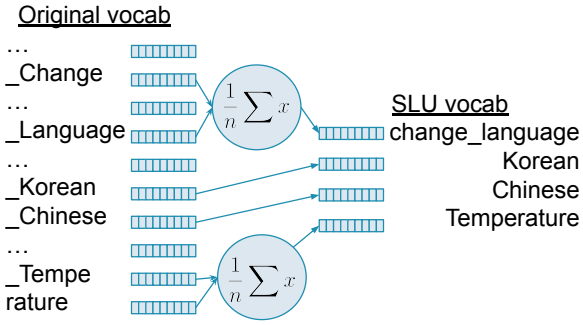


Figure 5.3: The embeddings are constructed from the original token embeddings by selecting the vector for the token with the closest meaning. When multiple candidate tokens exist, the average vector is taken.

decoder layer and sums them together, and it maps Whisper's embeddings dimension to the module dimension when these numbers differ. We adopt a multitask learning approach:

$$\mathcal{L}_{\text{NER}} = \lambda \cdot \mathcal{L}_{\text{token}} + (1 - \lambda) \cdot \mathcal{L}_{\text{tag}} \tag{5.1}$$

where the model loss, $\mathcal{L}_{\text{NER}}$, is a linear combination of the cross-entropy loss relative to the transcription, $\mathcal{L}_{\text{token}}$, and the focal loss relative to the named entity tags (Equation 2.6 on page 10). We use the SLUE-VoxPopuli [86] dataset for this task.

5.2.4 Slot Filling

As mentioned above, SF could be seen as a more difficult case of IR, where the number of slots and the possible slot values are not always known in advance. In this regard, SF is very similar to predicting entities together with an intent class. Therefore, it makes sense to keep our NER architecture. To predict the intent, we simply insert a token corresponding to the intent class in the first position of the target sequence. The loss is the same as the NER task. The evaluation metric is the accuracy, where an example is considered correct when both the intent and all slots are predicted correctly. We use SmartLights [83] and SL Challenge [5] for this task.

5.3 Experimental Setup

5.3.1 Low Resource Scenario

To simulate a low resource scenario, we sample from the original training set a predefined number of examples corresponding to 1% or 10% of the original dataset size. We repeat the operation 10 times with new random seeds to get 10 different training set. We do the same on the validation set. Once trained, the model is evaluated on the full test set.

Table 5.1: Average accuracy scores on Fluent Speech Commands (original splits) and SmartLights (random splits). Standard deviations for Whisper-SLU on FSC 1%, 10% and 100% are respectively 0.31, 0.05 and 0.03 and 1.28 on SmartLights.

Dataset	Fluent Speech Commands			SmartLights
Train size	1%	10%	100%	100%
Whisper-SLU	**97.19**	**99.46**	99.66	**95.35**
Lugosch et al. [59]	82.78	97.96	98.80	-
CMCL [24]	96.89	99.37	**99.69**	92.5

5.3.2 Implementation Details

The models were implemented in PyTorch [71] with the Transformers library [108]. The pretrained model is whisper-large-v2 for NER and whisper-small for SF and IR. We use different models because we want to keep the IR and SF model sizes reasonable, as those models will typically be used on small appliances. The task modules for NER and SF have 2 layers and 12 attention heads with each 64 hidden units for a total dimension of 768. Since whisper-large-v2 has a hidden dimension of 1,280, we apply a linear transformation to get the desired 768 dimension, after having applied the convex linear combination of the hidden representations of the decoder (adaptor in Figure 5.2). The weights for the SLU and ASR losses are respectively 0.1 and 0.9. We set the focus of the focal loss to 1. We train the models with AdamW [57] and a learning rate that peaks at 4e-6 for NER and 1e-4 for SF and IR, with a cosine schedule with warm up corresponding to 10% of the total number of steps. The large model was trained with an effective batch size of 128 for 3,000 steps, or about 40 epochs. Following Watanabe et al. [105], after training the models, we average the weights of the last 5 checkpoints. When training in the low-resource settings, we sample either 1% or 10% of the training set with a different seed and train the model with the resulting datasets, and we measure the performance on the test set. We report the average performance along with the standard deviation obtained over 10 runs. The number of steps for 1% and 10% subsets was 500 and 1000 for the full SF and IR datasets. The batch size, learning rate and dropout were chosen experimentally by measuring the performance on the validation set.

Table 5.2: Named Entity Recognition errors breakdown (dev).

	F1-score (↑)	label-F1 (↑)	WER (↓)
QUANT	73.3	87.3	-
PLACE	91.3	94.1	-
WHEN	63.0	83.7	-
ORG	70.7	82.4	-
LAW	30.3	65.5	-
NORP	84.2	90.9	-
PERSON	47.6	89.5	-
overall_micro	76.3	88.1	-
overall_macro	65.8	84.8	7.3
Oracle text [86]	87.5	91.1	0.0
Pipeline w/ LM [86]	74.9	87.4	9.1
E2E w/ LM [86]	70.2	79.0	9.1
SSL pretraining [72]	74.5	88.0	9.3
SLU pretraining [72]	75.7	87.5	9.0
Ensemble [72]	77.2	88.7	8.6

Table 5.3: Average accuracy and standard deviation (%) on FSC Challenge and SL Challenge test sets.

Test set	FSC Challenge (Spk.)			FSC Challenge (Utt.)			SL Challenge	
Train size	1%	10%	100%	1%	10%	100%	Spk.	Utt.
Whisper-SLU	93.0 (1.38)	96.1 (0.26)	98.3 (0.06)	88.7 (1.49)	91.6 (2.09)	93.3 (1.19)	90.6	81.1
Lugosch et al. [59]	-	-	92.3	-	-	78.3	82.6	78.5
ESPnet-SLU [3]	-	-	97.5	-	-	78.5	-	-
MTL [63]	-	-	98.2	-	-	88.1	-	-

5.4 Experimental Results

In this section, we present the results of our experiments on the three considered tasks using the proposed models. We compare our models to several baselines defined in Section 5.1.

5.4.1 Intent Recognition

We explore the robustness of our models to small training sets. We perform this analysis on the challenge splits in order to evaluate the robustness of our models to unknown speakers and utterances in these data scarcity scenarios. Since the training set has 19,262 examples, we train our 1% and 10% models on 192 and 1926 examples. Since these splits have few published results, we perform the same analysis on the original splits and compare our findings with two other papers, described in

Section 5.1. We sample 10x from the training set and report the average score and standard deviation on the test sets (Table 5.1 and Table 5.3).

To fine-tune the models, we first train the embeddings for 300 steps with a learning rate that reaches 5e-3. The fully-connected layers and layer norm are then unfrozen and fine-tuned at a lower learning rate of 1e-4 for an additional 100 steps. We observe that if this unfreezing schedule is not followed, the performance drops drastically (from $94.79 \pm 1.56\%$ to $37.9 \pm 18.5\%$ on the 1% dev sets). This is because when training starts, the embeddings are still random while the other parameters are pretrained, leading to the first few updates being counterproductive. We compare the resulting embeddings with Whisper's and discover that for each token vector, the closest embedding is a token from the original vocabulary that is semantically close (e.g. down / decrease, language / change_language, more / increase, etc.). This motivates us to build the initial embeddings from the pretrained model by selecting tokens with a similar meaning and extracting their representations.
After training the embeddings, we fine-tune the fully-connected layers from the transformer decoder with a low learning rate. Although the performance was already high from only training the embeddings ($94.91 \pm 1.20\%$ on the 1% dev sets), this additional step gets us to $96.18 \pm 1.67\%$. We also try to unfreeze the whole decoder and fine-tune the self- and cross-attention layers, but the performance does not increase any further. These experiments are conducted on the 1% datasets. Similar results are observed on the larger datasets, although with a smaller effect as more training data is available.

5.4.2 Named Entity Recognition

Our proposed model achieves an F1-score of 76.3% and a label-F1 of 88.0% on the development set, outperforming the end-to-end and pipeline baselines of the SLUE benchmark and all approaches from Peng et al. [72] except for the ensemble of their four best models. The WER on the development set is 7.3%, or 1.3 absolute points lower than the ensemble in Peng et al. [72], 1.8 lower than the best baseline in Shon et al. [86]. The organizer of the challenge communicated us a WER of 8.8% and a F1-score of 70.1% on the test set. Our breakdown analysis (on the dev set, as the test set is blind) shows that most errors are due to transcription errors. This is shown by the difference between F1-score and label-F1 in Table 5.2. In other words, the model is able to correctly identify entities, although it sometimes spells them incorrectly. This is especially the case for names, laws, and events.

5.4.3 Slot Filling

SL Challenge [5] only contains 1 hour of training data. The evaluation is done on two test sets: speaker and utterance. The former contains utterances by speakers that are not present in the training set, to evaluate the model's robustness to new speakers. The latter contains utterances whose choice of word and vocabulary is different from

the training set, to evaluate how the model generalizes to unseen sentences. Only the original paper reports results on these splits, by reevaluating the model proposed in Lugosch et al. [59] to get respectively 82.6% and 78.5%. Our method gives an accuracy of 90.55% and 81.1% respectively (Table 5.3). This shows that our model generalizes well to unknown utterances and is robust to new speakers. We also present results on random splits, averaged over 10 runs (Table 5.1). As in the previous section, a breakdown of the errors shows that the intent classification and the entity identification are mostly correct, and entity spelling mistakes are almost 3x more frequent.

5.5 Discussion and Conclusions

In this chapter, we explored the transferability of pretrained speech-to-text transformer models for the tasks of intent recognition, named entity recognition, and slot filling with a particular focus on low-resource datasets. For the first task, we proposed a new approach to fine-tuning by modifying a pretrained model's embeddings to fit a new vocabulary. We observed that, when done correctly, we can reach impressive performance by updating only a handful of parameters. We showed that there is much to gain by carefully selecting and building the embeddings from the original model rather than random initialization. For the other tasks, we added dedicated modules to the pretrained model. In all tasks, we proposed a Markov network that does not require training to enforce the task-specific output structure. The proposed models outperformed several baselines in all three tasks, using varying amounts of training data.

Overall, our study contributes to the development of robust and efficient conversational interfaces by addressing key challenges in SLU, such as working with very few training data, robustness to accents and speakers, and small footprint for training and inference. In the next chapters, we will continue exploring this topic applied to different languages.

Chapter 6

MSNER: A Multilingual Speech Dataset for Named Entity Recognition

6.1 Introduction

In our increasingly interconnected world, where language barriers are slowly fading, the field of speech processing is rapidly turning its focus towards multilingual applications. A crucial area within this domain is Spoken NER, (2.4.3 on page 27). However, there are few resources available to researchers and even fewer are freely accessible. One such resource is SLUE-VoxPopuli, introduced in Section 2.5.2 and used in Chapter 5. However, to our knowledge, no other openly distributed Spoken NER dataset exists.

Responding to the growing demand for cross-lingual research and comprehensive evaluation resources in Spoken NER, we have manually annotated the test sets of the widely-used VoxPopuli speech dataset in four languages: Dutch, French, German, and Spanish. Additionally, we offer machine-generated annotations for the training and validation sets, facilitating further research and model development in this domain.

This chapter was originally published as a conference paper with the following reference: Q. Meeus, M.-F. Moens, and H. Van hamme. "MSNER: A Multilingual Speech Dataset for Named Entity Recognition". In: *LREC-COLING ISA-20 Workshop*. 2024. arXiv: 2405.11519. It was presented at the Twentieth Workshop on Interoperable Semantic Annotation (LREC-COLING) on May 20th 2024 in Torino, Italy.

This chapter allows us to explore the question raised in Section 1.1.4 (page 4). In the following sections, we provide a detailed overview of our efforts in the domain of Spoken NER.

First, we introduce the newly annotated dataset and provide information about its size, multilingual coverage, and its potential significance in advancing Spoken NER technology. Additionally, we describe the methodology employed to create the dataset, breaking down the annotation process and data preparation. We also introduce the user-friendly annotation interface we have developed for this purpose. Finally, we present the results of various experiments and benchmarks conducted using this dataset. These experiments demonstrate its utility in evaluating Spoken NER models across the chosen languages, highlighting its role in advancing research and development in this field.

Table 6.1: Dataset statistics. The MSNER dataset [65] covers L1–L4, and SLUE-VoxPopuli dataset [86] provides L5.

Subset	Train			Validation			Test		
Language	hours	examples	entities	hours	examples	entities	hours	examples	entities
DE – L1	224.5 h	86,410	97,492	4h	1,610	1,880	5h	1,966	2,061
ES – L2	141.5 h	47,611	66,482	4h48	1,529	2,094	5h	1,512	2,198
FR – L3	186h	65,952	80,255	4h22	1,527	1,884	4h30	1,656	2,004
NL – L4	38.5 h	16,533	19,566	2h16	963	1,074	2h30	1,120	1,272
EN – L5	14.5 h	5,000	5,382	5h	1,753	1,682	5h	1,842	1,854

6.2 Dataset Description

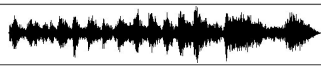
ID	20090423-0900-PLENARY-26-fr_20090423-21:55:26_4
Audio	
Text	200 milliards d'euros qu'il faut rapprocher aussi du niveau des déficits des pays européens.
Entities	(MONEY, 200 milliards d'euros) (NORP, européens)

Figure 6.1: Annotated Example

The MSNER dataset is an annotated version of the VoxPopuli dataset [102] in four languages – Dutch, French, German, and Spanish. VoxPopuli is a collection of recorded sessions from the European Parliament, segmented to contain one or more sentences by one speaker. For each language in scope, we provide three annotated subsets (Table 6.1): a training and development set with machine-generated "silver" annotations, and a test set with manual "gold" annotations. The subsets of the four languages in scope were annotated according to OntoNotes' 18 classes. The test sets were manually

Table 6.2: Number of annotated entities per entity type in the test sets. Column SLUE correspond to the 'combined' entity set proposed by Shon et al. [86].

OntoNotes5	SLUE	DE	ES	FR	NL	Examples
date	WHEN	307	276	243	113	125 years ago, 15 maart, 1815—1830, 1997
time		12	21	10	8	24 hours, acht uur, de hele dag, mañana
cardinal number	QUANT	136	167	123	91	1, 10, 10 miljoen, 11, 11 billion
ordinal number		82	100	79	45	First, Ten derde, dritten
quantity		6	2	5	1	one and a half meter, two inches
money		26	16	18	8	200 million EUR, Dertig miljoen euro
percent		21	28	13	22	1 procent, 100%, 15 Prozent
geopolitical area	PLACE	259	285	283	176	Amsterdam, Australië, Barcelona, Belgium
location		128	139	214	110	Afrika, Balkanlanden, Europe
group	NORP	229	244	285	213	African, American, Christian
organization	ORG	621	638	527	362	Amnesty International, Charlie Hebdo
law	LAW	64	108	33	22	Paris Accords, US Constitution
person	PERSON	123	131	100	67	Angela Merkel, Barroso, Beyoncé
facility	-	6	2	8	12	Guantánamo, White House
event	-	23	25	21	8	Europees Semester, Rio conferentie
work of art	-	6	3	4	4	Green Book, Koran
product	-	4	1	2	8	2G, 4G, 5G, iPhone
language	-	3	12	6	2	Latin, Nederlands, Español

annotated by the authors following the methodology outlined in Section 6.3. Each example in the annotated dataset contains the VoxPopuli ID to identify the relevant audio recording in the original dataset, the transcribed sentence and the annotated named entities, that is, the list of entities, each composed of a text and a label component (Figure 6.1). For the silver label datasets, we also provide a probability score of each predicted entity. We discuss in Section 6.5 how this number is related to the uncertainty of the model.

We use the 18-classes OntoNotes label set [107]. However, following the example from Shon et al. [86], we provide annotations by using an alternative label set that combines entity types like places or numbers and discard the rarest classes like languages, events, and work of art (Table 6.2 Column 2).

6.3 Methodology

We provide two kinds of label quality: machine-generated "silver" labels and human-annotated "gold" labels. For obvious reasons, the silver labels are much cheaper and easier to produce. Therefore, we only provide human-made annotations for the test sets, and the training and validation sets annotations are entirely machine-generated. The methodology follows these four broad steps: (1) filtering out recordings without or with misaligned transcripts, (2) generate silver labels for all subsets, (3) manually annotate the test sets and (4) verify the human-made annotations to identify and rectify potential labelling errors. We detail each step in the following paragraphs.

6.3.1 Filtering

The VoxPopuli dataset contains a few alignment errors between the spoken content and its corresponding transcript. To address this issue, we employed an automatic speech recognition (ASR) system, initially transcribing the spoken utterances and subsequently calculating the word error rate by comparing the ASR-generated sentence to the provided transcript. For this task, we opted for the Whisper large v2 ASR model [75], because it showed near state-of-the-art performance across the selected languages. Notably, this model has been meticulously trained on extensive, well-curated data to perform both audio translation and transcription tasks.

For the training and development sets, we filter out examples with a WER larger than 20%, without verifying that the excluded examples were indeed problematic. This discards about 20% of the German and Dutch utterances, 10% of the French examples and 6% of the Spanish utterances.

For the test sets, instances where the word error rate (WER) between the machine-generated transcription and the original transcript exceeded 20%, we conducted a meticulous review process. This involved listening to the audio recording and cross-referencing it with the existing transcript. When feasible, we made necessary corrections to the transcript. However, in cases where multiple speakers were heard in the recording or no speech is present, we removed the problematic utterance from the dataset.

6.3.2 Pseudo-annotations

We employed an established text-based Named Entity Recognition (NER) model to predict entities within the gold transcript. We chose to use the XLM-RoBERTa large pretrained model [18], fine-tuned specifically on the OntoNotes v5 dataset [107]. This model is readily accessible through the HuggingFace repository[1].

While it's important to note that this particular model's fine-tuning was conducted solely on English data, we were surprised by its ability to transfer across multiple languages. In our evaluation, we observed impressive performance, with most sentences annotated correctly and few requiring manual intervention, as we see in the next section.

6.3.3 Annotation Tool

For each of the 6,254 pre-annotated sentences in the test sets, we corrected the annotations predicted by the model. For this purpose, we have developed a command line tool to quickly add, edit, merge or remove annotations in a sentence. This utility displays the pre-annotated sentence with a summary of the annotations below.

[1] hf.co/asahi417/tner-xlm-roberta-base-ontonotes5

Annotations appear as colored XML tags both in the text and in the summary. An annotated English translation can be displayed. The annotator then has access to both the original sentence and the translation to make sure that the annotations are as accurate as possible. When presented with a sentence, the annotator has the choice to add a new annotation, delete an existing one, merge two annotations together or modify an annotation, either by changing the type or by adding or removing words. Once a sentence has been annotated, it is saved to a file in JSON format. Following this methodology and with the help of this tool, we were able to save a lot of time and effort without sacrificing accuracy. For this reason, we make the tool available online so that others will have the opportunity to contribute to this field of research by easily annotating more data in many more languages.

As mentioned in Section 6.2, we not only provide annotations according to OntoNotes 18 classes, but also the 7-classes combined set proposed in Shon et al. [86]. However, we chose to completely reannotate the examples where entities are removed, instead of simply removing all the annotations of the same type from the dataset. To illustrate this, consider the following example: `<event> 15th conference on speech of Toronto </event>`. According to the combined set conversion rules (Table 6.2), all the entities of type `<event>` are to be discarded. Doing that would lead to two unannotated entities, “15th” as a number and “Toronto” as a place. Instead, we re-annotate the examples containing removed entities to make sure that we are not penalizing the models for correct assumptions.

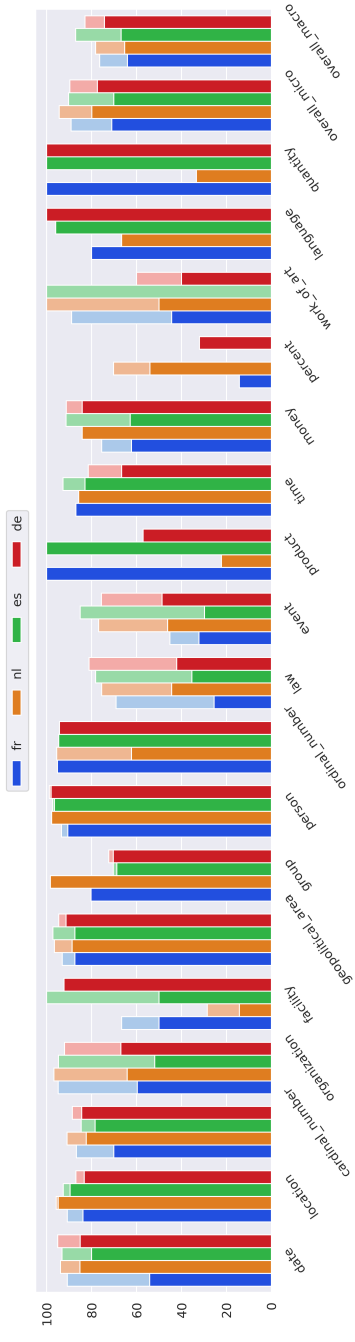


Figure 6.2: Evaluation of text-based pretrained NER model against our annotations. Bright colors correspond to the F1-score and faded colors correspond to the label-F1 score, a metric that ignores spelling mistakes and segmentation errors.

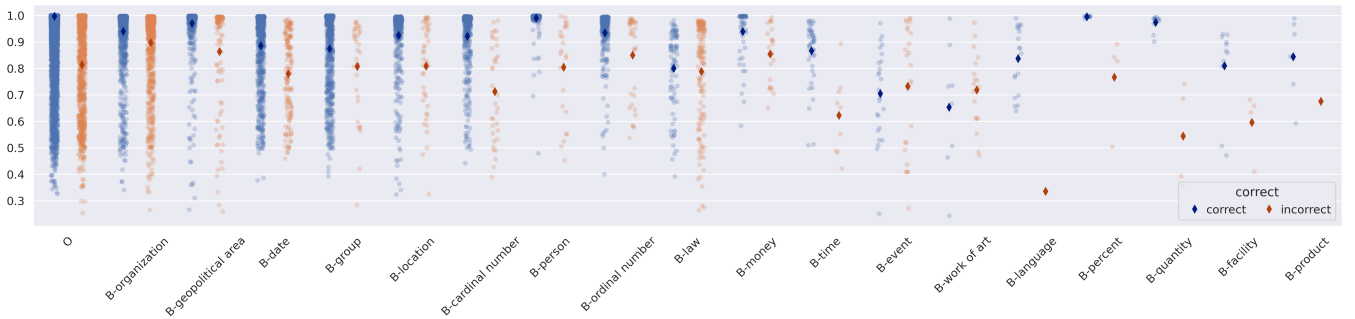


Figure 6.3: Distribution of predicted probability score per class given the target class for the predictions of the text-based NER model

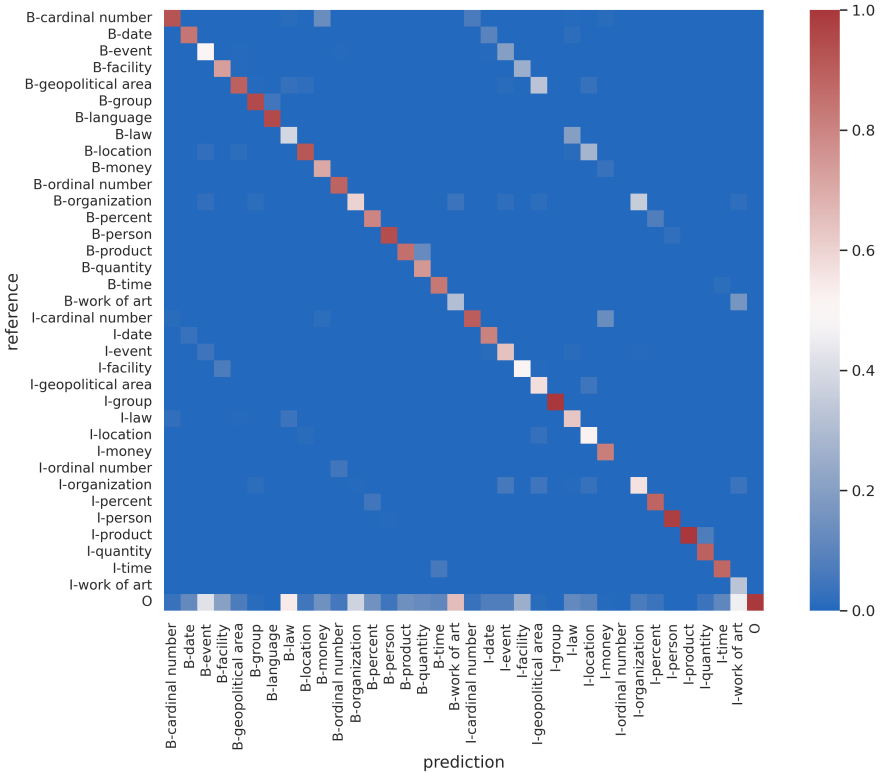


Figure 6.4: Confusion matrix, normalized to show the probability distribution of the tags predicted with the text-based NER model.

6.3.4 Verification

Finally, we verify the integrity of the test annotations by deriving a number of heuristics and rules that the annotations must abide. This involved grouping the annotations by category and verify each list one by one, comparing them to one another, searching in the text for frequent annotated terms to identify missing annotations, etc. In this last step, we also fix some remaining transcription issues. For example, we realized that VoxPopuli transcripts omitted the symbol "%", and sometimes the word "thousands" (in all languages). Consequently, for all entities marked as cardinal number, we added the missing tokens when necessary, following the rules specific to the language[2]. Another error often made by the text-based NER model is to predict the article as

[2] In French and in Spanish, the symbol "%" is generally used, but in German and in Dutch, the word is more commonly spelled as "Prozent" or "procent", respectively.

being part of the entity. As multiple sources advocate against doing so, we abided by the main guidelines [61, 9].

6.3.5 Distribution

The annotated datasets are distributed in two formats: As JSON Lines files available on GitHub[3], and on the HuggingFace repository [108]. There is one file per subset and per language, where each line is an annotated example. The audio files can be obtained by downloading VoxPopuli and matching the audio ID. The dataset version hosted on HuggingFace contains the audio recordings and the preprocessed annotations in BIO format, so that a researcher can already use the dataset after only two lines of code.

6.4 Evaluation Metrics

Following Shon et al. [86], we recommend evaluating model predictions with the micro-averaged F1-score. The F1-score is the harmonic mean of precision and recall, calculated from an unordered list of named entities predicted for each utterance. Precision is the proportion of correctly predicted entities among all predicted entities, and recall is the proportion of ground truth entities that were correctly detected. An entity is considered to be predicted correctly if both the type and spelling are identical to the ground truth. To allow multiple entities with the same spelling and type in a sentence, we add a unique identifier to each entity/type pair. We recommend using the micro-averaged F1-score because the dataset is unbalanced. The label F1-score only considers the predicted type of the entity for correctness, leaving the transcribed entity out of the computations. This metric ignores spelling mistakes and segmentation errors. We provide an evaluation script[4] to compute these metrics and generate a breakdown of the prediction results per entity type.

6.5 Experiments

6.5.1 Setup

The first analysis compares the annotated test sets to the pseudo-annotations generated by the text-based NER model. Since the silver-label training and validation sets were generated with this model, this analysis is valuable for anyone intending to use these datasets for training. Indeed, it gives insights into the entities that are often confused with one another or remain undetected. It also gives some insights on the reliability of the model's confidence score in assessing whether a prediction is correct.

[3] github.com/qmeeus/MSNER
[4] https://raw.githubusercontent.com/qmeeus/MSNER/main/src/evaluate.py

We also consider two methods to predict named entities from speech, with a pipeline and an end-to-end model. The end-to-end model is a transformer encoder-decoder trained to perform both ASR and NER with a multitask objective [67]. This model is initialized from Whisper Large V2 [75], with an additional SLU module connected to the layers of the decoder with an adaptor. The end-to-end model was fine-tuned on English SLUE-VoxPopuli [86]. The pipeline model transcribes the audio files and subsequently annotates the transcriptions. For the ASR model, we use Whisper Large V2 [75]. For the pipeline model, we provide two options to allow for a better comparison. In Table 6.3, we use XLM-RoBERTa fine-tuned on OntoNotes v5 [107] and compare it to the predictions generated by the text-based NER model from the gold transcripts. In Table 6.4, we fine-tuned the same XLM-RoBERTa on SLUE-VoxPopuli [86], which provides a fair comparison to the end-to-end model.

Although both models rely on multilingual pretrained models, the fine-tuning dataset is entirely in English. Therefore, we evaluate the ability of these models to generalize from one language (English) to other languages (Dutch, French, German, and Spanish). Before computing the F1-scores, we normalize the text by putting it in lower case and removing symbols. It should be noted that the evaluation script does normalize the text further, which could have its importance depending on the model to be evaluated.

All results are presented on the human-annotated test sets proposed in this chapter.

6.5.2 Results

Figure 6.3 shows the distribution of calculated probabilities for predicted 'B' and 'O' tags conditional to whether they were predicted correctly or not. For each token position k, the probability of the most likely tag i^* is computed as follows:

$$P(y^k = i^*) = \max_i \frac{e^{z_i^k}}{\sum_j e^{z_j^k}}$$

where $z_{1..N}^k$ are the logits predicted by the model for the token at position k. We observe that, on average, annotations for which there was no agreement between the annotator and the NER model were predicted with a lower probability than annotations that were correctly annotated from the start. However, we observe major differences between the class distributions. For the most frequent classes, like 'O', 'organization' or 'date', the probability distributions overlap considerably, and one should be careful if using this score as a proxy for the model's uncertainty. This is not surprising, as transformers are known to be overconfident [110]. For rare quantitative classes like 'percent' and 'quantity', the model shows confidence when predictions are correct, and uncertain otherwise. This indicates that for those particular classes, the given probability could be relied upon when estimating the model's uncertainty. The score breakdown by entity and language (Figure 6.2 on page 72) indicates that in general, there are no major differences across languages, except for rare classes, where the variability increases significantly.

Figure 6.4 shows the confusion matrix of the NER model predictions against the manual annotations. Most errors are undetected entities (bottom row in Figure 6.4) and segmentation errors (I-tags predicted instead of B-tags and inversely, are visible on the lighter diagonals above and below the main diagonal). Some entities remain undetected more often than not, e.g. “work of art” and “event”, which is a sign that predictions are less reliable for these rare classes. Some other types are often confused with one another, like “money” and “cardinal number”. However, all types seem to have at most two confused types. We notice that “geopolitical area” is most often confused with “location” and “law”. In the latter case, this is because many laws are named after cities (e.g. the Paris Agreement, the Warsaw Treaty).

Table 6.3 compares the text-based NER predictions with the NER predictions obtained from the ASR transcript and generated by the same text-based NER model. The OntoNotes dataset, although in English, provides many well-curated annotations and the NER model trained on this dataset seems to generalize well to the other languages. However, this model was not trained to handle automatic transcripts and we observe a considerable drop in performance when it is asked to process ASR outputs. To make a fair comparison with the end-to-end model, we fine-tune XLM-RoBERTa on SLUE-VoxPopuli and report the results in Table 6.4. The fine-tuning dataset being of smaller size (14.5 hours of training data), the models do not have many examples to learn from. The end-to-end model has a slight advantage because it learns simultaneously the ASR and NER tasks, and it is able to share part of its architecture between both tasks. For example, it is able to identify the presence of entities despite a lot of transcription and segmentation errors, as evidenced by the large label F1-score. In contrast, the pipeline suffers much more from the transcription errors because it was pretrained on curated texts and is not expecting noisy ASR transcriptions.

The text-based NER model performs best for Dutch, then German, French and finally Spanish. As the model was trained on English annotations, this ranking is not a surprise, although the ability of the model to transfer to other languages is impressive. However, for the speech processing models, the same conclusion cannot be drawn. The entity F1-score seems to be correlated with the word error rate, which is influenced by the availability of the different languages in the pretraining set. In other words, for speech models, this is the model’s ability to transcribe foreign languages that will drive the quality of the predictions, rather than how similar the evaluation and the pretraining language are. The label-F1 indicates how accurate a model is at detecting the presence of entity types, disregarding its ability to transcribe it correctly. Looking at those numbers, we observe again the same behavior as with the text-based entity predictions, namely that entities are more likely to be accurately detected when the evaluation language is more similar to the fine-tuning language.

6.6 Conclusion

In this chapter, we have presented MSNER, a new dataset for evaluating multilingual Spoken NER systems. Although NER is a popular topic in NLP, this task has

Table 6.3: Performance of text-based NER model trained on OntoNotes. Gold corresponds to the model's predictions from the gold transcripts and ASR corresponds to the model's predictions on the ASR transcripts.

Model	Metric	DE	ES	FR	NL
Gold	F1 (↑)	77.4	70.1	71.1	79.9
	Label-F1 (↑)	89.7	90.3	89.1	94.4
ASR	F1 (↑)	52.4	50.6	44.7	52.7
	Label-F1 (↑)	66.2	63.6	59.4	66.1
	WER (↓)	12.0	8.6	11.1	13.1

Table 6.4: Provided baselines on the annotated test sets for a pipeline ASR/NER model and an end-to-end multitask model. Both models were fine-tuned on SLUE-VoxPopuli [86]

Model	Metric	DE	ES	FR	NL
Pipeline	F1 (↑)	30.8	36.3	37.2	36.3
	Label-F1 (↑)	42.7	51.6	49.5	45.9
	WER (↓)	12.0	8.6	11.1	13.1
End2End	F1 (↑)	38.3	41.3	39.6	31.2
	Label-F1 (↑)	76.8	77.1	78.3	78.4
	WER (↓)	13.3	10.5	14.5	18.2

remained mostly unexplored in speech processing and spoken language understanding. To address this issue, we have used a pretrained model to annotate the VoxPopuli training and validation subsets in Dutch, French, German, and Spanish. Additionally, to provide researchers with a gold standard dataset for evaluating their Spoken NER models, we have manually annotated the test sets for these subsets. By analyzing the predictions of a text-based NER model, and comparing them with our annotations, we were able to identify points of attention for researchers who intend to train a model on silver annotations. For example, in some cases, the model confidence on the predictions can serve as a basis to estimate the correctness of the prediction, but this must be done carefully, since we have seen that transformers can be overconfident. Counter-intuitively, we have shown that most frequent classes are not always the ones where the model's confidence is most trustworthy. We also looked at the classes that were often confused with one another, which gave us some ideas about which errors might be present in the training and validation sets.

We also provide baselines on the newly annotated evaluation subsets. A pipeline and an end-to-end SLU model were selected, both fine-tuned on English SLUE VoxPopuli [86], and we evaluate them on the manually annotated test sets. We saw that in a low resource scenario, the end-to-end model seems to benefit from learning simultaneously

to transcribe and to annotate, which allows a better generalization across languages than the pipeline model fine-tuned on the same dataset. Finally, we found that the performance of text-based models on unseen languages is correlated with the similarity of the evaluation language to English. However, for speech models, this is the multilingual transcription accuracy that is the main driver for NER performance. Interestingly, we have seen that the end-to-end model was able to identify the presence of entities much better than the pipeline model, despite a similar overall performance, which illustrates the advantage of sharing parameters across tasks.

While the results demonstrate the potential of using text-based models for efficient dataset creation, our research also highlighted potential pitfalls. We observed instances of model overconfidence and varying reliability across different entity types. This could introduce biases into the resulting silver datasets, potentially impacting the performance of downstream models. Researchers leveraging such automatically annotated data should be aware of these biases and consider strategies to mitigate them, either through manual correction or programmatic adjustments.

To effectively utilize text-based NLU models for multilingual SLU, we propose the following approach. First, high-performing text-based NER models can be used to annotate large volumes of speech transcripts, particularly for low-resource languages where manually annotated data is scarce. Second, to ensure the quality of these annotations, it is crucial to apply confidence thresholds and filter out or correct uncertain predictions. Focusing on high-confidence predictions increases the likelihood of accurate annotations. Third, manual annotation efforts should be directed towards examples where the model demonstrates low confidence or struggles to differentiate between specific entity types. This targeted approach maximizes the impact of human intervention. Finally, a thorough error analysis of the text-based model's performance on speech data is essential. This can uncover patterns of misclassification and reveal common issues that need to be addressed during the subsequent training of speech-based SLU models.

By combining the efficiency of automated annotation with targeted human intervention and careful error analysis, we can create high-quality training datasets that accelerate the development of more accurate and robust multilingual SLU models. This hybrid approach not only reduces the manual annotation burden but also allows us to harness the strengths of both text-based and speech-based models to overcome the challenges inherent in multilingual SLU.

Chapter 7

Multilingual Spoken Named Entity Recognition with Transformers

7.1 Introduction

In Chapter 6, we have introduced a dataset, MSNER, for the task of spoken NER in four different languages. This chapter seeks to go past the preliminary analyses and addresses multilingual Spoken NER by proposing end-to-end SLU approaches for this task, while answering the question introduced in Section 1.1.5 (page 5).

We introduce Whisper-SLU end-to-end, a model for end-to-end Multilingual Spoken NER and compare its performance across 5 languages. We show that multilingual training leads to better performance than using a monolingual dataset. In a second time, we look at methods to mitigate catastrophic forgetting, a phenomenon where a model forgets during fine-tuning previously learned tasks and languages. Three methods are proposed to achieve significantly less forgetting compared to a naive fine-tuning approach: embedding reset, disassembling the embeddings and experience replay.

In this chapter, we research how we can fine-tune pretrained speech-to-text models for

This chapter is submitted as a journal paper with the following reference: Q. Meeus, M.-F. Moens, and H. Van hamme. "Multilingual Spoken Named Entity Recognition with Transformers (under review)". In: *Natural Language Processing* (2024).

multilingual SLU tasks, with a specific focus on catastrophic forgetting and mitigation methods. The first method resets the embeddings to their pretrained value after training. A second proposed method disassembles the embeddings and freezes the pretrained embeddings during training, while allowing the task specific parameters to be learned. Finally, we propose to use the pretrained model to generate a pseudo-dataset for experience replay, a useful method when pretraining data is lacking.

We propose to make architectural modifications to Whisper to enable multilingual SLU tasks. Our first model, Whisper-SLU (Chapter 5), adds token classification layers on top of Whisper and processes directly the hidden states without forcing a transcription. Whisper-SLU end-to-end extends Whisper's input vocabulary to produce multilingual speech transcriptions with inline annotations.

7.2 Whisper-SLU

Whisper-SLU builds on Whisper by adding two additional transformer encoder layers for SLU (Figure 5.2). The pretrained Whisper model is trained to transcribe (X $\rightarrow$ X) and translate (X $\rightarrow$ en), at the same time generating hidden representations of the spoken content at each layer. The SLU module directly takes decoder hidden states as input. The model learns to weigh the contributions from each decoder layer and combines them into a single representation, further processed by a linear layer to map these embeddings to the classifier's hidden dimension. This combined representation is fed into a 2-layer transformer encoder and projected to the output space to generate, for each position, an entity tag prediction in the BIO format.

Whisper-SLU bridges the gap between fully end-to-end and pipeline approaches. It possesses dedicated SLU layers offering more flexibility compared to the end-to-end model. Unlike a pipeline with separate ASR and NLU modules, both tasks occur within the same model. Crucially, the SLU module processes hidden states directly from the ASR decoder instead of relying on outputs like logits or language tokens. This enables handling uncertainty in ASR predictions. Additionally, the bidirectional attention mechanism of the transformer encoder allows the model to consider the entire sequence for informed predictions.

This architecture allows separate training for each module or a multitask learning approach using a weighted sum of respective module losses:

$$\mathcal{L}^{\text{mtl}} = \frac{1}{L}\sum_{l=1}^{L}[(1-\alpha)\mathcal{L}_l^{\text{asr}} + \alpha\mathcal{L}_l^{\text{ner}}] \tag{7.1}$$

$$\mathcal{L}_l^{\text{asr}} = -\sum_{k=1}^{K} y_{k,l}^{\text{asr}} \cdot \log p_{k,l}^{\text{asr}} \tag{7.2}$$

$$\mathcal{L}_l^{\text{ner}} = -\sum_{c=1}^{C}(1-p_{c,l}^{\text{ner}})^{\phi} \cdot y_{c,l}^{\text{ner}} \cdot \log p_{c,l}^{\text{ner}} \tag{7.3}$$

where $\mathcal{L}^{\text{mtl}}$ is the model loss for one sequence of length L, $\mathcal{L}_l^{\text{asr}}$ and $\mathcal{L}_l^{\text{ner}}$ are the ASR and NER losses for the item in l^{th} position. K and C are respectively the

number of tokens and NER classes, $y^{\text{asr}}_{k,l}$ and $y^{\text{ner}}_{c,l}$ equal to one if the l^{th} item in the sequence is respectively token k and tag c, and zero otherwise, and $p^{\text{asr}}_{k,l}$ and $p^{\text{ner}}_{c,l}$ are the probabilities predicted by the model for token k and tag c at position l. The hyperparameter α controls the relative weight of each task in the final loss, and ϕ the degree of focus on difficult examples. The focal loss allows focusing on difficult predictions by putting more weight on token labels predicted with a low probability.

7.3 Whisper-SLU End-to-End

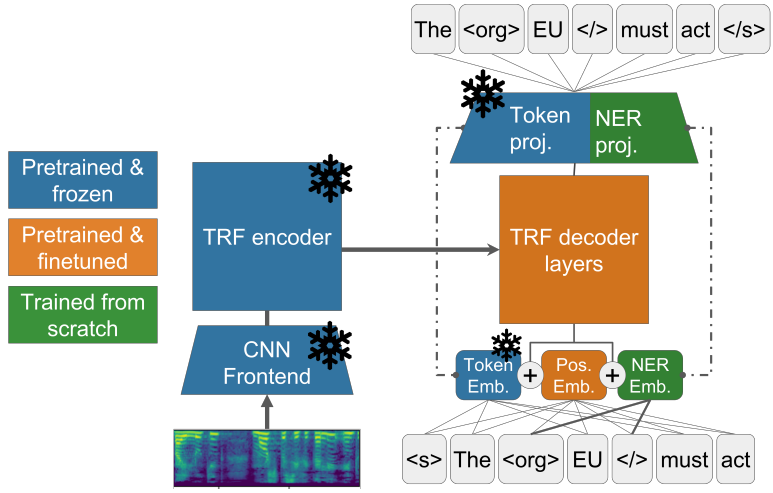


Figure 7.1: Whisper-SLU End2End is an extended version of Whisper that includes additional tokens for NER. The model is trained to predict the named entity tags alongside the text. To mitigate catastrophic forgetting, we dissociate the new NER tokens from the original pretrained token embeddings and freeze the latter.

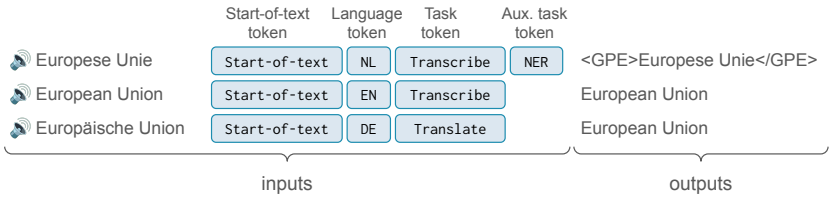


Figure 7.2: Prompting for SpokenNER, transcription and translation. The `<notimestamps>` token has been omitted for clarity.

Whisper-SLU end-to-end extends Whisper's vocabulary with additional named entity tokens (Figure 7.1). These tokens are injected during training to delimit entities (e.g. `<date>` Yesterday `</date>` was `<event>` National Day `</event>`.) We define one token per entity type, a `end-of-entity` token, and an auxiliary token indicating the model to generate NER tags. This adds 20 new tokens to the original vocabulary. The first example in Figure 7.2 shows how the model is prompted to trigger NER token prediction.

The end-to-end model loss is given by:

$$\mathcal{L}^{\text{e2e}} = -\frac{1}{L}\sum_{l=1}^{L}\sum_{k=1}^{K'} y_{k,l}^{\text{asr}} \cdot \log p_{k,l} \tag{7.4}$$

where K' is the amount of tokens in the vocabulary, after adding the additional NER tokens.

This setup is akin to a causal language model with left-to-right attention conditioned on the speech evidence produced by the encoder. In other words, since the named entities are predicted as regular language tokens, when predicting the next token, the model only has access to encoded speech and previously generated tokens. This prevents looking ahead, forcing the model to rely on encoder hidden states for future entity information.

7.4 Experimental Setup

7.4.1 Datasets and Languages

We make use of MSNER [65], a recently released dataset for multilingual spoken NER based on VoxPopuli [102]. It provides silver-quality annotated training and validation sets and a gold-quality annotated test set, respectively 590, 15 and 17 hours of annotated speech (Table 6.1 on page 68). The dataset uses the same 18 entity types as OntoNotes v5 [107]. The four covered languages are German, Spanish, French, and Dutch. We also make use of SLUE-VoxPopuli [86], an additional English-only dataset. This dataset was also derived from VoxPopuli, with gold-quality fine-tuning, development and test sets, although the annotation methodology differs slightly from OntoNotes / MSNER, and conversion from one to the other is nontrivial. We train our models on L1–L5 and compare the performance on SLUE-VoxPopuli with state-of-the-art models (Table 7.1). To investigate further multilingual training, we also train our models on L1–L4 and evaluate them on L1–L5 (Table 7.2). By keeping L5 out of our training set, we want to evaluate the ability of our trained models to generalize to "unseen" languages and to measure catastrophic forgetting. To ensure the reliability of our conclusions, we also train our models on all possible combinations of three languages from L1–L4 and evaluate them on L1–L5 (Figure 7.3).

7.4.2 Evaluation Metrics

Following Meeus et al. [65], we evaluate the NER predictions with the micro-averaged F1-score. The F1-score is the harmonic mean of precision and recall, calculated from an unordered list of named entities predicted for each utterance. Precision is the proportion of correctly predicted entities among all predicted entities, and recall is the proportion of ground truth entities that were correctly detected. An entity is considered to be predicted correctly if both the type and spelling are identical to the ground truth. To allow multiple entities with the same spelling and type in a sentence, we add a unique identifier to each entity/type pair. We recommend using the micro-averaged F1-score because the dataset is unbalanced. The label F1-score only considers the predicted type of the entity for correctness, leaving the transcribed entity out of the computations. This metric ignores spelling mistakes and segmentation errors. We evaluate the ASR performance with the word error rate (WER).

7.4.3 Hyperparameters

The hyperparameters for Whisper-SLU and Whisper-SLU end-to-end are selected by considering the performance on the validation sets. The loss balancing weight α in Equation 7.1 is set to 1 when training the SLU module standalone and 0.2 for multitask fine-tuning. The focal factor ϕ in Equation 7.3 is set to 1. We combine the training sets in different languages by assigning an equal sampling probability for each subset. When a specific language subset is exhausted, its data loader is reset. In other words, examples from smaller datasets have a higher probability of occurrence, but batches are composed of approximately the same number of examples per language. We adapt the gradient accumulation steps to ensure that all models are trained with an effective batch size of 128 examples. The number of training steps is set to 5,000 for all models (about 3 epochs for L1–L4), except for the second training stage of Whisper-SLU fine-tuned for 2,000 additional steps (Section 7.5) and the experience replay models trained, which are adapted to account for the fewer proportion of NER examples per batch (Section 7.6.2). The maximum learning rate was set experimentally at 5e-5 for Whisper-SLU small (train), and for Whisper-SLU End-to-End small. For Whisper-SLU small (fine-tune), the learning rate maxes out at 1e-5, and 1e-6 for the large models. The learning rate follows a cosine schedule with warm-up for 500 steps. In all our experiments, Whisper's encoder remains frozen.

7.5 Training Whisper-SLU

We train Whisper-SLU in two stages: We first train the SLU module only, while keeping all the other parameters frozen. We refer to this model as "Whisper-SLU standalone" (*WSLU (standalone)*). Then, we unfreeze the Whisper-SLU decoder and fine-tune it together with the SLU module for 1,000 additional steps at a lower learning rate. We refer to this model as "Whisper-SLU Multitask Learning" (*WSLU MTL*). We

Table 7.1: Large models comparison on English SLUE-voxpopuli [86]. NLP corresponds to DeBERTa-large [36], best pipeline corresponds to Wav2Vec2-large + LM [6] followed by DeBERTa-large and best end-to-end to Wav2Vec2-large + LM [6]. These three models are the best models for each category, as reported on the SLUE leaderboard.

Model	WER (↓)	F1-score (↑)	Label-F1 (↑)
NLP [86, 36]	-	81.4	85.7
Best pipeline [86, 36, 6]	9.3	71.8	82.2
Best end-to-end [86, 6]	9.3	64.8	73.4
Whisper-SLU MTL (English)	7.3	76.3	88.1
Whisper-SLU E2E (English)	7.7	70.3	78.1
Whisper-SLU MTL (L1–L5)	6.5	78.1	86.9
Whisper-SLU E2E (L1–L5)	6.4	76.1	83.6

adopt this methodology because we want to avoid destabilizing the pretrained model while learning the parameters of the newly initialized SLU module. We first train the model on L5 (SLUE dataset) and on L1–L5 and report the model performance on L5 in Table 7.1. We observe a small advantage of training on multilingual data compared to English-only data, both with a lower word error rate and higher F1-score. A slightly lower label F1-score is explained by the small differences in annotation methodology between L1–L4 and L5 (see Meeus et al. [65] for more details). Nonetheless, the small bias emerging from these is compensated by the gains from multiple languages.

We also train the small version of our model on L1–L4 and evaluate it on L1–L5 to explore the limitations of the proposed approach. For this experiment, we report the performance for the two stages in Table 7.2. Even though the model is not updated to handle the specifics of the fine-tuning dataset, we already reach a decent SLU performance averaging above 50% F1-score. Multitask fine-tuning has the effect of considerably decreasing the word error rate on the languages observed during training and improves the F1-score by 6.3% on average. However, this has a disastrous effect on languages not observed during training, namely the WER on English increases from 8.8% to 32.9%, while the performance in F1-score decreases by almost 6% (Table 7.2). This loss of performance in NER seems to be entirely due to the drop of performance in WER, since the Label-F1, which does not take into account transcription errors but only detection errors, increases by 7.8%. Also in Figure 7.3, the same behaviour can be observed, with observed WER on out-of-domain languages well above 100%. This unfortunate behaviour is primarily explained by the fact that a large proportion of the tokens in the vocabulary is not observed in the training set. As the model adapts to the new task, it will learn to assign a lower output probability to those unobserved tokens, which is reflected in the (tied) token embeddings. To show this, we reset the token embeddings (and thus the output projection) to their pretrained value, while keeping all the other parameters at their value post-fine-tuning, and reevaluate the model (*WSLU MTL +reset*). Although we observe small increases in WER for L1–L4,

the ASR performance in English comes back to more reasonable levels. The F1-score also drops slightly for L1–L3 but increases for Dutch and for English, which suggests that the transformer decoder layers learn to identify patterns that transfer across languages. Note that training on L1–L4 seemingly leads to less drastic forgetting on L5 (Table 7.2) than training on 3 languages from L1–L4 (Figure 7.3).

To get a sense of how this model compares to a pipeline model, we train XLM-RoBERTa base [18] to predict BIO-tags from MSNER transcriptions, and evaluate the resulting model on Whisper small's predictions. Although XLM contains significantly more parameters than the NER module in Whisper-SLU, results are not that far apart and the fine-tuned Whisper-SLU performs similarly to the pipeline model. Where the latter really stands out is its ability to transfer knowledge to English, as witnessed by the NER performance on this language.

Table 7.2: Small models comparison: all models are trained on L1–L4 and evaluated on L1–L5. The WER on L5 (English) shows the forgetting for the transcription task from out-of-domain languages.

Experiment	WER (↓)					F1-score (↑)					Label-F1 (↑)				
	DE	ES	FR	NL	EN	DE	ES	FR	NL	EN	DE	ES	FR	NL	EN
Whisper+XLM-RoBERTa	16.1	11.6	13.6	21.8	8.8	59.8	56.9	54.4	57.4	47.8	85.9	82.9	78.2	83.4	68.4
WSLU (standalone)	16.1	11.6	13.6	21.8	8.8	53.7	48.4	51.2	51.5	33.6	78.5	78.9	79.1	81.7	52.6
WSLU MTL	12.9	9.6	12.1	14.6	32.9	59.1	56.9	56.9	57.2	27.9	80.5	82.5	80.8	85.3	60.4
WSLU MTL (+reset)	13.3	9.9	12.9	16.0	10.3	57.7	55.4	54.6	62.4	41.9	80.2	81.5	79.8	84.3	62.6
WSLU-e2e	12.1	9.5	11.1	15.3	86.2	61.5	57.6	58.7	64.3	23.2	81.7	82.3	83.1	84.4	8.5
WSLU-e2e (+reset)	14.4	10.3	13.1	17.2	11.8	54.9	53.6	53.3	57.7	40.2	77.5	81.4	80.3	81.4	63.1
WSLU-e2e (+DE)	12.2	9.5	11.3	15.0	18.5	60.0	57.2	57.7	65.6	27.6	81.1	82.7	82.1	86.2	54.8
WSLU-e2e (+DE +ER)	12.6	9.5	11.4	15.6	10.7	61.1	56.3	58.4	65.4	22.4	81.9	82.4	83.0	85.1	40.2

7.6 Training Whisper-SLU End-to-End

Once again, we first report the results of the large end-to-end model trained on L5 and L1–L5 in Table 7.1. Here too, there is an advantage from multilingual training compared to English-only dataset. To investigate the effects of fine-tuning, let us look at the small end-to-end model's results when trained on L1–L4 (*WSLU-e2e* in Table 7.2). Although we expect good performance on in-domain languages, we also suspect catastrophic forgetting of pretraining tasks, as for Whisper-SLU. Indeed, in Figure 7.3, two distinct clusters correspond to in-domain and out-of-domain languages. Training the end-to-end models on new data leads to considerable drops of performance in some tasks, such as the ability to transcribe speech in languages not observed in the fine-tuning set, or translate (X → en), even though the pretrained model reports good performance on both tasks prior to fine-tuning (WER=8.8% on English subset, increases to 86.2% after fine-tuning on other languages). To check whether the translation task was subject to forgetting, consider the following Dutch sentence: "Nog dramatischer is de situatie in Libanon", which translates to "The situation in Lebanon is even more dramatic". After fine-tuning, and given the prompt `<sot><NL><translate>`, the model outputs the following sentence in Dutch, where English was expected: "Nog dramatischer oog de situatie en Libanon". Similar examples can be observed in other

languages, where the model transcribes instead of translating. This suggests that the embedding vector corresponding to the `<translate>` token got disrupted because it was not observed during fine-tuning. We try once again to reset the token embeddings to their pretrained value (*WSLU-e2e +reset*). However, for obvious reasons, we leave the embeddings corresponding to the newly introduced NER tokens to their fine-tuned value. Although this inevitably introduces some distortion because the Transformer weights are optimized for slightly different embedding values, this simple trick allows to limit the increase of WER for the transcription task in English to +3.0% after resetting the embeddings (Table 7.2). Note that it involves a small increase in WER in L1–L4 languages averaging +1.1% across the board. A similar observation is made on the NER F1 score and the label-F1 score: resetting the embeddings lowers the L1–L4 F1 score by 5.2% on average (-2.8% label-F1) while increasing the performance of L5 F1 score by 16.9% to 40.2%.

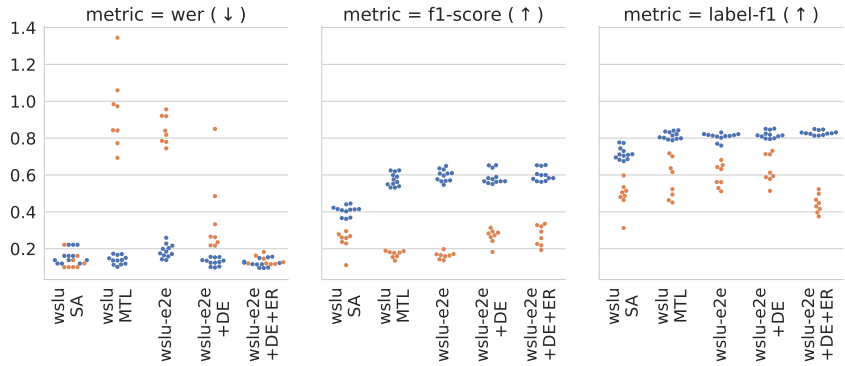


Figure 7.3: Comparison of Whisper-SLU and Whisper-SLU End2End small models. The graph shows the value of the three evaluation metrics on the test sets L1–L5 for a model trained on 3 languages out of L1–L4. For example, the score on L4 of a model trained on L1–L3 will be coloured **orange**, while the scores on L1 and L5 for the same model will be coloured **blue**.

7.6.1 Dissociated Token Embeddings

Encouraged by this finding, we propose to dissociate the embedding matrix into two distinct matrices: one that contains the original pretrained embeddings, and one containing the newly introduced NER token embeddings (*WSLU-e2e (+DE)*). The latter will be updated normally during fine-tuning while we freeze the former to avoid any undesired update to the original token embeddings. Although this seems to have the desired effect on the transcription task, the model looses the ability to translate (X $\rightarrow$ en). We suspect that it is not actual forgetting that is happening but rather the `<translate>` token given as a prompt looses its purpose as it is not observed during training. As a result, even though the model is still able to translate speech, it is

not possible to trigger this behavior anymore through any combination of the special tokens.

7.6.2 Experience Replay

To solve the issue mentioned in the previous paragraph, we train the model with both examples from the pretraining tasks, and examples from the fine-tuning dataset (*WSLU-e2e +DE+ER*). However, since we do not have access to the pretraining data, we create an artificial ER dataset[1] by using the pretrained model to generate the targets for VoxPopuli training sets. In practice, for the 10 most common languages in VoxPopuli, we transcribe (X $\rightarrow$ X) and translate (X $\rightarrow$ en) the 12,000 first examples. We create our training set by interleaving examples from MSNER with examples from the ER dataset with a certain probability P. A batch with B examples will thus be composed with approximately $B \times (1 - P)$ examples from MSNER and $B \times P$ examples from the ER dataset. Unlike our other experiments, we set the number of training steps to $5000/(1 - P)$, a necessary adjustment to account for the fewer NER examples per batch. We found that the probability P has only a limited effect on the performance (0.4% WER improvement (L5) when P is increased from 10% to 60%, and no apparent effect on the F1-score). With the adaptive number of training steps, the performance on L1–L4 remains unchanged.

Training the models with this strategy gives the best WER on English among end-to-end models, albeit at the expense of a slight WER increase of L1–L4. It also seems to solve the issue highlighted in the previous paragraph, and utterances can be correctly translated to English. The effect on SLU performance is comparable for L1–L4 but worse than the other approaches for English. Indeed, contrarily to all other methods, the model is trained with English examples which are not annotated for NER. As a result, the model likely learns to emit NER tokens for L1–L4 only, and not L5. A similar observation is made when the special token `<translate>` is used in combination with the special token `<ner>`, the model fails to perform the expected task, that is, to produce an annotated speech translation. This is because this particular combination of prompt tokens was never observed during training. Instead, the model predicts the annotated speech transcription. Since this prefix combination was never seen during training, the model associates the `<ner>` token with the transcription task, and it seems impossible to trigger the annotations in translate mode. One exception to this is when the source language is English. In this case, the entity annotation task is done correctly. One way to solve this issue would be to provide the model with relevant training examples, that is utterances with paired annotated translation. However, this is out of the scope of this thesis.

[1] Available here: hf.co/datasets/qmeeus/vp-er-10l

7.7 Conclusion

This chapter explored the application of Transformers for Multilingual Spoken Named Entity Recognition. We investigated two architectures: Whisper-SLU and Whisper-SLU end-to-end, both leveraging the strengths of Transformers for spoken language tasks. Although both models were subject to catastrophic forgetting, Whisper-SLU offers more control because the NER component is distinct from the rest of the model. It can thus be partially frozen and trained in standalone mode. Whisper-SLU end-to-end has the advantage of performing ASR and NER at the same time, while having fewer parameters. However, it cannot benefit from the bidirectional attention mechanism and must predict the annotated transcription from the encoded speech and previous tokens only.

We have shown that a single end-to-end model trained on multiple languages outperforms a fine-tuned pipeline model on the same dataset. This result highlights the potential of multilingual Spoken NER systems, especially in scenarios with limited resources. Multilingual models offer several key advantages: they leverage shared knowledge across languages, improving their understanding of common patterns and linguistic representations. This shared knowledge also enhances generalization, making the models more robust to variations in pronunciation, accent, and dialect. From an efficiency perspective, maintaining one multilingual model is more cost-effective than handling multiple monolingual models. Additionally, multilingual training implicitly acts as data augmentation, exposing the model to a wider range of linguistic phenomena, which can benefit individual language performance.

Despite these advantages, multilingual Spoken NER presents challenges. Larger datasets are often required to train such models effectively, and data imbalances across languages can introduce biases. Furthermore, handling language-specific complexities remains a challenge. Nevertheless, the benefits of multilingual Spoken NER often outweigh these challenges, particularly in low-resource scenarios. This approach holds significant promise for advancing the field of speech processing and enabling more inclusive and accessible applications across diverse linguistic communities.

A significant challenge we encountered was catastrophic forgetting, the phenomenon where a model loses its ability to perform previously learned tasks as it adapts to new ones. In our case, the models were prone to forgetting how to transcribe languages not included in the fine-tuning dataset or losing their ability to translate into English.

To mitigate catastrophic forgetting, we explored three primary strategies:

- Embedding Reset: This involved resetting the token embeddings to their pretrained values after fine-tuning. This surprisingly simple technique proved effective in preventing the model from forgetting the transcription task.
- Disassembling Embeddings: In this approach, we separated the pretrained token embeddings from the NER-specific ones, hoping to preserve the model's ability to perform both tasks. While this strategy maintained most English

transcription capabilities, it caused the model to lose its translation ability, as the token indicating the "translate" task became meaningless during fine-tuning.

- Experience Replay (ER): This strategy involved using the pretrained model to generate additional training data for fine-tuning. This helped the model retain previously learned tasks and languages, provided the ER dataset contained relevant examples. However, the model's inability to generate annotated translations, a task it wasn't explicitly trained for, proved to be a limitation.

Our findings offer valuable insights for the development and maintenance of robust Spoken NER models. As more and more foundation models are released to the public, it is important to understand how these models can be adapted, modified, and fine-tuned while conserving their original capabilities. Future research directions include addressing the highlighted limitation and enable Whisper-SLU to generate annotated translations, improving the proposed mitigation methods, and extending Whisper-SLU to other language understanding tasks.

Chapter 8

Conclusion

This chapter concludes the thesis. We first answer the research questions introduced in Section 1.1, then we summarize our contributions and point to some exciting directions for future research building on our work.

8.1 Answering the Research Questions

8.1.1 How can Transformer architectures be effectively pretrained to capture robust speech representations suitable for downstream SLU tasks?

In Chapter 3, we have explored the potential of Bidirectional Transformers for learning and generating expressive representations of speech. We proposed to use a Transformer Encoder-Decoder with a bidirectional decoder. By training the model to reconstruct masked portions of text given a speech input, then evaluating the learned representations offline, we have found that they capture essential linguistic information crucial for downstream SLU tasks. During inference, we use a CTC submodule to generate a template, i.e. a list of masked tokens of the same length as the greedy CTC prediction, that is iteratively filled and refined by our bidirectional decoder.

This approach effectively bridges the gap between speech and text, enabling the use of large unlabeled speech corpora for pretraining, thereby mitigating the data scarcity issue prevalent in SLU. Furthermore, our findings reveal that these pretrained representations can be successfully utilized to train smaller, task-specific SLU models with limited annotated data, requiring very few parameters and computational resources to achieve this goal. The masked language modeling objective used to train the decoder leads to more expressive representations than a decoder with

autoregressive attention, which confirmed our hypothesis that bidirectional attention is essential to representation learning. Finally, we have proposed to use class attention for our SLU classifier, an innovative architecture that identifies the elements in the input sequence that are relevant to make a prediction in the context of a classification task. This layer produces attention weights that can be used to identify potential issues and explain the predictions of the model.

We have identified three limitations to our proposed approach. Firstly, the quality of the representation depends on the performance of the CTC submodule, in particular since the output of the CTC directly impacts the length of the representation sequence. Secondly, because we use the model as a representation model without further fine-tuning, the expressiveness of the representations is directly correlated with the diversity of the pretraining dataset. For this reason, it might not always be possible to discriminate between concepts that are semantically close to one another. This limitation can only be solved by using an appropriate pretraining set, which is difficult since we do not always know in advance the specificities of the downstream dataset, or by fine-tuning the representation model on the downstream task, which defeats the purpose of this research. Finally, the CTC objective constrains the encoder to represent speech in a way that is compatible with the subword tokens and consequently discard any information that is not directly useful for predicting those output units. Although this proved sufficient in the context of this research, we could easily imagine scenarios where it would fail to encode relevant information useful for the downstream task. For example, such a system is likely to fail if the machine learning task was to detect sarcasm or emotion.

8.1.2 To what extent does multitask learning, combining spoken language understanding with speech recognition, improve performance in low-resource scenarios?

Our exploration of low resource SLU scenarios continued through multitask learning in Chapter 4, which proved to enhance SLU performance when combined with ASR, particularly in low-resource SLU settings. By jointly training our bidirectional representation model presented in Chapter 3 on both speech recognition and a chosen SLU task (intent recognition or sentiment classification), we observed that the model was able to leverage complementary information from both tasks, leading to improved performance on the low resource SLU task, as the model can transfer knowledge from the speech recognition task to compensate for the data scarcity. Our experiments on the Grabo and Patience datasets highlight that these representations generated by the decoder layers outperform representations derived from the encoder alone, especially when trained simultaneously on multiple tasks. This further confirms the hypothesis that the decoder plays the role of an implicit language model, leading to linguistic representations that prove to be considerably more expressive than the embeddings produced by the encoder.

Our results on the Fluent Speech Commands dataset, where our model surpasses the

current state-of-the-art especially in low-resource scenarios, underscores the potential of our approach for practical applications. The proposed class attention mechanism not only contributes to model efficiency but also provides valuable insights into the prediction decision-making of the model. Overall, this research demonstrates the feasibility and efficacy of combining our bidirectional speech representation model with multitask learning, and we have achieved significant improvements in intent recognition performance, particularly in challenging low-resource environments.

8.1.3 How can pretrained speech-to-text Transformers be successfully adapted to SLU tasks while keeping low data requirements and computational footprint for the downstream tasks, and while retaining previously acquired knowledge?

In Chapter 4, the SLU tasks explored in this research remain relatively simple. In Chapter 5, we extended this research to more challenging tasks like slot filling and named entity recognition, while keeping the low resource requirement at the center of our analysis. To address these more complicated tasks, we proposed to adapt an established pretrained model, OpenAI's Whisper.

We have investigated techniques like adding task-specific decoder modules, freezing certain model parameters, and pruning embedding layers to maintain a small model footprint and reduce data requirements. This enables the efficient deployment of SLU models with low computational requirements while still achieving remarkable performance. We have demonstrated how we can obtain very good models with minimal training by modifying and repurposing the embeddings to fit a new task, rather than training them from scratch. We have also highlighted the importance of designing specific output structure and decoding strategies to account for the specifics of downstream datasets, and their advantage to reach good performance while keeping a low computational footprint. In Chapters 3 and 4, we have used a simple algorithm to find the discrete output sequence that minimizes the cross-entropy with the continuous predictions generated by the model. In Chapters 5 to 7, we have applied a similar approach, essentially preventing illegal assignments by tweaking the parameters of a Markov model and using the Viterbi algorithm to decode the model logits into valid output sequences.

During our exploration of multilingual Spoken Named Entity Recognition (Chapter 7), we encountered a significant challenge: catastrophic forgetting. This phenomenon, common in machine learning, occurs when a model loses its ability to perform previously learned tasks as it adapts to new ones. In our case, this meant the models struggled to transcribe languages not included in the fine-tuning dataset or lost their ability to translate into English. To mitigate this, we first employed embedding reset, which proved surprisingly effective in preserving the transcription task given the concept simplicity. However, while this approach maintained English transcription capabilities, it did not solve the model forgetting its translation ability. We formalized this approach

during training by dissociating pretrained and NER-specific embeddings, which prove to be effective in learning the NER task while preserving transcription capabilities. Finally, we experimented with experience replay (ER), using the pretrained model to generate additional training data for fine-tuning. This approach successfully retained previously learned tasks and languages, provided the ER dataset contained relevant examples. However, the model's inability to generate annotated translations remained a limitation. Finally, there are additional costs associated with ER such as the computational cost of producing the examples and replaying them during training, and the associated extra storage requirements.

8.1.4 How can we use text-based NLU models to build training datasets for multilingual SLU research?

Having identified the need for multilingual SLU datasets to advance research in this domain, we have built a multilingual Spoken NER dataset, named MSNER, that we have presented in Chapter 6, with the goal of researching the potential for machine-made annotated data to train multilingual SLU models. We have also created human-made annotated test sets to provide researchers with gold-quality annotations for evaluation purposes.

In Chapter 7, we showed that an end-to-end SLU model trained on machine-generated data could yield a better performance than a pipeline model, which validated our hypothesis that silver-quality datasets had their use in advancing SLU research. We know first-hand how difficult and time-consuming it can be to generate human-made annotations, and when data is scarce, it might be opportune to rely on pretrained models to generate artificial data. Nonetheless, we also highlighted potential pitfalls like model overconfidence and varying reliability across entity types, leading to potential biases in the resulting silver datasets. Knowing this, researchers creating such datasets for training might need to correct for these biases either manually or programmatically.

Here's how text-based NLU models can be effectively used:

- Silver Data Generation: Use high-performing text-based NER models to annotate large amounts of speech transcripts. This can be particularly beneficial for low-resource languages where manually annotated data is scarce.
- Uncertainty Filtering: The reliability of the model's confidence can be assessed by evaluating it on a small representative gold-standard dataset. From there, we can determine the confidence reliability on a per-class basis, and use the results to recalibrate the model. We can also use the recomputed scores to filter out examples with low confidence.
- Targeted Manual Annotation: Prioritize manual annotation of examples where the model exhibits low confidence or frequently confuses specific entity types. This focuses human effort on areas where the model needs the most improvement.

- Error Analysis: Thoroughly analyze the errors made by the text-based model on the speech data. This can reveal patterns of misclassifications or common issues that need to be addressed during model training.

By combining the efficiency of automated annotation with targeted human intervention, we can create high-quality training datasets that accelerate the development of more accurate and robust multilingual SLU models.

8.1.5 What are the benefits of multilingual Spoken NER systems compared to monolingual Spoken NER systems?

In Chapter 7, we have demonstrated that a single end-to-end model trained on multiple languages (English and others) can outperform a pipeline model fine-tuned on the same dataset. This suggests that multilingual Spoken NER systems offer advantages, particularly in low-resource scenarios. Among the key benefits of multilingual Spoken NER, we can cite:

- Shared Knowledge Transfer: Multilingual models can leverage shared knowledge across languages, learning common patterns and representations that enhance performance on all languages.
- Improved Generalization: By training on a diverse set of languages, the model becomes more robust to variations in pronunciation, accent, and dialect, resulting in better generalization.
- Efficiency: A single multilingual model is more efficient than training and maintaining separate models for each language, reducing computational costs and resource requirements.
- Data Augmentation: Multilingual training acts as implicit data augmentation, exposing the model to a wider range of linguistic phenomena, which can benefit performance on individual languages.

However, multilingual Spoken NER also presents challenges, including the need for larger datasets, potential biases due to data imbalance, and difficulties in handling language-specific complexities. Overall, the benefits of multilingual Spoken NER often outweigh the challenges, especially when data is limited. These models hold the potential to significantly advance the field of speech processing, enabling more inclusive and accessible applications for diverse language communities.

8.2 Summary of our Contributions and Directions for Future Research

8.2.1 Speech Representations Learning

We have proposed two methods for building representations of speech to perform SLU in low resource scenarios. In Chapter 3, we have presented an approach leveraging bidirectional attention and a masked language modeling objective, by iteratively refining a template provided by a greedy CTC module. The representations, used offline to train multiple lightweight downstream SLU models, outperformed other representations like those obtained with causal attention, thus validating our approach compared to autoregressive models. We addressed limitations arising from domain shift between the pretraining and fine-tuning datasets with multitask learning. In Chapter 4, we saw that the representation model could be fine-tuned with very limited training data by adopting a multitask learning approach combining SLU and ASR tasks in a single objective. This allows the model to learn specialized representations tuned for the specific downstream dataset under low resource constraints. For both approaches, we have presented experiments using varying amounts of training data to show the robustness of the representations to low resource scenarios for the task of intent recognition, both in Dutch and in English.

In our exploration of speech representations, we have used a masked language modeling objective to build bidirectional representations. This proved to be a challenge for more complicated datasets, like the Patience corpus, where it might be beneficial to keep a memory of previous utterances from the same gaming session. To incorporate such considerations, future research could look at context-aware models, where a small memory of previous utterances is stored as a context token. At each utterance or conversation turn, the model makes a prediction, taking into account the conversation history. Much like BERT and GPT, the learned context can be used in complex language understanding tasks. In multitask fine-tuning, we have not explored the interactions of more than two tasks at once. It would be interesting to evaluate the impact of adding more SLU tasks to the model.

8.2.2 Low Resource Spoken Language Understanding

In Chapter 3, we introduced a novel architecture for spoken utterance classification in the form of class attention transformer. This architecture computes attention weights for a `CLS` token given the input sequence. This architecture was shown to be particularly well-suited for low resource settings compared to a transformer encoder. Furthermore, it can explain its own predictions by means of visualizations, something that is notoriously difficult with deep neural networks. We have shown that, given the appropriate output dimensions, the resulting weights indicate which positions are driving the prediction, given the class. In Chapter 3, we have also proposed to use the focal loss for various classification problems in SLU, showing its benefits

for classification tasks with unbalanced classes and difficult examples. We validated experimentally the class attention transformer and the focal loss for low resource SLU.

Finally, we highlighted the benefits of integrating specific output structures like Markov models, and specific decoding algorithms that find the most probable output given the model prediction and the set of rules constraining the output structure. This demonstrated to be beneficial for intent recognition (Chapters 3 and 4) as well as for slot filling (Chapter 5) and Spoken-NER tasks (Chapters 5 to 7). When dealing with small datasets, carefully designing the architecture by incorporating these examples allows training the model with fewer training examples.

Nowadays, deep neural network sizes are growing rapidly, and the larger models require expensive graphic cards and computers to run. Nonetheless, we still need to develop models for specific use cases, performing unique tasks, and running on affordable equipments. To do this, it is important that future research continues looking at how to develop models that are both simple and efficient.

8.2.3 Pretrained speech-to-text Model Adaptation

Going beyond simply fine-tuning, we proposed to modify the pretrained models by pruning the token embeddings to adapt the original model to a new task while reusing as much of the pretrained embeddings as possible. In practice, we showed that when dealing with a new task, it is possible to build a new embedding matrix from pretrained embedding vectors by selecting appropriate pretrained embedding vectors for the new token vocabulary. We have also studied the effects of freezing or fine-tuning layers, showing that in low resource scenarios, it is often more appropriate to freeze attention layers to avoid issues like overfitting and poor generalization. In the context of multilingual Spoken NER, we have tackled catastrophic forgetting of pretrained tasks and languages by proposing methods to mitigate the greedy overwriting of token embeddings. More specifically, we proposed to reset the token embeddings after training, or to dissociate the pretrained and the task-specific embeddings during training to allow the model to learn the new task, while protecting the original embeddings. We have also explored experience replay as a means to avoid performance degradation on pretraining tasks. To work around the absence of pretraining data, we proposed to use the pretrained model to generate an experience replay dataset. We have presented experiments with varying amounts of training data and in different languages.

The latest breakthroughs in artificial intelligence open up opportunities for using large models in the field of SLU. Parameter efficient fine-tuning has become an attractive line of research. One example uses low rank adapter, a method that adds so-called low rank adapters between the model's layers, of much smaller dimension than the layers [43, 112]. The pretrained model is frozen, and only the adapters are trained. This method and the works building on this have proved to be worth exploring, succeeding in fine-tuning models with little data and small computational footprint. Another example is prompting. A recent study has shown that prompting a generative

spoken language model can be used to solve multiple SLU tasks with fewer trainable parameters than fine-tuning methods [13, 14]. Going further, UniverSLU combines natural language instructions fine-tuning with multitask learning to equip Whisper with zero-shot SLU capabilities [4]. These lines of research offer promising methods to adapt large pretrained models while keeping a relatively low carbon footprint. However, the model sizes after fine-tuning are still the same. This is why we think that it is also worth researching methods aiming at decreasing the model sizes, like pruning embeddings, attention heads or layers.

In Chapter 6, we made use of a model to provide us with pseudo-labels. However, we could have gone much further and used the NLP model as a teacher, producing targets to train our model. One advantage of this approach: the targets are soft-labels, and contain information about probability distribution and uncertainty toward the predictions. Although there is nothing new with this training strategy, known as knowledge distillation, few are applying it to speech models to improve, for example, the intrinsic language model in the transformer decoder. However, when training data is scarce or expensive to produce, or when devices are constrained, this strategy can be used to train smaller models with few or no labelled training data.

8.2.4 Multilingual Spoken NER Dataset

In Chapter 6, we have presented a new dataset to advance the field of multilingual SLU. MSNER is an annotated version of the larger VoxPopuli [102], with named entities for utterances in four languages, Dutch, French, German, and Spanish, following the 18-classes OntoNotes label set [107]. The training and validation sets are silver-quality annotations, generated by a text-based NER model, and the evaluation sets are gold-quality annotations, entirely reviewed by a human annotator. A thorough analysis of the dataset was realized and indications were made towards the intended use of the datasets for training purposes, as well as potential points of caution. The dataset was distributed open source, as the annotation tool. We look forward to seeing models trained on this dataset, improving the results that we have published on the evaluation sets.

8.2.5 Multilingual SLU

In Chapter 7, the MSNER dataset gave us the opportunity to explore spoken NER in multiple languages, and compare the performance of models trained with a monolingual dataset or with multiple languages. We showed that the model trained with the multilingual dataset performed better than the monolingual model in this particular language. This showed the benefits of multilingual fine-tuning, as the model can learn patterns in other languages and apply them to the target language. This has implications in low resource languages, suggesting that in cases when few training examples exist in one language, the training set can be augmented with examples

in different languages, potentially improving the performance in all the languages included in the training set.

In our experiments with MSNER, we have not used the uncertainty predicted by the text-based model, and available in the dataset, to fix potential issues. It would be interesting to integrate this number as an additional variable during training, for example to reweigh examples in the loss based on the model's uncertainty and see whether such information could be useful to mitigate error cascading from the NLU model to the SLU model. Another promising approach is through knowledge distillation, by adding a term to the loss that encourages the output probability distribution to match the distribution predicted by the NLU model.

Regarding catastrophic forgetting, we have proposed methods to mitigate the performance degradation due to language forgetting and task forgetting. However, we have not explored in depth the inability of the model to learn a new task without training examples, as was the case with the `<translate><ner>` token combination. The model having never observed this particular combination of tokens, it would respond by transcribing and annotating the utterance rather than translating and annotating. We have emitted the hypothesis that by including a few annotated and translated training examples, we could steer the model towards learning this task in one or a few shots. However, this remains to be proved and is a good starting point for future research.

8.2.6 Contributions to the Open Source Community

In the making of this thesis, contributions to multiple open source libraries were made in the form of new features development, issue reports and bug fixes, etc. Additionally, various pretrained models were released on the HuggingFace platform[1]: for Dutch ASR, a small and a large model, pretrained on CGN, and for multilingual spoken NER, Whisper-SLU and Whisper-SLU end-to-end with different training configurations. The pieces of code and repositories created for the models and experiments presented in the previous chapters are distributed on GitHub[2], free to use for both commercial and non-commercial reasons, following the guidelines of the open source definition[3].

[1] hf.co/qmeeus
[2] github.com/qmeeus
[3] opensource.org/osd

Bibliography

[1] B. Agrawal, M. Müller, S. Choudhary, et al. "Tie Your Embeddings Down: Cross-Modal Latent Spaces for End-To-End Spoken Language Understanding". In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. 2022, pp. 7802–7806. DOI: 10.1109/ICASSP43922.2022.9747759. arXiv: 2011.09044 (p. 31).

[2] R. Alec, W. Jeffrey, C. Rewon, et al. "Language Models are Unsupervised Multitask Learners". In: *OpenAI Blog* 1.8 (2019), p. 9. Blog post (pp. 16, 17).

[3] S. Arora, S. Dalmia, P. Denisov, et al. "ESPnet-SLU: Advancing Spoken Language Understanding Through ESPnet". In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* (2022), pp. 7167–7171. DOI: 10.1109/ICASSP43922.2022.9747674. arXiv: 2111.14706 (pp. 30, 54, 63).

[4] S. Arora, H. Futami, J.-w. Jung, et al. "UniverSLU: Universal Spoken Language Understanding for Diverse Tasks with Natural Language Instructions". In: *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL–HLT)*. Ed. by K. Duh, H. Gomez, and S. Bethard. Mexico City, Mexico: Association for Computational Linguistics, 2024, pp. 2754–2774. DOI: 10.18653/v1/2024.naacl-long.151. ACL: 2022.findings-naacl.60 (p. 100).

[5] S. Arora, A. Ostapenko, V. Viswanathan, et al. "Rethinking end-to-end evaluation of decomposable tasks: A case study on spoken language understanding". In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. Vol. 5. 2021, pp. 3881–3885. DOI: 10.21437/Interspeech.2021-1537. arXiv: 2106.15065 (pp. 31, 41, 42, 60, 61, 64).

[6] A. Baevski, H. Zhou, A. Mohamed, and M. Auli. "wav2vec 2.0: A framework for self-supervised learning of speech representations". In: *Advances in Neural Information Processing Systems*. Curran Associates Inc., 2020. DOI: 10.5555/3495724.3496768. arXiv: 2006.11477 (pp. 2, 50, 54, 58, 86).

[7] Y. Bai, J. Yi, J. Tao, et al. "Fast End-to-End Speech Recognition Via Non-Autoregressive Models and Cross-Modal Knowledge Transferring From BERT". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), pp. 1897–1911. DOI: 10.1109/TASLP.2021.3082299. arXiv: 2102.07594 (p. 34).

[8] J. Baxter. "A model of inductive bias learning". In: *Journal of Artificial Intelligence Research* 12.1 (Mar. 2000), pp. 149–198. DOI: 10.5555/1622248.1622254. arXiv: 1106.0245 (pp. 3, 12).

[9] D. Benikova, C. Biemann, and M. Reznicek. "NoSta-D named entity annotation for German: Guidelines and dataset". In: *Proceedings of the 9th International Conference on Language Resources and Evaluation, LREC 2014*. Reykjavik, Iceland: European Language Resources Association (ELRA), May 2014, pp. 2524–2531. ACL: L14-1251 (p. 75).

[10] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 1st ed. Springer, 2006. DOI: 10.5555/1162264. Online book (p. 8).

[11] T. B. Brown, B. Mann, N. Ryder, et al. "Language models are few-shot learners". In: *Advances in Neural Information Processing Systems*. 2020. DOI: 10.5555/3495724.3495883. arXiv: 2005.14165 (pp. 16, 17).

[12] W. Chan, N. Jaitly, Q. Le, and O. Vinyals. "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016, pp. 4960–4964. DOI: 10.1109/ICASSP.2016.7472621. IEEE: 7472621 (p. 24).

[13] K.-W. Chang, W.-C. Tseng, S.-W. Li, and H.-y. Lee. "An Exploration of Prompt Tuning on Generative Spoken Language Model for Speech Processing Tasks". In: *Proc. Interspeech 2022*. 2022, pp. 5005–5009. DOI: 10.21437/Interspeech.2022-10610. arXiv: 2303.00733 [eess.AS] (p. 100).

[14] K.-W. Chang, Y.-K. Wang, H. Shen, et al. "SpeechPrompt v2: Prompt Tuning for Speech Classification Tasks". In: (2023). arXiv: 2303.00733 [eess.AS] (p. 100).

[15] N. Chen, P. Żelasko, L. Moro-Velázquez, J. Villalba, and N. Dehak. "Align-Denoise: Single-Pass Non-Autoregressive Speech Recognition". In: *Proc. Interspeech 2021*. 2021, pp. 3770–3774. DOI: 10.21437/Interspeech.2021-1906 (p. 34).

[16] E. A. Chi, J. Salazar, and K. Kirchhoff. "Align-Refine: Non-Autoregressive Speech Recognition via Iterative Realignment". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Ed. by K. Toutanova, A. Rumshisky, L. Zettlemoyer, et al. Association for Computational Linguistics, June 2021, pp. 1920–1927. DOI: 10.18653/v1/2021.naacl-main.154 (p. 34).

[17] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. "Attention-Based Models for Speech Recognition". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc., 2015. DOI: 10.5555/2969239.2969304. arXiv: 1506.07503 (p. 24).

[18] A. Conneau, K. Khandelwal, N. Goyal, et al. "Unsupervised cross-lingual representation learning at scale". In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics* abs/1911.0 (2020), pp. 8440–8451. DOI: 10.18653/v1/2020.acl-main.747. ACL: 2020.acl-main.747 (pp. 4, 70, 87).

[19] G. Crichton, S. Pyysalo, B. Chiu, and A. Korhonen. "A neural network multi-task learning approach to biomedical named entity recognition". In: *BMC Bioinformatics* 18.1 (2017). DOI: 10.1186/s12859-017-1776-8. PMID: 28810903 (p. 18).

[20] Z. Dai, Z. Yang, Y. Yang, et al. "Transformer-XL: Attentive Language Models beyond a Fixed-Length Context". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Ed. by A. Korhonen, D. Traum, and L. Màrquez. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 2978–2988. DOI: 10.18653/v1/P19-1285. arXiv: 1901.02860 (p. 15).

[21] R. De Mori. "History of Knowledge and Processes for Spoken Language Understanding". In: *Spoken Language Understanding*. John Wiley & Sons, Ltd, 2011. Chap. 2, pp. 9–40. DOI: https://doi.org/10.1002/9781119992691.ch2. Google Books: RDLyT2FythgC (p. 27).

[22] K. Deng, Z. Yang, S. Watanabe, et al. "Improving Non-Autoregressive End-to-End Speech Recognition with Pre-Trained Acoustic and Language Models". In: *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2022, pp. 8522–8526. DOI: 10.1109/ICASSP43922.2022.9746316. arXiv: 2201.10103 (p. 34).

[23] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of deep bidirectional transformers for language understanding". In: *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*. Vol. 1. ACL, June 2019, pp. 4171–4186. DOI: 10.18653/V1/N19-1423. arXiv: 1810.04805 (pp. 4, 16–19, 30, 37).

[24] J. Dong, J. Fu, P. Zhou, H. Li, and X. Wang. "Improving Spoken Language Understanding with Cross-Modal Contrastive Learning". In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. 2022, pp. 2693–2697. DOI: 10.21437/Interspeech.2022-658 (pp. 30, 62).

[25] L. Dong, S. Xu, and B. Xu. "Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition". In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. 2018, pp. 5884–5888. DOI: 10.1109/ICASSP.2018.8462506. IEEE: 8462506 (p. 24).

[26] A. Fan, E. Grave, and A. Joulin. "Reducing Transformer Depth on Demand with Structured Dropout". In: *Computing Research Repository (CoRR)* cs.CL/1909.11556 (2019). arXiv: 1909.11556 (p. 9).

[27] Y. Fujita, T. Wang, S. Watanabe, and M. Omachi. "Toward Streaming ASR with Non-Autoregressive Insertion-Based Model". In: Aug. 2021, pp. 3740–3744. DOI: 10.21437/Interspeech.2021-1131. arXiv: 2107.09428 [eess.AS] (p. 34).

[28] P. Gage. "A new algorithm for data compression". In: *C Users J.* 12.2 (Feb. 1994), pp. 23–38. DOI: 10.5555/177910.177914 (p. 16).

[29] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. "Convolutional sequence to sequence learning". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. JMLR.org, 2017, pp. 1243–1252. DOI: 10.5555/3305381.3305510. arXiv: 1705.03122 (p. 15).

[30] M. Ghazvininejad, O. Levy, Y. Liu, and L. Zettlemoyer. "Mask-predict: Parallel decoding of conditional masked language models". In: *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*. 2019, pp. 6112–6121. DOI: 10.18653/v1/d19-1633. arXiv: 1904.09324 (p. 37).

[31] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. Online book (pp. 7, 8, 11).

[32] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks". In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, pp. 369–376. DOI: 10.1145/1143844.1143891. Online document (pp. 22, 36, 51).

[33] A. Gulati, J. Qin, C. C. Chiu, et al. "Conformer: Convolution-augmented transformer for speech recognition". In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. 2020, pp. 5036–5040. DOI: 10.21437/Interspeech.2020-3015. arXiv: 2005.08100 (pp. 29, 30).

[34] P. Haghani, A. Narayanan, M. Bacchiani, et al. "From Audio to Semantics: Approaches to End-to-End Spoken Language Understanding". In: *2018 IEEE Spoken Language Technology Workshop, SLT 2018 - Proceedings*. IEEE, 2018, pp. 720–726. DOI: 10.1109/SLT.2018.8639043. arXiv: 1809.09190 (pp. 29, 49).

[35] A. Hannun. "Sequence Modeling with CTC". In: *Distill* (2017). DOI: 10.23915/distill.00008. Blog post (pp. 22, 23).

[36] P. He, X. Liu, J. Gao, and W. Chen. "DeBERTa: Decoding-enhanced BERT With Disentangled Attention". In: *ICLR 2021 - 9th International Conference on Learning Representations* (2021). arXiv: 2006.03654 (pp. 16, 54, 86).

[37] Y. Higuchi, T. Ogawa, T. Kobayashi, and S. Watanabe. "BECTRA: Transducer-Based End-To-End ASR with Bert-Enhanced Encoder". In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2023, pp. 1–5. DOI: 10.1109/ICASSP49357.2023.10095186. IEEE: 10095186 (p. 34).

[38] Y. Higuchi, A. Rosenberg, Y. Wang, M. K. Baskar, and B. Ramabhadran. "Mask-Conformer: Augmenting Conformer with Mask-Predict Decoder". In: *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. 2023, pp. 1–8. DOI: 10.1109/ASRU57964.2023.10389803. IEEE: 10389803 (p. 34).

[39] Y. Higuchi, S. Watanabe, N. Chen, T. Ogawa, and T. Kobayashi. "Mask CTC: Non-autoregressive end-to-end ASR with CTC and mask predict". In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. 2020, pp. 3655–3659. DOI: 10.21437/Interspeech.2020-2404. arXiv: 2005.08700 (pp. 33–35, 37, 40).

[40] Y. Higuchi, B. Yan, S. Arora, et al. “BERT Meets CTC: New Formulation of End-to-End Speech Recognition with Pre-trained Masked Language Model”. In: *Findings of the Association for Computational Linguistics: EMNLP 2022*. Ed. by Y. Goldberg, Z. Kozareva, and Y. Zhang. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 5486–5503. DOI: 10.18653/v1/2022.findings-emnlp.402. arXiv: 2210.16663 (p. 34).

[41] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. “Improving neural networks by preventing co-adaptation of feature detectors”. In: *Computing Research Repository (CoRR)* cs.NE/1207.0580 (2012). arXiv: 1207.0580 (p. 9).

[42] W. N. Hsu, B. Bolte, Y. H. H. Tsai, et al. “HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units”. In: *IEEE/ACM Transactions on Audio Speech and Language Processing* 29 (2021), pp. 3451–3460. DOI: 10.1109/TASLP.2021.3122291. arXiv: 2106.07447 (pp. 2, 58).

[43] E. Hu, Y. Shen, P. Wallis, et al. “Lora: Low-Rank Adaptation of Large Language Models”. In: *ICLR 2022 - 10th International Conference on Learning Representations*. International Conference on Learning Representations, ICLR, June 2022. arXiv: 2106.09685 (p. 99).

[44] G. Jawahar, B. Sagot, and D. Seddah. “What Does BERT Learn about the Structure of Language?” In: *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference* (2019), pp. 3651–3657. DOI: 10.18653/V1/P19-1356. ACL: P19-1356 (pp. 7, 18).

[45] R. Józefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. “Exploring the Limits of Language Modeling”. In: *Computing Research Repository (CoRR)* cs.CL/1602.02410 (2016). arXiv: 1602.02410 (p. 17).

[46] D. Jurafsky and J. H. Martin. *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Upper Saddle River, N.J.: Pearson Prentice Hall, 2009. Online book (pp. 19, 20).

[47] M. Kim, G. Kim, S. W. Lee, and J. W. Ha. “ST-BERT: Cross-modal language model pre-training for end-to-end spoken language understanding”. In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. 2021, pp. 7478–7482. DOI: 10.1109/ICASSP39728.2021.9414558. arXiv: 2010.12283 (pp. 29, 30, 34, 40–42).

[48] D. P. Kingma and J. L. Ba. "Adam: A method for stochastic optimization". In: *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. 2015. arXiv: 1412.6980 (p. 40).

[49] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, et al. "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the National Academy of Sciences of the United States of America* 114.13 (2017), pp. 3521–3526. DOI: 10.1073/pnas.1611835114. arXiv: 1612.00796 (p. 11).

[50] T. Kudo and J. Richardson. "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Ed. by E. Blanco and W. Lu. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 66–71. DOI: 10.18653/v1/D18-2012. arXiv: 1808.06226 (p. 16).

[51] J. Lafferty, A. Mccallum, and F. Pereira. "Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data Abstract". In: *Proceedings of the Eighteenth International Conference on Machine Learning*. Vol. 2001. ICML June. Morgan Kaufmann Publishers Inc., 1999, pp. 282–289. ACM: 10.5555/645530.655813 (pp. 19, 60).

[52] C.-J. Lee, S.-K. Jung, K.-D. Kim, D.-H. Lee, and G. G.-B. Lee. "Recent Approaches to Dialog Management for Spoken Dialog Systems". In: *Journal of Computing Science and Engineering* 4.1 (2010), pp. 1–22. DOI: 10.5626/jcse.2010.4.1.001 (p. 30).

[53] M. Lewis, Y. Liu, N. Goyal, et al. "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension". In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics* (2020), pp. 7871–7880. DOI: 10.18653/v1/2020.acl-main.703. arXiv: 1910.13461 (p. 16).

[54] M. Li and R. S. Doddipatla. "Non-Autoregressive End-to-End Approaches for Joint Automatic Speech Recognition and Spoken Language Understanding". In: *2022 IEEE Spoken Language Technology Workshop (SLT)* (2023), pp. 390–397. arXiv: 2304.10869 (p. 34).

[55] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. "Focal Loss for Dense Object Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.2 (2020), pp. 318–327. DOI: 10.1109/TPAMI.2018.2858826. arXiv: 1708.02002 (p. 10).

[56] Y. Liu, M. Ott, N. Goyal, et al. “RoBERTa: A Robustly Optimized BERT Pretraining Approach”. In: *Computing Research Repository (CoRR)* cs.CL/1907.11692 (2019). arXiv: 1907.11692 (pp. 16, 18).

[57] I. Loshchilov and F. Hutter. “Decoupled weight decay regularization”. In: *7th International Conference on Learning Representations, ICLR 2019*. 2019. arXiv: 1711.05101 (p. 62).

[58] A. S. Luccioni, S. Viguier, and A.-L. Ligozat. “Estimating the Carbon Footprint of BLOOM, a 176B Parameter Language Model”. In: (2022). arXiv: 2211.02001 [cs.LG] (p. ii).

[59] L. Lugosch, M. Ravanelli, P. Ignoto, V. S. Tomar, and Y. Bengio. “Speech model pre-training for end-to-end spoken language understanding”. In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. 2019, pp. 814–818. DOI: 10.21437/Interspeech.2019-2396. arXiv: 1904.03670 (pp. 27, 29, 31, 40–42, 60, 62, 63, 65).

[60] L. van der Maaten and G. Hinton. “Visualizing Data using t-SNE”. In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605 (p. 45).

[61] E. Maekawa. *Annotation Guidelines for Named Entities*. 2018. Online document (p. 75).

[62] S. Mdhaffar, J. Duret, T. Parcollet, and Y. Estève. “End-to-end model for named entity recognition from speech without paired training data”. In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. 2022, pp. 4068–4072. DOI: 10.21437/Interspeech.2022-10231. arXiv: 2204.00803 (p. 29).

[63] Q. Meeus, M.-F. Moens, and H. Van Hamme. “Multitask Learning for Low Resource Spoken Language Understanding”. In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. 2022, pp. 4073–4077. DOI: 10.21437/Interspeech.2022-11401. arXiv: 2211.13703 (pp. 49, 63, 119).

[64] Q. Meeus, M.-F. Moens, and H. Van hamme. “Bidirectional Representations for Low-Resource Spoken Language Understanding”. In: *Applied Sciences* 13.20 (2023). DOI: 10.3390/app132011291. arXiv: 2211.14320 (pp. 18, 24, 28, 33, 119).

[65] Q. Meeus, M.-F. Moens, and H. Van hamme. “MSNER: A Multilingual Speech Dataset for Named Entity Recognition”. In: *LREC-COLING ISA-20 Workshop*. 2024. arXiv: 2405.11519 (pp. 18, 19, 67, 68, 84–86, 120).

[66] Q. Meeus, M.-F. Moens, and H. Van hamme. “Multilingual Spoken Named Entity Recognition with Transformers (under review)”. In: *Natural Language Processing* (2024) (pp. 25, 28, 81, 120).

[67] Q. Meeus, M.-F. Moens, and H. Van hamme. “Whisper-SLU: Extending a Pretrained Speech-to-Text Transformer for Low Resource Spoken Language Understanding”. In: *2023 IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2023*. 2023, pp. 189–194. DOI: 10.1109/ASRU57964.2023.10389786. IEEE: 10389786 (pp. 57, 76, 120).

[68] A. Mohamed, D. Okhonko, and L. Zettlemoyer. “Transformers with convolutional context for ASR”. In: *Computing Research Repository (CoRR)* cs.CL/1904.11660 (2019). arXiv: 1904.11660 (pp. 24, 25, 36, 50).

[69] N. Oostdijk. *Het Corpus Gesproken Nederlands*. Ed. by Taalunie. Version 2.0.3. 2014. HDL: 10032/tm-a2-k6 (pp. 21, 30, 52).

[70] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. “Librispeech: An ASR corpus based on public domain audio books”. In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. 2015, pp. 5206–5210. DOI: 10.1109/ICASSP.2015.7178964. IEEE: 7178964 (pp. 21, 30, 52).

[71] A. Paszke, S. Gross, F. Massa, et al. “PyTorch: An imperative style, high-performance deep learning library”. In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019. DOI: 10.5555/3454287.3455008. arXiv: 1912.01703 (p. 62).

[72] Y. Peng, S. Arora, Y. Higuchi, et al. “A Study on the Integration of Pre-Trained SSL, ASR, LM and SLU Models for Spoken Language Understanding”. In: *2022 IEEE Spoken Language Technology Workshop, SLT 2022 - Proceedings*. IEEE, 2023, pp. 406–413. DOI: 10.1109/SLT54892.2023.10022399. arXiv: 2211.05869 (pp. 29, 30, 63, 64).

[73] M. E. Peters, M. Neumann, M. Iyyer, et al. “Deep contextualized word representations”. In: *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*. Vol. 1. 2018, pp. 2227–2237. DOI: 10.18653/v1/n18-1202. arXiv: 1802.05365 (p. 37).

[74] O. Press and L. Wolf. “Using the output embedding to improve language models”. In: *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference*. Vol. 2. 2017, pp. 157–163. DOI: 10.18653/v1/e17-2025. arXiv: 1608.05859 (p. 25).

[75] A. Radford, J. W. Kim, T. Xu, et al. “Robust speech recognition via large-scale weak supervision”. In: *Proceedings of the 40th International Conference on Machine Learning*. ICML’23. JMLR.org, 2023. arXiv: 2212.04356 (pp. 24–26, 70, 76).

[76] L. A. Ramshaw and M. P. Marcus. “Text Chunking Using Transformation-Based Learning”. In: *Third Workshop on Very Large Corpora*. 1999, pp. 157–176. DOI: 10.1007/978-94-017-2390-9_10. arXiv: cmp-lg/9505040 (p. 19).

[77] M. Rao, A. Raju, P. Dheram, B. Bui, and A. Rastrow. “Speech to semantics: Improve ASR and NLU jointly via all-neural interfaces”. In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. Ed. by H. Meng, B. Xu, and T. F. Zheng. ISCA, 2020, pp. 876–880. DOI: 10.21437/Interspeech.2020-2976. arXiv: 2008.06173 (p. 49).

[78] M. Ravanelli, T. Parcollet, P. Plantinga, et al. *SpeechBrain: A General-Purpose Speech Toolkit*. 2021. arXiv: 2106.04624 (p. 21).

[79] V. Renkens, S. Janssens, B. Ons, J. F. Gemmeke, and H. Van Hamme. “Acquisition of ordinal words using weakly supervised NMF”. In: *2014 IEEE Workshop on Spoken Language Technology, SLT 2014 - Proceedings*. 2014, pp. 30–35. DOI: 10.1109/SLT.2014.7078545. IEEE: 7078545 (p. 52).

[80] V. Renkens and H. Van Hamme. “Capsule networks for low resource spoken language understanding”. In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. International Speech Communication Association, 2018, pp. 601–605. DOI: 10.21437/Interspeech.2018-1013. arXiv: 1805.02922 (p. 31).

[81] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne. “Experience replay for continual learning”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, et al. Vol. 32. Curran Associates, Inc., 2019. arXiv: 1811.11682 (p. 11).

[82] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, et al. “Progressive Neural Networks”. In: *Computing Research Repository (CoRR)* cs.LG/1606.04671 (2016). arXiv: 1606.04671 (p. 11).

[83] A. Saade, J. Dureau, D. Leroy, et al. “Spoken Language Understanding on the Edge”. In: *Proceedings - 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing, EMC2-NIPS 2019* (2019), pp. 57–61. DOI: 10.1109/EMC2-NIPS53020.2019.00021. arXiv: 1810.12735 (pp. 31, 61).

[84] S. Schneider, A. Baevski, R. Collobert, and M. Auli. "wav2vec: Unsupervised Pre-Training for Speech Recognition". In: *Proc. Interspeech 2019*. 2019, pp. 3465–3469. DOI: 10.21437/Interspeech.2019-1873. arXiv: 1904-05862 (p. 2).

[85] D. Serdyuk, Y. Wang, C. Fuegen, et al. "Towards End-to-end Spoken Language Understanding". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, pp. 5754–5758. DOI: 10.1109/ICASSP.2018.8461785. arXiv: 1802.08395 (p. 27).

[86] S. Shon, A. Pasad, F. Wu, et al. "Slue: New Benchmark Tasks for Spoken Language Understanding Evaluation on Natural Speech". In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* (2022), pp. 7927–7931. DOI: 10.1109/ICASSP43922.2022.9746137. arXiv: 2111.10367 (pp. 30, 31, 54, 56, 61, 63, 64, 68, 69, 71, 75, 76, 78, 84, 86).

[87] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2015. arXiv: 1409.1556 (p. 7).

[88] M. Someki, N. Eng, Y. Higuchi, and S. Watanabe. "Segment-Level Vectorized Beam Search Based on Partially Autoregressive Inference". In: *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. 2023, pp. 1–8. DOI: 10.1109/ASRU57964.2023.10389796. IEEE: 10389796 (p. 34).

[89] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. "Rethinking the Inception Architecture for Computer Vision". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2016, pp. 2818–2826. DOI: 10.1109/CVPR.2016.308. arXiv: 1512.00567 (pp. 10, 24).

[90] S. Tedeschi, S. Conia, F. Cecconi, and R. Navigli. "Named Entity Recognition for Entity Linking: What Works and What's Next". In: *Findings of the Association for Computational Linguistics, Findings of ACL: EMNLP 2021*. 2021, pp. 2584–2596. DOI: 10.18653/v1/2021.findings-emnlp.220. ACL: 2021.findings-emnlp.220 (p. 19).

[91] S. Tedeschi and R. Navigli. "MultiNERD: A Multilingual, Multi-Genre and Fine-Grained Dataset for Named Entity Recognition (and Disambiguation)". In: *Findings of the Association for Computational Linguistics: NAACL 2022 - Findings*. Seattle, United States: Association for Computational Linguistics, 2022, pp. 801–812. DOI: 10.18653/v1/2022.findings-naacl.60. ACL: 2022.findings-naacl.60 (p. 19).

[92] N. Tessema, B. Ons, J. van de Loo, et al. “Metadata for corpora patcor and domotica-2”. In: *ESAT, KU Leuven, Belgium* (2013). Lirias: 236565 (p. 31).

[93] E. F. Tjong and K. Sang. “Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition”. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. ACL, 2002, pp. 142–147. ACL: W02-2024 (p. 19).

[94] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou. “Going deeper with Image Transformers”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2021, pp. 32–42. DOI: 10.1109/ICCV48922.2021.00010. arXiv: 2103.17239 (pp. 37, 38, 51).

[95] H. Touvron, T. Lavril, G. Izacard, et al. “LLaMA: Open and Efficient Foundation Language Models”. In: (Feb. 2023). arXiv: 2302.13971 (pp. 16, 17).

[96] A. Ushio and J. Camacho-Collados. “T-NER: An all-round python library for transformer-based named entity recognition”. In: *EACL 2021 - 16th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the System Demonstrations*. Association for Computational Linguistics, 2021, pp. 53–62. DOI: 10.18653/v1/2021.eacl-demos.7. arXiv: 2209.12616 (p. 19).

[97] Ö. Uzuner, I. Solti, F. Xia, and E. Cadag. “Community annotation experiment for ground truth generation for the i2b2 medication challenge”. In: *Journal of the American Medical Informatics Association* 17.5 (2010), pp. 519–523. DOI: 10.1136/jamia.2010.004200. PMID: 20819855 (p. 18).

[98] S. Vander Eeckt and H. Van hamme. “Weight Averaging: A Simple Yet Effective Method to Overcome Catastrophic Forgetting in Automatic Speech Recognition”. In: *ICASSP*. IEEE, June 2023. DOI: 10.1109/icassp49357.2023.10095147. arXiv: 2210.15282 (p. 11).

[99] A. Vaswani, N. Shazeer, N. Parmar, et al. “Attention is all you need”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 5999–6009. arXiv: 1706.03762 (pp. 12–17, 29, 40, 51).

[100] A. J. Viterbi. “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm”. In: *IEEE Transactions on Information Theory* 13.2 (1967), pp. 260–269. DOI: 10.1109/TIT.1967.1054010. IEEE: 1054010 (p. 60).

[101] W. de Vries, A. van Cranenburgh, A. Bisazza, et al. “BERTje: A Dutch BERT Model”. In: *Computing Research Repository (CoRR)* cs.CL/1912.09582 (2019). arXiv: 1912.09582 (pp. 42, 54).

[102] C. Wang, M. Rivière, A. Lee, et al. "VoxPopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation". In: *ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference*. ACL, 2021, pp. 993–1003. DOI: 10.18653/v1/2021.acl-long.80. arXiv: 2101.00390 (pp. 21, 31, 68, 84, 100).

[103] P. Wang and H. Van hamme. "Benefits of pre-trained mono-and cross-lingual speech representations for spoken language understanding of Dutch dysarthric speech". In: *EURASIP Journal on Audio, Speech, and Music Processing* 2023 (2023), p. 15. DOI: 10.1186/s13636-023-00280-z. ACM: 10.1186/s13636-023-00280-z (p. 44).

[104] P. Wang and H. Van hamme. "Pre-training for low resource speech-to-intent applications". In: *Computing Research Repository (CoRR)* eess.AS/2103.16674 (2020). arXiv: 2103.16674 (pp. 43, 44).

[105] S. Watanabe, T. Hori, S. Karita, et al. "ESPNet: End-to-end speech processing toolkit". In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. 2018, pp. 2207–2211. DOI: 10.21437/Interspeech.2018-1456. arXiv: 1804.00015 (pp. 16, 21, 24, 50, 52, 62).

[106] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi. "Hybrid CTC/Attention Architecture for End-to-End Speech Recognition". In: *IEEE Journal of Selected Topics in Signal Processing* 11.8 (2017), pp. 1240–1253. IEEE: 8068205 (pp. 24, 25, 36, 42).

[107] R. Weischedel, M. Palmer, M. Marcus, et al. *OntoNotes Release 5.0*. Version 5.0. Linguistic Data Consortium, 2013. DOI: 10.35111/xmhb-2b84. LDC: LDC2013T19 (pp. 69, 70, 76, 84, 100).

[108] T. Wolf, L. Debut, V. Sanh, et al. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Ed. by Q. Liu and D. Schlangen. Association for Computational Linguistics, Oct. 2020, pp. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6. arXiv: 1910.03771 (pp. 57, 62, 75).

[109] R. Xiong, Y. Yang, D. He, et al. "On layer normalization in the transformer architecture". In: *37th International Conference on Machine Learning, ICML 2020*. Vol. PartF16814. ICML'20. JMLR.org, 2020, pp. 10455–10464. arXiv: 2002.04745 (pp. 15, 37).

[110] W. Ye, Y. Ma, X. Cao, and K. Tang. "Mitigating Transformer Overconfidence via Lipschitz Regularization". In: *Proceedings of Machine Learning Research*. Ed. by R. J. Evans and I. Shpitser. Vol. 216.

Proceedings of Machine Learning Research. PMLR, 2023, pp. 2422–2432. arXiv: 2306.06849 (p. 76).

[111] T. Zenkel, R. Sanabria, F. Metze, et al. "Comparison of decoding strategies for CTC acoustic models". In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. 2017, pp. 513–517. DOI: 10.21437/Interspeech.2017-1683. arXiv: 1708.04469 (p. 36).

[112] Q. Zhang, M. Chen, A. Bukharin, et al. "AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning". In: (2023). arXiv: 2303.10512 (p. 99).

[113] H. Zhao, Y. Higuchi, Y. Kida, T. Ogawa, and T. Kobayashi. "Mask-CTC-Based Encoder Pre-Training for Streaming End-to-End Speech Recognition". In: *2023 31st European Signal Processing Conference (EUSIPCO)*. 2023, pp. 56–60. DOI: 10.23919/EUSIPCO58844.2023.10290112. IEEE: 10290112 (p. 34).

Short Biography

Hugo Van hamme, Joris Pelemans and Quentin Meeus – University Trail 2023

Quentin Meeus was born on March 19th, 1992 in Brussels, Belgium. In 2015, he receives a Master degree in Business Engineering (op. Advanced Finance) from Université Libre de Bruxelles. His master thesis, "*Topmob: A medical transport platform for a greater mobility*", introduces an intelligent booking tool for medical transport. After working for a while as a business developer, and then as a consultant, he goes back to school to specialize in machine learning and completes a Master degree in Artificial Intelligence (op. Engineering and Computer Science) at KU Leuven in 2019. He wrote his master thesis, titled "*Semi-supervised autoencoders for long text classification*", under the supervision of his future PhD co-supervisor Sien Moens. The same year, he started a joint PhD at KU Leuven under the supervision of Sien Moens (CS-LIIR) and Hugo Van hamme (ESAT-PSI). His research interests lie at the intersection of speech processing and language modeling, with a focus on low resource spoken language understanding, named entity recognition, and multilingual systems.

Publications

- Q. Meeus, M.-F. Moens, and H. Van hamme. "Bidirectional Representations for Low-Resource Spoken Language Understanding". In: *Applied Sciences* 13.20 (2023). DOI: 10.3390/app132011291. arXiv: 2211.14320
  Abstract: Speech representation models lack the ability to efficiently store semantic information and require fine tuning to deliver decent performance. In this research, we introduce a transformer encoder–decoder framework with a multiobjective training strategy, incorporating connectionist temporal classification (CTC) and masked language modeling (MLM) objectives. This approach enables the model to learn contextual bidirectional representations. We evaluate the representations in a challenging low-resource scenario, where training data is limited, necessitating expressive speech embeddings to compensate for the scarcity of examples. Notably, we demonstrate that our model's initial embeddings outperform comparable models on multiple datasets before fine tuning. Fine tuning the top layers of the representation model further enhances performance, particularly on the Fluent Speech Command dataset, even under low-resource conditions. Additionally, we introduce the concept of class attention as an efficient module for spoken language understanding, characterized by its speed and minimal parameter requirements. Class attention not only aids in explaining model predictions but also enhances our understanding of the underlying decision-making processes. Our experiments cover both English and Dutch languages, offering a comprehensive evaluation of our proposed approach.

- Q. Meeus, M.-F. Moens, and H. Van Hamme. "Multitask Learning for Low Resource Spoken Language Understanding". In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. 2022, pp. 4073–4077. DOI: 10.21437/Interspeech.2022-11401. arXiv: 2211.13703
  Abstract: We explore the benefits that multitask learning offer to speech processing as we train models on dual objectives with automatic speech recognition and intent classification or sentiment classification. Our models, although being of modest size, show improvements over models trained end-to-end on intent classification. We compare different settings to find the optimal disposition of each task module compared to one another. Finally, we study the performance of the models in low-resource scenario by training the models with as few as one example per class. We show that multitask learning in these scenarios compete with a baseline model trained on text features and performs considerably better than a pipeline model. On sentiment classification, we match the performance of an end-to-end model with ten times as many parameters. We consider 4 tasks and 4 datasets in Dutch and English.

- Q. Meeus, M.-F. Moens, and H. Van hamme. "Whisper-SLU: Extending a Pretrained Speech-to-Text Transformer for Low Resource Spoken Language Understanding". In: *2023 IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2023*. 2023, pp. 189–194. DOI: 10.1109/ASRU57964.2023.10389786. IEEE: 10389786
  Abstract: Human-computer interactions require systems that work out of the box without requiring lots of data to adapt to a new task or user. In this research, we address low resource spoken language understanding tasks such as named entity recognition (NER), intent recognition (IR), and slot filling (SF) to research how a pretrained model can be modified for a new task, then finetuned with few labelled data. We propose extending the Whisper model with task-specific modules for NER, SF, and IR, leveraging a Markov network as output structure. We develop a novel approach to finetuning by removing irrelevant weights and reorganizing the embeddings to drastically improve the performance in a low-resource setting. Our approach outperforms previous models without external language models and demonstrates effective transfer learning, even with very limited training data. The models exhibit a small footprint, making them suitable for applications requiring robustness, few-shot learning, and efficiency.

- Q. Meeus, M.-F. Moens, and H. Van hamme. "MSNER: A Multilingual Speech Dataset for Named Entity Recognition". In: *LREC-COLING ISA-20 Workshop*. 2024. arXiv: 2405.11519
  Abstract: While extensively explored in text-based tasks, Named Entity Recognition (NER) remains largely neglected in spoken language understanding. Existing resources are limited to a single, English-only dataset. This paper addresses this gap by introducing MSNER, a freely available, multilingual speech corpus annotated with named entities. It provides annotations to the VoxPopuli dataset in four languages (Dutch, French, German, and Spanish). We have also releasing an efficient annotation tool that leverages automatic pre-annotations for faster manual refinement. This results in 590 and 15 hours of silver-annotated speech for training and validation, alongside a 17-hour, manually-annotated evaluation set. We further provide an analysis comparing silver and gold annotations. Finally, we present baseline NER models to stimulate further research on this newly available dataset.

- Q. Meeus, M.-F. Moens, and H. Van hamme. "Multilingual Spoken Named Entity Recognition with Transformers (under review)". In: *Natural Language Processing* (2024)
  Abstract: Multilingual Spoken Named Entity Recognition (Spoken NER) is a hardly explored field of Spoken Language Understanding (SLU) which aims to extract named entities directly from speech. We propose Whisper-SLU and Whisper-SLU end-to-end, two transformer-based speech models for Spoken NER. Both approaches offer advantages compared to more traditional pipeline models: they avoid error cascading and benefit from both acoustic and linguistic information. In a second step, we investigate catastrophic forgetting, where models forget past languages and tasks during fine-tuning. Three mitigation methods are considered: resetting embeddings, disassembling embeddings, and experience replay. Resetting embeddings to their pretrained value after fine-tuning proved to be simple but effective. Disassembling embeddings replicates this during training by separating pretrained and new embeddings. Both approaches specifically target language forgetting. Experience replay tackles task forgetting by mixing training examples with pseudo-data generated by the pretrained model. Our findings offer insights for developing robust multilingual Spoken NER models, emphasizing the importance of understanding the effects of fine-tuning while preserving pretrained model capabilities. Future work will address limitations and extend Whisper-SLU to other SLU tasks and languages.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
ESAT-PSI/CW-LIIR
Kasteelpark Arenberg 10
B-3001 Leuven