End-to-end named entity recognition for spoken Finnish

Dejan Porjazovski

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.
Espoo 28.9.2020

Supervisor

Prof. Mikko Kurimo

Advisor

MSc Juho Leinonen

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Dejan Porjazovski

Title End-to-end named entity recognition for spoken Finnish

Degree programme Computer, Communication and Information Sciences

Major Machine Learning, Data Science and Artificial Intelligence (Macadamia) **Code of major** SCI3044

Supervisor Prof. Mikko Kurimo

Advisor MSc Juho Leinonen

Date 28.9.2020 **Number of pages** 57 **Language** English

Abstract

Named entity recognition is a natural language processing task in which the system tries to find named entities and classify them in predefined categories. The categories can vary, depending on the domain in which they are going to be used but some of the most common include: person, location, organization, date and product. Named entity recognition is an integral part of other large natural language processing tasks, such as information retrieval, text summarization, machine translation, and question answering.

Doing named entity recognition is a difficult task due to the lack of annotated data for certain languages or domains. Named entity ambiguity is another challenging aspect that arises when doing named entity recognition. Often times, a word can represent a person, organization, product, or any other category, depending on the context it appears in.

Spoken data, which can be the output of a speech recognition system, imposes additional challenges to the named entity recognition system. Named entities are often capitalized and the system learns to rely on capitalization in order to detect the entities, which is neglected in the speech recognition output.

The standard way of doing named entity recognition from speech involves a pipeline approach of two systems. First, a speech recognition system transcribes the speech and generates the transcripts, after which a named entity recognition system annotates the transcripts with the named entities. Since the speech recognition system is not perfect and makes errors, those errors are propagated to the named entity recognition system, which is hard to recover from.

In this thesis, we present two approaches of doing named entity recognition from Finnish speech in an end-to-and manner, where one system generates the transcripts and the annotations. We will explore the strengths and weaknesses of both approaches and see how they compare to the standard pipeline approach.

Keywords named entity recognition, speech recognition, end-to-end, low-resource

Preface

This thesis summarizes the most important work that I have done during my Master's studies at Aalto University. The process of doing this thesis taught me a lot about speech recognition, named entity recognition and machine learning in general. Moreover, it taught me how to conduct meaningful research, develop various hypotheses and test them.

I would like to thank my supervisor, Prof. Mikko Kurimo, for allowing me the opportunity to be part of his team and allowing me to work on this exciting topic. I would also like to thank my advisor, Juho Leinonen, for guiding me through the process and giving me important technical and writing feedback, whenever needed. I also thank the whole research group for giving me valuable feedback during the various stages of the thesis. The computational resources were provided by Aalto Science-IT, without which, doing the experiments would have been impossible. Finally, I would like to thank every one of you that took the time to read this thesis. I hope that it was as entertaining to you as it was to me.

Espoo, 7.9.2020

Dejan Porjazovski

Contents

List of Figures

List of Tables

Symbols and abbreviations

Symbols

e	Euler's number, a mathematical constant
tanh	Hyperbolic tangent function
argmax	Argument with the highest probability
cos	Cosine function
π	Pi, a mathematical constant
$\sum$	Summation
$\prod$	Product
$P(y\|x)$	Probability of y given x
P_{LM}	Language model probability
$\langle\rangle$	Dot product
exp	Exponential function
L	Multi-task loss
L_{asr}	Automatic speech recognition loss
L_{ner}	Named entity recognition loss
λ	Weighting factor for multi-task loss
β	Weighting factor during shallow fusion
θ	Model parameters

Abbreviations

O	"Other" entity tag
PER	"Person" entity tag
LOC	"Location" entity tag
ORG	"Organization" entity tag
PRO	"Product" entity tag
EVENT	"Event" entity tag
DATE	"Date" entity tag
<UNK>	Unknown token
<sos>	Start of string
<eos>	End of string
ML	Machine learning
NN	Neural network
NER	Named entity recognition
NLP	Natural language processing
NLU	Natural language understanding
SLU	Spoken language understanding
E2E	End-to-end
ASR	Automatic speech recognition
SD	Speaker dependent
SI	Speaker independent
CD	Context dependent
MFCC	Mel-frequency cepstral coefficient
DFT	Discrete Fourier transform
DCT	Discrete cosine transform
OOV	Out of vocabulary
HMM	Hidden Markov model
GMM	Gaussian mixture model
DNN	Deep neural network
CTC	Connectionist temporal classification
RNN	Recurrent neural network
ME	Maximum entropy
MENE	Maximum entropy named entity
CRF	Conditional random field
GRU	Gated recurrent unit
LSTM	Long short-term memory
BLSTM	Bidirectional long short-term memory
WER	Word error rate
AED	Attention-based encoder-decoder
LM	Language model

1 Introduction

Named entity recognition (NER) is one of the main natural language processing (NLP) tasks. The goal of this task is to find entities and classify them in predefined categories. These categories can vary depending on the area they are used in but the most common ones include person, location, date, and organization.

Named entity recognition systems have a wide variety of applications. They are integral part of other larger areas such as: text summarization [1], machine translation [2] and information extraction [3, 4]. Text summarization usually contains people, dates, and locations, which can be detected using a named entity recognition system. When doing machine translation, it is important to detect the named entities in the source language so that they won't be excluded when doing the translation to the target language. The search queries usually contain named entities, which are important to detect in order to retrieve the relevant documents. Named entities are also heavily used in natural language understanding (NLU) and spoken language understanding (SLU) areas, which are essential for personal assistants in home automation and smartphone devices. These personal assistants usually use speech as input, in which case the named entities need to be recognized from spoken data.

Doing named entity recognition from speech imposes several challenges for the system. There is far less annotated data for spoken language than for textual data. The speech can be informal, not following the conventional syntax of the language, which can cause difficulties in detecting the entities. The generated transcripts from a speech recognition system usually don't contain capitalization and punctuation, which can cause the system to miss the entities.

This thesis will focus on end-to-end (E2E) named entity recognition for spoken Finnish. End-to-end in this sense means that the named entities will be learned directly from speech, without relying on an external speech recognition system to generate the transcripts. The conventional methods for extracting named entities from speech are done in a pipeline approach where first a speech recognizer generates transcripts and then named entity recognition is done on those transcripts. Even though this approach can work well, the speech recognition systems are not perfect and they make errors. The generated transcripts can have misspelled or missing words, that can have an impact on the performance of the named entity recognition system. To mitigate this, in the thesis we will explore ways of directly extracting named entities from speech and see how well they perform compared to the standard pipeline approach.

1.1 Scope of the thesis

The scope of this thesis is to explore ways of extracting named entities from spoken Finnish in an end-to-end manner. We will explore two methods of doing that: augment the labels with named entity tags, meaning that we will add the named entity tags to the original transcripts and a multi-task approach by learning both to transcribe speech and recognize named entities.

1.2 Research questions

The main research questions that this thesis is going to answer are:

1. Can the named entity tags be learned directly from acoustic features?
2. Does doing automatic speech recognition help in learning the named entities?
3. How well these approaches perform, compared to the named entity annotations produced by the pipeline approach?

1.3 Outline of the thesis

In **Chapter 2** we will talk about what machine learning is and the data needed for training a machine learning model. Furthermore, we will talk about the different types of machine learning algorithms and focus on the ones that are most relevant to our problem, the neural networks, and conditional random fields.

In **Chapter 3** we will familiarize ourselves with what automatic speech recognition is, why is it important, as well as different types of speech recognition systems. Later, we will go into the building blocks of speech recognition systems and explore ways of assessing their performance. Moreover, we will explore various research that was done in the automatic speech recognition field and see their strengths and weaknesses.

Chapter 4 explains the basic principles of named entity recognition. We will explore why named entity recognition is one of the main NLP tasks and see different approaches of constructing a named entity recognition system. Further, we will explore the challenges that arise when doing NER, as well as different ways of measuring the performance of the system. We will also familiarize ourselves with previous research done in this area and see various applications that benefit from NER system.

Chapter 5 explains what a language model is. Furthermore, it covers the most widely used language model types and how they are used in the end-to-end speech recognition systems.

Chapter 6 explains the two approaches of doing an end-to-end named entity recognition. It covers the augmented labels approach, as well as the multi-task learning approach and their mathematical foundations. Furthermore, it covers the baseline named entity recognition system that was used for annotating the training data.

Chapter 7 starts with explanation of the data that was used in the experiments. Further, it shows the experimental setup that we used, as well as the results obtained for the various experiments. At the end, we analyze the results and give an interpretation of them.

Chapter 8 talks about the conclusion of the thesis and the lessons leaned throughout the experiments. Furthermore, it explores ways of improving the existing approaches and building upon this work.

2 Machine learning

The standard paradigm of programming is to make a set of instructions that the computer will follow to achieve a specific goal. In contrast to that, machine learning (ML) algorithms provide an ability for the system to learn and improve without being explicitly programmed to do so. These algorithms similarly perform the tasks as humans. When we hear someone talking, we immediately understand what has been said, even though we often don't know how that is done.

2.1 Data in machine learning

The machine learning methods typically rely on large amounts of data, so that they can learn the patterns that emerge from it. The dataset that is used to train the machine learning model is called a training set. Depending on the content of the training set, the machine learning algorithm will be biased towards it. For example, if we are training a speech recognition system using a training set consisting of medical talks, the model will be good at recognizing that data but bad at recognizing other speech that is not part of the training set. In other words, it will not be able to generalize. To overcome that, it is a good practice to have a separate development (also known as evaluation) set that will be used to assess the model's performance during training. Since, the development set can also impose some bias to the model, if we want to assess the performance of the trained model, we need to use a separate test set. The test set is typically used only for measuring the performance of the trained model.

2.2 Types of machine learning methods

There are many different types of machine learning algorithms and choosing which one to use is not always trivial. Depending on the way the ML algorithms are trained, they are divided into several categories: supervised, unsupervised, semi-supervised, and reinforcement learning.

Supervised machine learning algorithms require labeled data (usually by humans) to learn from so that they can apply that knowledge to unseen data and predict the correct labels. The training data for these algorithms usually consists of input features and corresponding labels. The supervised machine learning methods are divided into classification and regression algorithms. The classification algorithms deal with problems where the output value is a category, whereas the regression algorithms deal with outputs containing continuous values.

Unsupervised machine learning algorithms, in contrast to the supervised ones, do not have corresponding labels and only rely on the input features. These algorithms try to model the distribution of the data to learn more about it and be able to predict new examples. The clustering algorithms, which try to divide the data into subgroups based on similarities, fall into this category.

Semi-supervised machine learning algorithms fall between the supervised and the unsupervised ones. They are usually applied in applications where there is both

labeled and unlabeled data.

Reinforcement learning algorithms typically involve an agent that interacts with the environment and tries to learn from it. The agent learns through trial and error by getting a positive reward for good actions and negative reward for undesired actions.

Many types of machine learning algorithms fall in one of the categories mentioned above and each one of them has its uses-cases. In this thesis, we will mostly focus on conditional random fields (CRFs) and neural networks, since they are most relevant to our problem that we are trying to solve.

2.3 Conditional random fields

Conditional random fields are a discriminative model, which means that they try to learn a decision boundary between the classes. This model is well suited for tasks where there is a dependency between the current and the previous predictions. This makes them a good candidate for sequence tagging tasks, like named entity recognition, part of speech tagging, and gene prediction.

CRFs work in a way that they try to model the conditional distribution as:

$$\widetilde{y} = argmax_y P(y|x) \tag{2.1}$$

The model uses feature functions to model the dependencies between the history and the current prediction. The feature function is defined as:

$$f(X, i, l_{i-1}, l_i) \tag{2.2}$$

where, X is the input, i is the index of the data point that we are predicting, l_{i-1} is the label of the data point $i-1$ and l_i is the label of the data point i.

Each feature function returns either 0 or 1, depending on the current label and the label of the previous data point. They do the decision based on a set of weights that are learned during training, using maximum likelihood estimation. An example of a CRF is shown in figure 1.

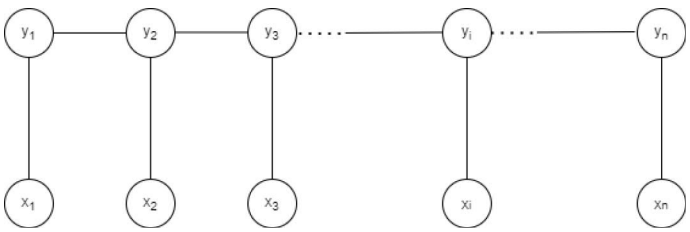

Figure 1: Conditional random field.

2.4 Neural networks

Neural networks (NNs) are a family of machine learning algorithms that are vaguely inspired by the neural network in our brain. The basic idea behind neural networks is shown in figure 2.

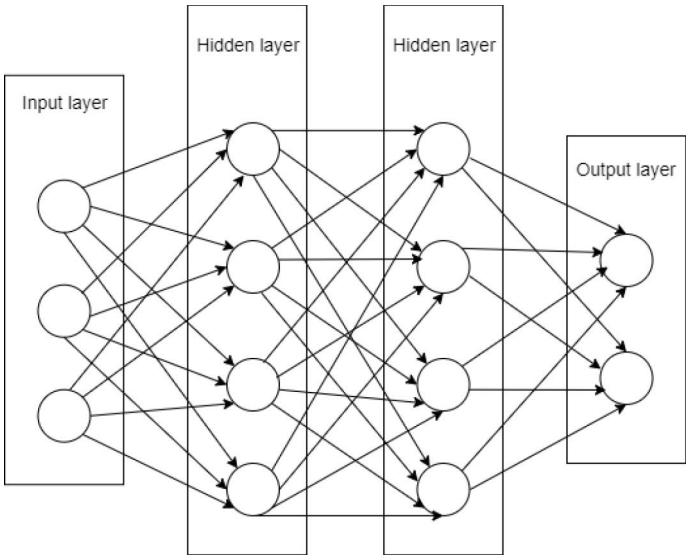

Figure 2: Neural network where circles represent the neurons and the rectangles represent the layers.

The network consists of layers and neurons. The input layers consist of the input features, which get passed to a set of hidden layers. The architecture works in a way that each neuron passes its output to the next neuron. A single layer is also called a perceptron and the way neurons work in a perceptron is shown in figure 3. Each neuron contains weight, which is a learnable parameter. The inputs are weighted and summed together with a bias term, which is also a learnable parameter. In the end, the weighted sum gets passed to a nonlinearity function. The nonlinearity function is important because that way the network can learn nonlinear representations. The number of layers and neurons is a design choice that has to be taken carefully into consideration.

The networks are usually trained with the backpropagation algorithm by propagating the errors from the end to the beginning of the network. That way the neural network can minimize an objective function and adjust the parameters in a way that yields optimal results. The network training and the backpropagation algorithm are beyond the scope of this thesis, so they won't be covered.

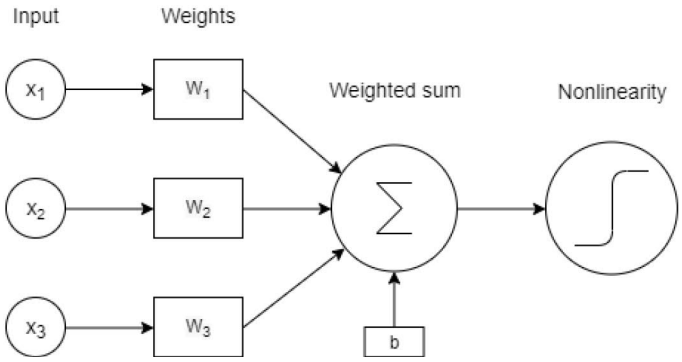

Figure 3: Single layer perceptron.

The neural networks play a vital role in many applications such as image recognition, fraud detection, speech recognition, and natural language processing tasks. They are a family of algorithms that differ between themselves in the way they do their computations. The most common architecture types include: feed-forward neural networks, convolutional neural networks, and recurrent neural networks. Some neural networks are better suited for processing images, whereas some work better for processing sequential data. Our models rely heavily on recurrent neural networks, so we will explore them in more detail in the following part of the thesis.

2.4.1 Recurrent neural networks

The standard neural networks, like the feed-forward one, assume that all the inputs and outputs are independent of each other. For some types of data, the independence assumption is correct but for others, such as time-series and signal data, this can be misleading. If we want to predict the next word in a sentence, it is important to keep track of the preceding words, since they may contain information relevant to the current prediction.

Recurrent neural networks (RNNs) are specialized at dealing with sequential data, where there is a dependence between the current and the previous predictions. This means that the previous predictions can impact future ones. In comparison to the standard feed-forward networks, where each input has its own weight, the recurrent neural networks do a parameter sharing, by sharing the weights across the inputs.

Often times, the future elements can also have an impact on the current predictions. In the sequence tagging tasks, like named entity recognition, the future words can have as much impact on the current prediction, as the past ones.

To capture the past and future information, bidirectional recurrent neural networks were developed. They work similarly as standard RNNs but they simultaneously process the input from the beginning and the end. An example of bidirectional RNN is shown in figure 4.

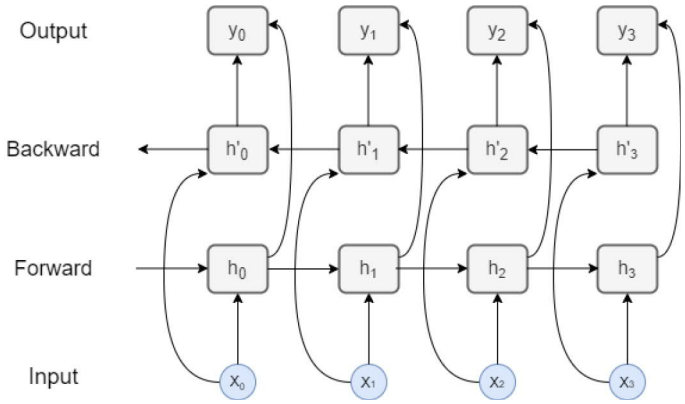

Figure 4: Bidirectional recurrent neural network.

In a standard RNN, the hidden state h, for each timestep t, is computed as:

$$h[t] = g(W_{hh}h_{t-1} + W_{xh}x_t) \tag{2.3}$$

where, W_{hh} and W_{xh} are learnable weights, h_{t-1} is the hidden state of the previous timestep, x_t is the input at the current timestep and g is a non-linear activation function.

The output y at timestep t is then computed as:

$$y[t] = W_{yh}h_t \tag{2.4}$$

where, W_{yh} is a learnable parameter and h_t is the hidden state of the current timestep.

Training the standard RNNs has two big issues, the vanishing, and exploding gradient problems, explored in [5, 6]. As a solution to these problems, two RNN variants became most popular: gated recurrent unit (GRU) [7] and long short-term memory (LSTM) [8]. In our experiments, we will use LSTM, so we will focus on that variant.

The LSTM neural network introduces three gates that determine the amount of information that passes through. The three gates are: forget gate, input gate, and output gate. The gates allow the neural network to remember longer contexts and deal with the vanishing and exploding gradient problems.

The forget gate determines how much of the past information is kept in the cell state C_t. That is done using a sigmoid function, defined as:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2.5}$$

where, W_f is a learnable weight associated with the gate, b_f is a bias term, which is also a learnable parameter, h_{t-1} is the hidden state of the previous timestep, x_t is the input at the current timestep and σ is a sigmoid function defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.6}$$

The output value of the gate is between 0 and 1, where 0 means that the gate is blocking all the information and 1 means that the gate is letting all the information pass through.

The input gate determines what new information is stored in the cell state. This gate also uses a sigmoid function and returns a value between 0, and 1. The input gate is defined as:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.7}$$

The output gate, similarly to the other gates, outputs values between 0 and 1. This gate decides which information from the current cell is passed to the output. The output gate is defined as:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.8}$$

After we have the three gates, we can calculate the cell input activation vector $\widetilde{c}_t$, similarly as we calculated the gates but instead of the sigmoid function, we have tanh:

$$\widetilde{c_t} = tanh(W_c[h_{t-1}, x_t] + b_c) \tag{2.9}$$

The cell state c_t, also known as the memory, can then be computed as:

$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c_t} \tag{2.10}$$

where, "$\circ$" represents element-wise multiplication.

Finally, the new hidden state h at timestep t can be computed as:

$$h_t = o_t \circ tanh(c_t) \tag{2.11}$$

Often, the input can be large and contain a lot of timesteps. This is especially the case when processing signals, such as speech. In that case, the number of computations becomes large and the training time is increased significantly. This can cause problems when dealing with large amounts of data, especially when the neural network has many layers. To overcome that, it is common to use a pyramidal LSTM architecture. This architecture reduces the time resolution by half in each consecutive layer. The reduction in resolution is usually done by taking every other timestep in each consecutive layer or averaging every two consecutive time steps. An example of a pyramidal bidirectional LSTM architecture is shown in figure 5.

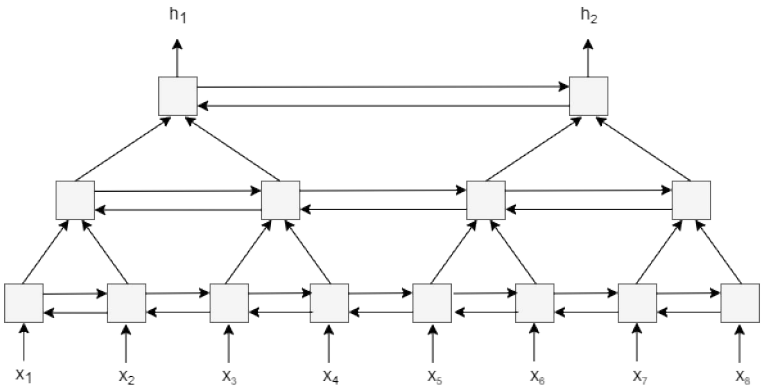

Figure 5: Pyramidal BLSTM network.

2.5 Multi-task learning

Multi-task learning involves optimization of more than one loss function. The motivation for doing multi-task learning can be taken from human life in general. When kids are young, they often start with learning smaller tasks that will help them later in learning other tasks. Walking can be seen as a task that needs to be learned first, for the person to learn how to run. Similarly, in basketball, if the main goal is to score, it can be helpful to learn also how to navigate with the ball, shoot, etc.

The above examples can be applied to machine learning as well. Often, an additional (auxiliary) task can be learned, which helps in achieving the main task. It is common for the additional task to just help with learning the main task, but that is not always the case. Sometimes, the additional task can be seen as equally important for the system, as we will see when constructing our multi-task learning model.

One big advantage of multi-task learning approaches is that they often involve parameter sharing. That can be helpful if we have a big system since it will reduce the computational time, in comparison to having two separate systems. In terms of parameter sharing, the multi-task learning approaches are divided into two categories: hard parameter sharing and soft parameter sharing approaches.

Hard parameter sharing is done by sharing some of the hidden layers, usually the lower ones, and having separate output layers for all the different tasks. This is the most commonly used approach. Since some of the layers are shared, this approach reduces the parameters of the model, in comparison to having to train separate models for all the specific tasks. This approach can also help with overfitting since it needs to find a good representation of all the tasks and not just focusing on one. In our experiments, we will be using a hard parameter sharing multi-task approach by learning to transcribe speech and annotate it with named entities. An example of hard parameter sharing can be seen in figure 6.

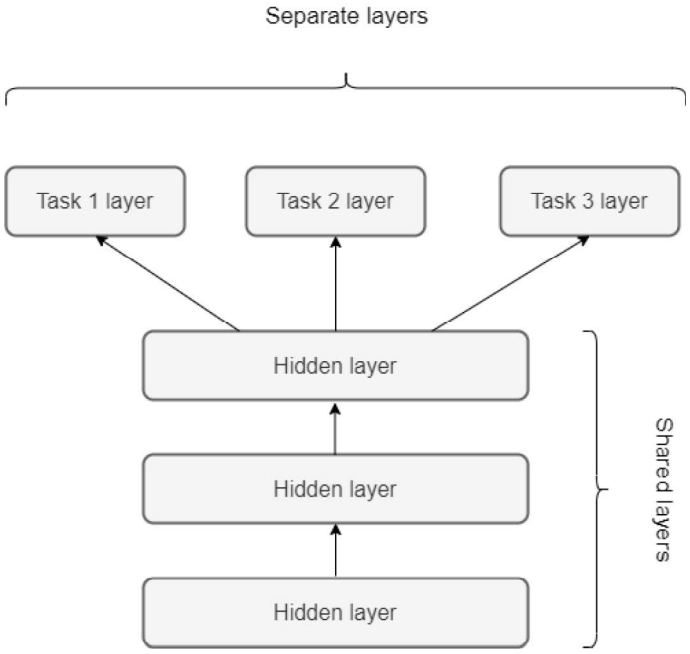

Figure 6: Multi-task learning with hard parameter sharing.

In soft parameter sharing, each of the tasks has its own layers and parameters. The main idea behind this approach is that the layers are regularized, meaning that the distance between each model's parameters is penalized, so that they can have similar parameters. This approach is more flexible by just loosely restricting the parameters of the models. An example of soft parameter sharing is presented in figure 7.

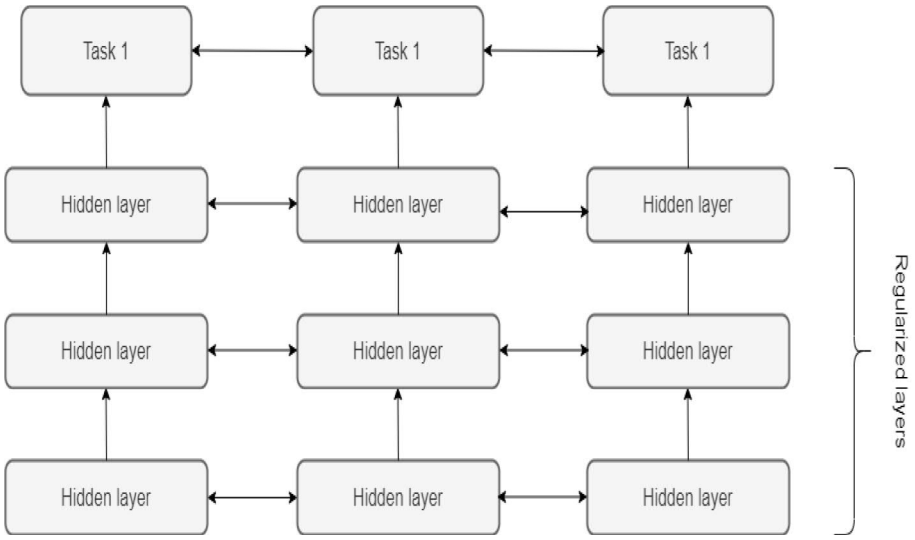

Figure 7: Multi-task learning with soft parameter sharing.

Even though the multi-task learning approaches seem appealing, they have some drawbacks. Since they involve optimizing multiple loss functions, we need to carefully

choose a weighting factor that determines the contribution of each of the separate losses. That is often not a trivial task and choosing a wrong weighting factor can have an undesirable impact on the learning process. For example, if we give more weight to one loss function, the model might focus on that part more and not learn the other one. Also, if one of the tasks is easier, it will be learned a lot faster and its contribution should have less impact in the later stages of the training.

3 Automatic speech recognition

This chapter focuses on the importance of automatic speech recognition (ASR) in our society. It also explains different types of automatic speech recognition systems and how they are used in various applications. Later, it dives into the different building blocks of speech recognition systems and ways of assessing their performance. Lastly, it gives an overview of previous research that was done in the field, as well as the strengths and weaknesses of various approaches for doing ASR. Some of these principles will be applied in our experiments when constructing the speech recognition part of the system.

3.1 Importance of ASR

Speech is a continuous audio stream composed of small chunks of sound called phones. The acoustic properties of these phones vary depending on the speaker style, gender, and age [9, 10]. These phones serve as main building blocks of larger chunks known as sub-words. The sub-words are then merged to form words. This process of mapping the speech signal to the corresponding transcript is done using automatic speech recognition.

Automatic speech recognition is the way for the computers to recognize and translate spoken language into text. It is a way for humans to interact with computers using speech the same way as humans interact with each other. The ASR systems play a vital role in hands-free interaction with various devices, such as smartphones, cars, home automation equipment, and computers. They are also used in medical, military, telephony, and various other fields.

3.2 Types of ASR systems

There are different ways of interaction with a speech recognition system. Depending on the way they are used, we can separate them into two groups: spoken dialogue and natural language systems.

There are different ways of interaction with a speech recognition system. Depending on the way they are used, we can separate them into two groups: spoken dialogue and natural language systems.

Spoken dialogue systems work in a way that they present the users with predefined options that they can choose from. This simplifies the speech recognition task by restricting the input to predefined words or phrases. Spoken dialogue systems are suited for tasks such as home automation, where we can have predefined commands like "turn on the light" or "unlock the door". One of the oldest applications involving spoken dialogue system is the JUPITER project [11], which allows users to obtain weather forecast information through their phones. These systems usually achieve good performance but are less flexible from a user's perspective and don't feel natural. It also requires the user to memorize the voice commands.

Natural language systems take as input any form of free speech and are not restricted to predefined phrases or words. These systems are easier to use since the

interaction is in a more natural way, similar to the way humans interact with each other. One of the oldest natural language systems is HARC [12], which operates in an airline information domain and has a vocabulary of 2000 words. Some more recent applications include personal assistants in computers and smartphones. Even though these systems seem appealing, developing them is more difficult. The conventional speech recognition systems rely on dictionaries and grammar. When a free speech is involved, the number of words increases, causing the system to misclassify them.

In subject to the way speech recognition systems are trained, they are divided into two groups: speaker-dependent (SD) and speaker-independent (SI) systems.

Speaker dependent systems are trained by the individuals that are going to use them. The advantage of these systems is that they tend to achieve high performance. They do that by learning the unique characteristics of different speakers. The disadvantage of these systems is that new speakers need to train the system to their voice, so that is can achieve good results.

Speaker independent systems are trained in a more general way, not relying on the individual patterns that people have when they speak. These systems are usually trained on a large variety of data including different genders, age groups, and environments. These systems are better suited for general purpose use but they tend to perform worse in comparison to speaker-dependent systems.

3.3 Building blocks of conventional ASR systems

The conventional ASR systems typically consist of several building blocks. These building blocks include feature extraction, acoustic model, language model, and lexicon. Each of these blocks needs to be created separately, which adds to the complexion of the whole system. An example of the building blocks of the conventional ASR system can be seen in figure 8.

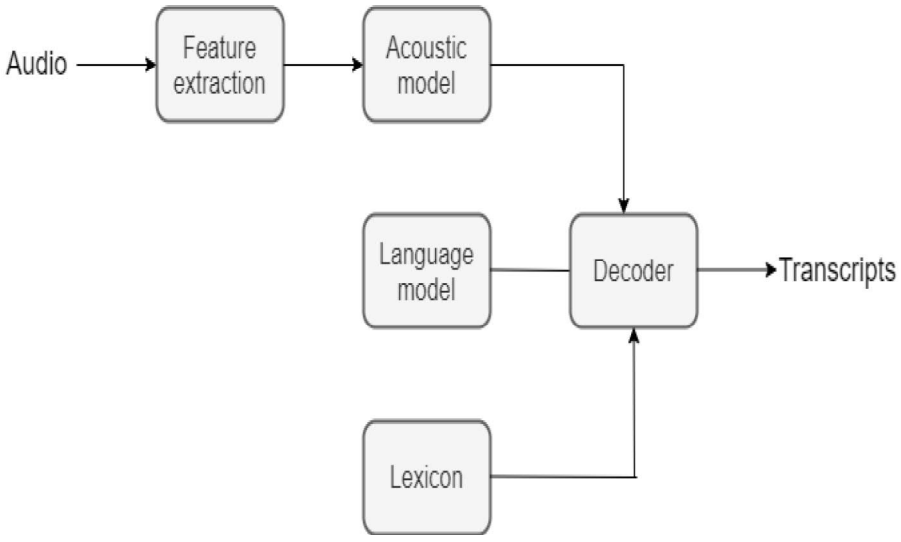

Figure 8: Conventional ASR components.

3.3.1 Feature extraction

The feature extraction process is an important part of the ASR systems. The most common features that are extracted from the audio waveform are Mel-frequency cepstral coefficients (MFCCs) and Mel-filter banks. The Mel-filter banks can be correlated, which may harm some machine learning algorithms, in which case the MFCCs are preferred. In our experiments, we will use logarithmic Mel-filter banks as input features to the network, because neural networks are more robust to correlated features. In the following part, we will explain how the Mel-filter banks and MFCCs are extracted.

The nature of the signals is such that their frequency changes over time. If we want to transform the signal from time to frequency domain, using a Fourier transform, we need some way of approximating the frequency. To do that, instead of processing the whole signal at once, we can split it into small frames, assuming that the frequency in each of the frames remains stationary. In speech, typically a frame length of 20ms - 40ms is used, with an overlapping window of 50% between the consecutive frames.

Since the Fourier transform assumes that we are dealing with an infinitely repeating signal, if the start and the end of the signal do not match, that can cause a discontinuity in the signal. To deal with that, a windowing function is applied to the signal. One of the most commonly used windowing functions is the Hamming window, defined as:

$$h[n] = 0.54 - 0.46 * \cos(\frac{2\pi n}{N-1}) \tag{3.1}$$

where, N is the window length and $0 \leq n \leq N-1$.

To compute the power spectrum of a signal, we can use a discrete Fourier transform (DFT). The DFT calculates the frequency spectrum of each of the frames as:

$$S_i(k) = \sum_{n=1}^{N} s_i(n)h(n)e^{-j2\pi kn/N} \qquad 1 \leq k \leq K \tag{3.2}$$

where, $h(n)$ is the Hamming window, $s_i(n)$ is the framed signal, n is the number of samples, i denotes the frame number and K is the length of the DFT.

The power spectrum is calculated as:

$$P_i(k) = \frac{|S_i(k)|^2}{N} \tag{3.3}$$

After we have the power spectrum, we can compute the Mel-filter banks. We do that by applying m number of triangular filters to the power spectrum. The number of filters usually ranges from 20 to 40. The filter banks can be calculated using the

following formula:

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \frac{k-f(m-1)}{f(m)-f(m-1)} & f(m-1) \leq k \leq f(m) \\ \frac{f(m+1)-k)}{f(m+1)-f(m)} & f(m) \leq k \leq f(m+1) \\ 0 & k > f(m+1) \end{cases} \tag{3.4}$$

where, $f()$ represents the boundary points. The first filter bank starts at the first point, reaches its peak of 1 at point two, and returns to 0 at the third point. The second filter bank starts at the second point, reaches its peak at the third point, and returns to 0 at the fourth point. This process goes for all the filter banks. In figure 9, we can see an example of 15 filter banks. In the end, we convert those filter banks to a logarithmic scale. This is done to mimic the human hearing since we don't hear loudness on a linear scale.

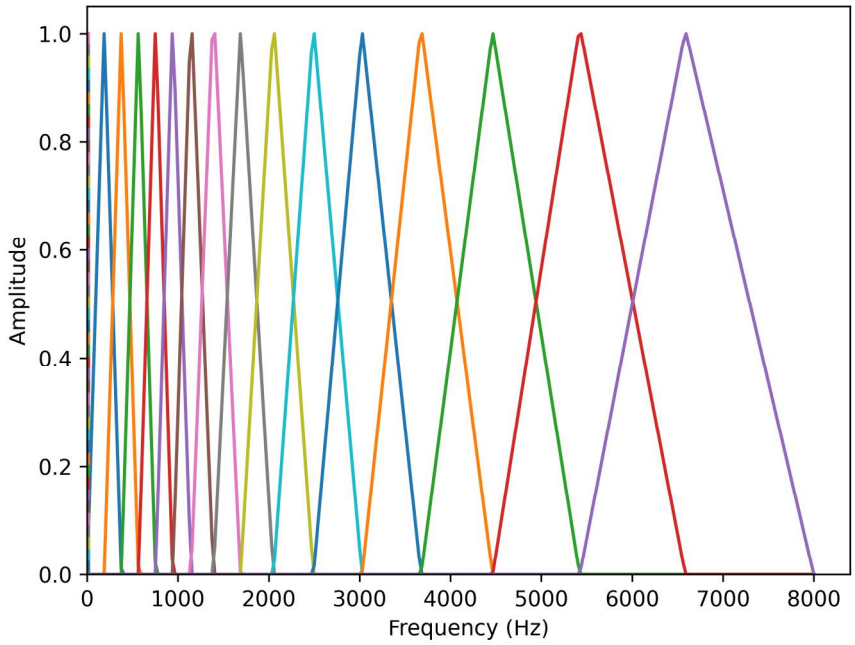

Figure 9: 15 Mel-filter banks.

Since we are working with an overlapping window, the filter bank energies are correlated, which can cause problems for some machine learning algorithms. To decorrelate them, we can apply a discrete cosine transform (DCT) and get the cepstral coefficients. When we are working with MFCCs, it is typical to keep only some part of the coefficients and discard the higher ones. The reason for that is because those coefficients represent fast changes in filter bank energies, which can harm the ASR performance.

3.3.2 Lexicon

The lexicon is usually developed by phonetic experts, using a phone set, which is specific for each language. The lexicon describes how different words are pronounced phonetically. Some words can have the same spelling but being pronounced differently. The lexicon contains information about all the alternative pronunciations.

The lexicon usually has two main roles in the ASR systems. Firstly, it covers what words will be known to the system. The larger vocabulary it has, the better coverage it will have, thus less out-of-vocabulary words.

The lexicon also provides a link between the acoustic model and the language model, by containing the phonetic pronunciations of each of the words.

3.3.3 Acoustic model

The acoustic model is responsible for modeling the acoustic information of the speech. It usually uses MFCC features, extracted from the raw audio.

The role of the acoustic model is to predict at each frame, which phone is being spoken. It is usually implemented using a hidden Markov model (HMM) or a combination between HMM and Gaussian mixture model (GMM) or a deep neural network (DNN).

The hidden Markov model is a probabilistic framework that is represented as a weighted directed graph that follows the Markov property. This means that the probability of moving from one state to another depends only on the current state. The term *hidden* in the hidden Markov model comes from the fact that the internal states are not visible and we only have the observations, which are the feature vectors. The probability of observing a feature vector, given the internal state is called emission probability, whereas, moving from one internal state to another is called transition probability. The HMM is a generative model, which means that it tries to learn the underlying distribution of the features, given the states.

Training an acoustic model requires large amounts of audio data and corresponding transcripts. The quality of the training data can have an impact on the performance of the acoustic model. We should take into account different genders, accents, background noise, and microphone types because that will affect what the acoustic model is capable of recognizing.

3.3.4 Language model

The language model is responsible for predicting which word is supposed to appear based on previous words. It outputs a probability for the next word based on a previous sequence. The language models are usually implemented using an n-gram model or a neural network.

Training the language model is done on a lot of textual data. Since the language model is learning the probabilities of the words from a training data, that can reflect on what kind of words will have a higher probability, since people's word choices are influenced by the topic. When training a language model, it is important to use

training data similar to what the language model is expected to encounter in the real world. The language models are explained in more detail in section 5.

3.4 End-to-end ASR

As the name suggests, the end-to-end ASR systems typically consist of one model that does all the work. In comparison to the conventional ASR systems, that are composed of acoustic model, language model, and lexicon, the E2E models incorporate everything into one model. One big advantage of the E2E models is that they don't require the audio and the transcripts to be aligned. There are two main approaches for doing E2E ASR, the connectionist temporal classification (CTC) and attention-based encoder-decoder.

3.4.1 Connectionist temporal classification

Speech recognition systems try to map an input audio sequence X to the corresponding output Y. This might seem trivial but some challenges need to be considered when designing such systems. One challenge is that the input and the output sequences can vary in length. That introduces another problem, we need to find a way to align the input and the output sequences. These issues are solved by the CTC algorithm [13].

CTC works in a way that it outputs a character for each timestep. This means that we end up with more output tokens than the original transcript. After the model generates the output, the consecutive characters that are repeated are removed. For example, if the correct word is "car" and one possible alignment is "ccarr", then, by collapsing the "c" and "r", we will end up with the correct word.

In some situations, collapsing all the repeating characters can be undesirable. If we have a word that contains characters that are consecutively repeated, for example, "balloon" and the alignment is "bballloon", then the output would be "balon", which is incorrect.

To overcome that, CTC introduces a special token ϵ, that the system can output at any timestep of the prediction. The repeating characters still get collapsed, unless the special token is between them. In the end, the special token is removed from the output. An example of that is shown in figure 10.

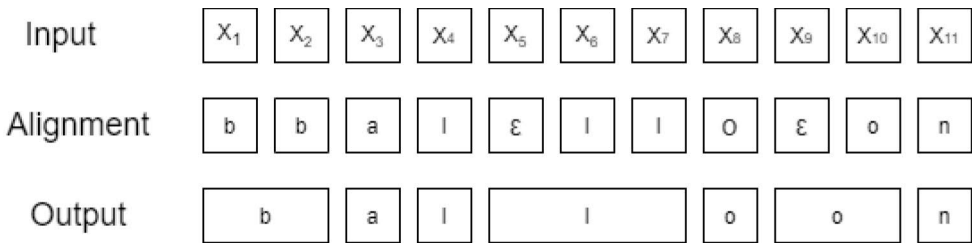

Figure 10: CTC algorithm

To get the probability of the output, given the input sequence, CTC sums all the possible alignments between the two. This approach is not efficient because there

are a lot of alignments that need to be considered. That is solved using a dynamic programming algorithm.

The CTC algorithm can work well in many cases but it also comes with some drawbacks. The CTC assumes that the outputs are conditionally independent of each other, which is a wrong assumption to make when dealing with speech. Another drawback is that the alignments are many-to-one mappings, where multiple inputs can be aligned to at most one output.

3.4.2 Attention-based encoder-decoder

Attention-based encoder-decoder architecture is another way of doing end-to-end speech recognition. Just like the CTC, this model also doesn't need alignment between the input and the output.

The main idea behind this method is that it takes acoustic features as inputs and produces characters as outputs. Each character output y_i is modeled as a conditional distribution over the previous characters and the input signal using the chain rule.

This model consists of two sub-modules: encoder and decoder. The encoder and decoder have two operations that they perform. The encoder processes the audio features and the decoder, which is attention-based does the decoding. The encoder takes the acoustic features and transforms them into a hidden representation h, which is then passed to the decoder. The decoder uses those hidden representations to produce probability distribution over character sequences. Both the encoder and the decoder usually consist of recurrent neural networks.

Since the acoustic signals can have a very large number of frames, training a conventional BLSTM is unfeasible. To reduce the length of the input sequence, a pyramidal BLSTM is used in the encoder.

The decoder uses an attention mechanism, together with an LSTM. The goal of the decoder is to produce a probability distribution over each character, conditioned on all the previous characters. Outputting character sequences makes the model more robust to out-of-vocabulary words. Another advantage of producing characters as output is that the model can produce multiple spelling variants.

This model treats the outputs as conditionally dependant, unlike the CTC, where the outputs are treated as they are independent of each other. This can be seen as an advantage over the CTC model.

The attention-based encoder-decoder model works reasonably well for ASR but it requires a lot of training data, in comparison to the conventional ASR systems.

The attention based encoder-decoder architecture is explained in more detail in section 6, since we are using it in our experiments.

3.5 Challenges in ASR

Constructing an ASR system has many challenges that need to be addressed for it to achieve good performance. The conditions in which the speech is recorded can affect the performance of the system significantly. In noisy environments such as crowded

places, it is hard even for humans to recognize speech correctly. Room reverberation can also have an impact on the quality of the speech signal.

The gender of the speakers can also have a significant impact on the system's performance. The difference between male and female speech was studied in detail in [14]. They observed that in most of the cases the fundamental frequency of female speakers was higher than the one in the male speakers. If the ASR system is exposed to only male or only female speakers, then it might have difficulties recognizing the opposite gender.

The age group of the speakers is another area that needs to be taken into consideration when designing an ASR system. Elderly people tend to have decreased speech rate and more speech disruptions compared to middle-aged people [15]. There is also a significant difference between children and mature speakers. The study done in [16] shows that the ratio between the vowel and consonant duration is higher for children. The study also points out that the confusion between different phones is higher in children's speech. Different age groups need to be taken into account when designing an ASR system that is going to be used by a variety of speakers.

The construction of a good ASR system requires a significant amount of data. This data is more available for high-resource languages such as English but for low-resource languages that can be difficult. Not having enough data can result in a higher number of out-of-vocabulary (OOV) words, which can degrade the performance of the system.

3.6 Speech recognition assessment

When developing a speech recognition system, it is important to have a way of measuring how well it performs in a real-world scenario. Knowing how well the system performs can help us distinguish between different algorithms and learn their strengths and weaknesses.

3.6.1 Word error rate

The goal of the speech recognition system is to generate transcripts for given audio. Thus, we need a way to measure the quality of the transcripts. One common way of assessing the performance of an ASR system is the word error rate metric. The WER metric is based on the *Levenshtein distance* [17], which measures the number of insertions, deletions, and substitutions in a string.

- Insertion occurs when a new word gets added to the generated transcript.
- Deletion occurs when a word is missing in the generated transcript but is present in the original.
- Substitution occurs when a word from the original transcript gets replaced with another in the generated transcript.

The word error rate can be computed as:

$$\begin{aligned} WER = & \frac{\text{Levenshtein distance}}{\text{Number of spoken words}} * 100 \\ & = \frac{\text{I} + \text{D} + \text{S}}{\text{N}} * 100 \end{aligned} \tag{3.5}$$

where I is the number of insertions, D is the number of deletions, S is the number of substitutions and N is the number of words in the reference (spoken words).

The word error rate is usually measured in percentage, where a lower percentage means higher quality transcripts. Since there are no restrictions on the number of insertions, the WER can exceed 100%.

Even though the word error rate is a good way of measuring the performance of an ASR system, it has its limitations. Often other external factors can affect the performance of the system, such as bad microphone, background noise, and pronunciation.

3.6.2 Processing time

Another aspect that is worth considering when evaluating an ASR system is the processing time. Usually, bigger models can achieve better WER but that can affect the processing time of the model and make it unusable in a real-time scenario. The processing time is usually estimated as a *real-time factor* and one needs to make a trade-off between that and the WER, according to the environment in which the system is going to be used.

3.7 Previous research on ASR

The earliest attempts of doing automatic speech recognition go back to the 1950s. In 1952, one of the first automatic speech recognition systems was developed [18]. This system was able to recognize spoken digits with high accuracy. Since then, the paradigm of doing automatic speech recognition shifted from pattern matching to statistical, especially with the hidden Markov model framework. HHM was successfully applied in automatic keyword spotting in unconstrained speech, achieving high accuracy [19]. In [20], the authors developed a speech recognition system that can handle speech contaminated with noise. To achieve that, they used signal decomposition using hidden Markov models [21].

The HMM framework was further extended with the Gaussian mixture model. This hybrid HMM-GMM model was successfully applied in various ASR tasks. In [22], the authors used a hybrid HMM-GMM model to develop a voice command system for a robot arm. They showed that better recognition results can be achieved using this hybrid approach. An ASR for recognizing 20 spoken Hindi words was developed using the HMM-GMM model, achieving better performance than the standard HMM model [23]. HMM-GMM model for large vocabulary word recognition was explored in [24]. The authors showed that with high enough training data, recognizers using

mixture HMM outperform the ones that use unimodal Gaussian HMM by a large margin. In [25], the authors applied the HMM-GMM model by doing automatic speech recognition for Arabic language. The HMM-GMM models played a dominant role in ASR for a long time. That changed with the popularization of the deep neural networks.

With the advancement of deep neural networks, new approaches for ASR were developed. A new context-dependent DNN-HMM (CD-DNN-HMM) model was proposed, outperforming the previous CD-GMM-HMM models by a significant margin [26].

All of the above methods fall into the category of conventional ASR systems. Even though these systems achieve high accuracy, especially the CD-DNN-HMM variant, their complexity makes them less accessible. They are usually composed of an acoustic model, a language model, and a lexicon. All of these components need to be trained separately, which adds to the complexity. Feature extraction is another drawback of these systems that requires domain knowledge.

To address the shortcomings of the conventional ASR systems, a new generation of end-to-end ASR systems was developed. These E2E systems rely heavily on neural networks, alleviating the need for hand-crafted features.

A novel method, relying on the connectionist temporal classification loss was proposed, outperforming the standard HMM and HMM-RNN approaches [13]. This approach does not require alignment between the input and the output, which is a challenge in the conventional methods. Another advantage of this approach is that it directly outputs the transcripts, unlike the conventional systems that usually output phonemes and other small units and then do post-processing to obtain the transcripts. One drawback of this approach is that it assumes independence between output sequences. Another drawback is that it can be viewed as an acoustic-only model because it defines a phoneme distribution that depends on the acoustic input sequence.

To incorporate a language model into CTC, a recurrent neural network (RNN) was incorporated into the system, known as RNN Transducer. [27]. This approach was explored in [28], achieving state-of-the-art results on the TIMIT dataset [29]. In [30], the authors explored different ways of improving the RNN Transducer model. In their experiments, they used deep recurrent neural network and sub-word units instead of whole word units in the output. Their model also uses hierarchical pre-training with CTC. Even though the RNN Transducer solves some of the issues that the CTC model has, it still faces some other challenges. This model can produce unreasonable paths that are hard to avoid.

Another E2E approach that gained popularity in recent years is the Listen, Attend and Spell architecture, presented in [31]. This architecture builds on top of the sequence-to-sequence model presented in [32] and its improved attention-based variant [33]. Unlike the CTC model, the attention-based sequence-to-sequence model does not make the independence assumption. This approach, just like CTC, does not require pre-segmented alignments. The soft alignment is learned using the attention mechanism. Besides the advantages mentioned above, this model also has some shortcomings. The alignment that this model does is flexible, which can result in

insertion and deletion errors. It can also prematurely stop the decoding process.

To deal with the shortcomings that the attention-based model has, a multi-task approach was proposed. In [34], the authors proposed a hybrid CTC and attention-based model. For training, they used a multi-task learning approach by optimizing two objective functions. For the decoding part, they combined both the attention-based and CTC scores. This approach achieved competitive results, in comparison to the previous state-of-the-art conventional system. Similar hybrid CTC/Attention architecture was applied for audio-visual speech recognition, achieving state-of-the-art results [35].

4 Named entity recognition

This chapter focuses on the importance of named entity recognition. It also explores different types of NER systems, their strengths, and weaknesses, as well as how they are applied in various fields. Furthermore, it explores a common architecture for doing NER, which uses BLSTM with a CRF layer on top, as well as the challenges that arise when designing a NER system. In the end, it delves into different ways of assessing the performance of the NER system and exploring previous research that was done in this area.

4.1 Importance of NER

Language is the main source of communication that humans use. We use language to express ideas, feelings, intentions, among many other things. Being able to communicate those thoughts with a computer has been an open area of research since the 1950s, which gave rise to the natural language processing field. NLP is a way for computers to process and analyze large amounts of natural language data. It can be viewed as a way for humans to interact with machines in the same was as humans interact with each other.

Languages have ambiguous nature. Same words can have a different meaning, depending on the context they are referred to. This is more formally known as lexical ambiguity. Another type of ambiguity that happens in natural languages is syntactic ambiguity, where the same sentence can have different meanings. People use language throughout their entire lives, so resolving these ambiguities does not cause difficulties, which is not the case for computers.

When we, as humans, hear or read a sentence, we immediately associate it with its meaning. The process of extracting meaning from a sentence is known as semantic analysis. The extraction of meaning is not bound to sentence-level only but it can be done on a word or sub-word level as well, which is known as lexical semantics. Semantic analysis is one of the key components when doing natural language understanding or spoken language understanding. The difference between these two terms is that NLU tries to extract meaning from text, whereas SLU does it from speech.

One of the first things when doing natural language or spoken language understanding is to do named entity recognition. It also serves as one of the main building blocks in extracting semantic information from documents. NER is also an integral part of other larger areas such as question answering, information extraction, machine translation, and text summarization.

In the question answering task, the goal is to find an answer in a paragraph of text. It is important to detect the named entities in the text because answers are often named entities. That is especially the case in the fact-based answers to questions. In that sense, most of the traditional question answering systems incorporate some form of named entity recognizer, which simplifies the task significantly.

When doing information extraction, many of the relations are associations between named entities. Being able to detect them is important for the system so that it can extract the relevant information. Misclassification of a named entity might lead to

wrong information being extracted.

Named entities have an important role in machine translation as well. The system needs to recognize them correctly because incorrectly translating or dropping a named entity can change the meaning of the sentence.

In text summarization, the goal is to extract relevant information from documents. The relevant information usually includes dates, locations, people, and organizations. All these categories can be detected using a NER system. This will make sure that the system won't exclude relevant information in the summarization.

4.2 Types of NER systems

There are different approaches of building named entity recognition system and the most common ones are rule-based, learning-based, and hybrid approaches. Each of these approaches has its own advantages and disadvantages.

Rule-based systems require hand-crafted features such as dictionaries, lexical and syntactic patterns. These systems usually need domain knowledge in order to construct the rules, which is expensive and time-consuming. The hand-crafted features are usually implemented using regular expressions.

The advantage of these systems is that they can be implemented with a small amount of data and they work well in domains where the text is formal, following the standard rules of the language.

These systems tend to have high precision, which means that if they detect a named entity, they will most probably classify it in the correct category. On the other hand, the rule-based systems may struggle to detect the named entities due to the informal nature of some texts that exhibit the grammatical rules of the language, which can result in a low recall.

Learning-based systems are trained using machine learning methods. The learning algorithms work in a way that they take as input training data features and learn patterns that emerge from them. These patterns are then used to predict new, unseen data. In the case of NER, each word is a token and a sentence can be viewed as a sequence of tokens. The task of predicting a sequence of tokens is known as a sequence tagging task. These tokens are usually represented as multidimensional vectors that describe them. The tokens are fed to a machine learning model that is trained to do a sequence tagging task.

The advantage of these systems is that they don't rely on hand-crafted features, thus unstructured texts can cause fewer difficulties compared to the rule-based systems. This may result in a high recall at expense of lower precision. The disadvantage of the learning-based systems is that they tend to require a lot of data to be able to learn and achieve good results. That data may not be always available, especially in low-resource languages and specific domains. In our experiments, we will be using this approach for doing NER.

Hybrid systems aim at combining the good characteristics of both rule-based and learning-based systems. This approach can yield good results for languages that have difficult syntax and semantics. By utilizing the patterns learned by the machine learning model, the system can detect the entities, which can result in a high recall.

The rule-based system on the other hand can correctly classify those entities in their categories, which can yield high precision. The hybrid approach may seem appealing considering that they combine the best of both worlds but constructing these systems requires a significant amount of time and resources.

4.3 BLSTM-CRF architecture for NER

The most common way of doing named entity recognition is using a bidirectional LSTM network with a CRF layer on top of it. This architecture gives good results since both BLSTM and CRF models are well suited for processing sequential data.

The BLSTM network usually processes a sentence at a time and the CRF layer does the named entity annotation for each sentence. The words in a sentence are replaced with n-dimensional vector representations, that describe those words. The vector representations of words are called word embeddings and they are described in more detail in section 5.

The BLSTM processes each word in a sentence and passes that information to the next word. In the end, the output of the BLSTM is passed through a fully connected layer that has an output size equal to the number of named entity classes. That information is then passed to the CRF layer, which produces tag probabilities for the whole sentence.

4.4 Challenges in NER

Several challenges need to be addressed when designing a named entity recognition system. The NER task requires the system to be able to disambiguate between entities. For example, the entity "YouTube" can refer to both company or product, depending on the context it appears in. Named entity disambiguation is a sub-task of NER that addresses that issue.

Personal names also pose certain challenges for the NER system. There are large varieties of personal names that the system hasn't seen but should be able to detect. Also, there are not many constraints of what a personal named can be, which can cause the system to ignore it or mistake it for another entity.

Named entities are usually capitalized, so the system relies on capitalization to detect them. That is usually the case when dealing with formal texts but in informal texts, capitalization can be neglected and cause degradation in the performance of the system. Informal texts also tend to have an incorrect sentence structure, abbreviations, and slang, which affect the performance of the system.

To achieve good results, NER systems require a significant amount of data. For high-resource languages such as English and Chinese, obtaining data is not difficult, which is not the case for low-resource languages, such as Finnish. Data sparsity can also happen in specific domains such as biology and chemistry, where there is not enough annotated data.

4.5 Assessment of named entity recognition systems

Having a proper evaluation metric for the named entity recognition system can help us analyze the strengths and weaknesses of the system. It can also help us distinguish between various architectures and hyper-parameters. Named entity recognition is a sequence tagging task that differs from speech recognition. Thus, we need a different way of assessing the quality of the named entity recognition system.

4.5.1 Precision, recall and F1

As a starting point we can consider the *accuracy* metric, which is calculated as:

$$accuracy = \frac{\text{correct predictions}}{\text{total predictions}} \tag{4.1}$$

This metric might look good at first glance but it comes with certain drawbacks. If we consider a case where there is a class imbalance, for all the examples the model can predict the majority class and achieve high accuracy, even though it struggles to predict the minority class. In a real-world scenario, we can consider a case where we have a classifier that predicts frauds. In this case, most of the time there will be no fraud, so if the model predicts no fraud all the time, it will still achieve high accuracy, which is highly undesirable. This is known as *accuracy paradox*. To mitigate this, we can use *precision*, *recall* or *F1 score*.

Precision takes into account how many of the positive predictions are truly positive. It is calculated as:

$$precision = \frac{\text{number of correct positive predictions}}{\text{total number of positive predictions}} \tag{4.2}$$

The precision score can be interpreted as a way of measuring how well the system can predict the cases that it has detected. A high precision score is important for systems where the cost of false positives is high. For example, if we have a classifier that predicts if a person has committed fraud, we don't want to have a false positive case because that can result in an innocent person being convicted.

Recall measures how many of the positive cases are captured by the model. It is calculated as:

$$recall = \frac{\text{number of correct positive predictions}}{\text{total number of positive samples}} \tag{4.3}$$

The recall can be interpreted as a way of determining how well the model can find a positive case. A high recall score is important when the cost of having false negatives is high. If we consider again the case of fraud detection, we don't want the person that has committed the fraud to be ignored and classified as non-fraudulent.

F1 is a measurement that combines precision and recall as follows:

$$F1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{4.4}$$

This measurement takes a balance between precision and recall and is commonly used in information retrieval and sequence tagging tasks, especially when dealing with imbalanced class distribution.

4.6 Previous research on NER

Named entity recognition was first introduced as a task in 1996 on the Sixth Message Understanding Conference [36]. Since then, many researchers tried to improve and develop new methods for doing named entity recognition. These approaches include various rule-based, statistical, and machine learning methods.

The traditional named entity recognition systems were mostly rule-based, relying on hand-crafted features and gazetteers (geographical dictionaries). These rule-based systems were successfully applied in various tasks and languages. In [37], the authors developed a rule-based named entity recognition system for Greek financial texts. A combination of clustering and gazetteers was explored in [38] by doing named entity recognition for Japanese language. The authors of [39] explored NER for low-resource Urdu language. Even though these methods achieve good results, they are mostly domain-specific and hard to adapt to new domains. The construction of gazetteers, especially for low-resource languages is challenging because there is a little amount of data available. These systems also suffer from lower recall and are unable to disambiguate between entities.

With the rise of machine learning popularity, new methods were developed for named entity recognition, which alleviate the use of external lexicons and gazetteers. The most successful ones are maximum entropy (ME) models and conditional random fields [40]. The maximum entropy named entity (MENE) model was first introduced in [41]. In [42], the authors tried to utilize the global document information when doing named entity recognition using the maximum entropy model. Conditional random fields also showed great success in various named entity recognition tasks. The authors of [43] were one of the first to apply CRFs in the named entity recognition task. In [44], CRFs were used to detect named entities in biomedical field.

Even though ME and CRF methods achieve good results, they still require domain knowledge to do feature engineering. This issue is directly addressed by deep neural networks, that became more appealing with the increase of the computational power. Recurrent neural networks are well suited for sequential data due to their ability to store information about sequences. RNNs, especially the LSTM variant, gained popularity in the NLP field after being successfully applied in language modeling, outperforming the current state of the art backoff language model [45]. In [46], the authors applied LSTM network for NER in clinical texts.

In 2015, a new architecture was proposed that combines LSTM with a CRF layer on top of it [47]. This architecture outperformed the previous neural network architectures and achieved state of the art results on the CoNLL 2003 dataset [48].

Recurrent neural networks usually take vector representation of words as input, known as word embeddings. This works well for languages that don't have a big vocabulary but morphologically rich languages have a large vocabulary that is hard to cover. This causes an increase in out-of-vocabulary words, which degrades the performance of the system. To overcome this, sub-word units were proposed instead of whole words [49].

To deal with the OOV words some researchers experimented with character-level sub-word units. Character-level LSTM was applied for the NER task, achieving competitive results [50]. In [51], the authors incorporated word and character embeddings into their LSTM network with a CRF layer on top of it. This approach achieved state of the art results on two biomedical datasets. The authors of [52] extracted character embeddings using a convolutional neural network and combined them with word embeddings. This approach achieved state-of-the-art results on CoNLL 2003 dataset. A similar approach of utilizing word and character embeddings was applied in [53], for doing NER for Spanish and Portuguese languages.

Another common approach for dealing with OOV words is to segment the words into morphs. This approach improved the performance of language models [54, 55] by constructing the unseen words from morphs. In [56], the authors explored the effects of morphology in the morphologically rich Turkish language when doing NER.

In recent years, pre-trained language models became famous for performing well on various NLP tasks. In 2018, Google presented its transformer architecture BERT [57], which gave competitive results on the NER task, compared to the state-of-the-art at that time. With the introduction of the ELMo word representations [58], the authors managed to get new state-of-the-art results on the CoNLL 2012 NER task. Since then, many researchers are shifting their attention towards transformer models. These models work well and achieve good results on various NLP tasks, but they come with several drawbacks. They require a lot of data, which is not always available, especially for low-resource languages. Their computational time is another challenge, which makes them not easily accessible to everyone.

Neural network approaches usually yield state-of-the-art results when dealing with high-resource languages where there is a lot of annotated corpora. For low-resource languages that is not the case and the performance of the model suffers due to lack of data. To improve the performance of the system in a low-resource scenario, various knowledge transfer techniques were proposed. In [59], the authors constructed a NER system for low-resource Dutch and Spanish languages by using a bilingual lexicon. They achieved that by transferring knowledge from high-resource to low-resource language. A similar approach was explored in [60], where the authors used bilingual embeddings to transfer knowledge by doing a nearest neighbor search from high-resource to low-resource language.

Named entity recognition is one of the main building blocks when doing spoken language understanding, where the goal of the system is to understand what has been spoken. The most common approach for extracting named entities from speech is through a pipeline approach, as described in figure 11. In this approach, the speech recognition system generates transcripts, and the named entity recognition system tries to detect entities in those transcripts. The output of the ASR systems

is usually lower-cased and noisy, in a sense that the word order can be mixed, words might be missing or misspelled, etc. When developing a NER system for speech data, these factors need to be taken into account. Some researchers tried restoring the capitalization and the punctuation from the transcribed speech as in [61]. A maximum entropy model was used for doing NER on transcripts generated by a speech recognition system for Chinese, utilizing n-best lists [62]. These approaches improve the performance of the system on noisy speech data but they are are still sensitive to the speech recognition output and error propagation. To deal with that, an end-to-end approach was proposed that directly extracts named entities from speech [63].

Extracting named entities directly from speech in an end-to-end manner is researched very little. This thesis will focus on extracting named entities from spoken Finnish language by combining the speech recognition and named entity recognition systems into one end-to-end model.

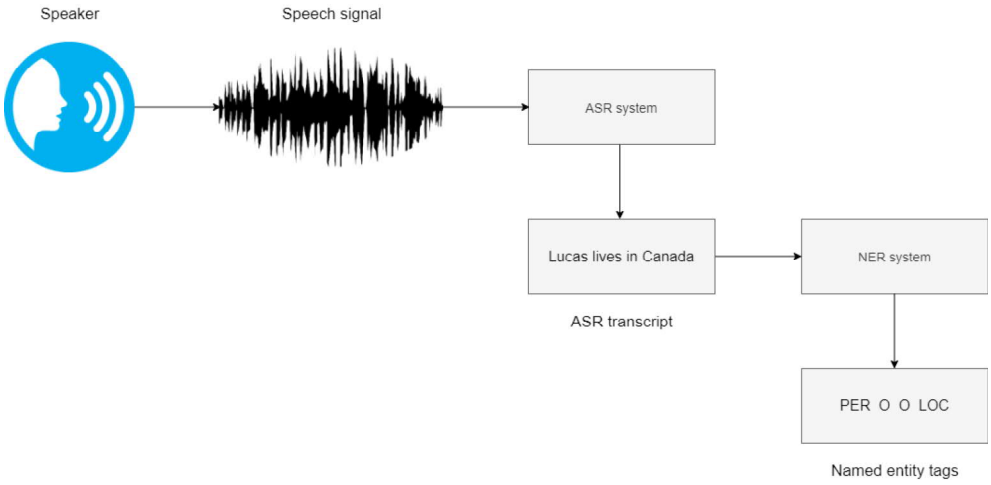

Figure 11: Pipeline NER system.

5 Language modeling

The language model (LM) can be used to compute the probability of a sentence or a sequence of words $P(W) = P(w_1, w_2, ..., w_n)$. It can also be used to compute the probability of the next word, given a sequence of words $P(w_3|w_1, w_2)$. The language model is one of the essential building blocks when making a conventional speech recognition system. Even though the end-to-end models don't necessarily require a language model, it can still be used to improve the results. The most common language model types are the n-gram and the neural network-based models.

5.1 N-gram language models

To calculate the probability of a sequence of words $P(W)$, we can use the chain rule to compute the conditional probability of each word w_i given the previous words $w_1, w_2, ..., w_{i-1}$.

$$\begin{aligned} P(W) =& P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)...P(w_n|w_1, w_2, w_3, ..., w_{n-1}) \\ =& \prod_i P(w_i|w_1, w_2, w_3, ..., w_{i-1}) \end{aligned} \tag{5.1}$$

A naive way of estimating these probabilities is to count and divide:

$$P(w_i|w_1, w_2, ..., w_{i-1}) = \frac{count(P(w_1, w_2, ..., w_i))}{count(P(w_1, w_2, .., w_{i-1}))} \tag{5.2}$$

This naive approach is unfeasible because there are too many sentences and we will need a lot of data to estimate all of them. To overcome this issue, we can consider only the $n - 1$, instead of the whole history:

$$P(w_i|w_1, w_2, ..., w_{i-1}) \approx P(w_i|w_{i-2}, w_{i-1}) \tag{5.3}$$

The probabilities can then be estimated as shown in Equation 5.4. This way we do an approximation of the probability of a word by taking part of the history.

$$P(w_i|w_{i-2}, w_{i-1}) = \frac{count(P(w_{i-2}, w_{i-1}, w_i))}{count(P(w_{i-2}, w_{i-1}))} \tag{5.4}$$

Even though the n-gram language models work well, they assign zero probability to word combinations that do not appear in the training set but are present in the test set. To overcome that issue, various smoothing techniques were developed, which take part of the probability of the occurring n-grams and assign it to the unseen combinations. One of the most widely used smoothing techniques is the Kneser-Ney smoothing [64]. Another drawback that n-gram language models have is that they can struggle to capture long-span dependencies between words. These types of language models are also not robust to word order changes. The issues mentioned above are addressed by neural network language models.

5.2 Neural network language models

The neural network language models typically use a recurrent neural network, more specifically the LSTM variant. The goal of this language model is to compute the probability of the next word given the previous words. The advantage of this type of language models is that they model long-term dependencies, unlike the n-gram language models. These models rely on learning distributed representations of words (or sub-words), which can make them more robust to word order changes. Words are typically represented as vectors of real values, where those values are learned in a way that similar words lie close in the shared vector space. Those vector representations of words are called word embeddings.

Word embeddings are typically pre-trained but they can also be learned during training. Depending on the way they are trained, there are different types of word embeddings, where the most common ones include word2vec [65], fastText [66] and Glove [67].

The fastText embeddings work particularly well for large-vocabulary languages, such as Finnish. Often, some words are not present in the vocabulary, so there is no vector representation for those words. Those words usually get a random vector assigned to them, which is not desired. This issue is solved with the fastText embeddings because they can utilize the sub-word information of the words, so even if the whole word is not present, its sub-word units may be present in the vocabulary.

The recurrent neural network language models are trained on large amounts of textual data. That data should be in the same domain as what the language model is expected to encounter in the real world. The network learns in a way that it processes each token (word) in the sentence and produces a probability distribution over the whole vocabulary. The word with the highest probability is usually the one that is chosen as the next word. The context of the previous words is taken into account when predicting the next one.

The training of neural network language models can be done on a word or a sub-word level. When training a word-level language model, we make sure that the output of the model will be a valid word. One big drawback of the word-level language models is that they are not robust to OOV words. This can cause problems for agglutinative languages like Finnish, that have a big vocabulary, which results in a high number of OOV words. To deal with the OOV issue, sub-word-based language models were developed, which use characters or other bigger sub-word units as output, instead of whole words. This way the OOV words can be modeled using the individual characters or other bigger sub-word units. Since the sub-word-based language models output characters or bigger sub-word units, instead of whole words, the output can have misspelled words.

The RNNs work better at capturing long-term dependencies compared to the n-gram models but still struggle with very large contexts. To overcome that, a new family of language models, called Transformers, was developed. The Transformer models typically rely on attention mechanism to capture the long-term dependencies. One of the first Transformer models that was developed is called BERT [57]. This model does not use any kind of recurrence and only relies on the attention mechanism.

This model was trained on the whole English Wikipedia data. Another popular Transformer is the Transformer-XL model [68], which uses recurrence together with an attention mechanism. According to the authors, this language model can capture 80% longer dependencies, compared to the standard RNN models. These models usually outperform the standard recurrent neural network language models but they require a lot of training data, as well as computational power.

5.3 Language models in E2E ASR

Unlike the conventional ASR systems that require a language model, that is not the case for the end-to-end approaches. Even though these approaches do not require an external LM, researchers experimented with integrating one so that they can achieve better results [69]. Integrating an external language model into the ASR is called language model fusion. The most common fusion approaches are shallow fusion [70], deep fusion [70] and cold fusion [71].

The shallow fusion approach incorporates an external language model at inference time by log-linear interpolation:

$$y^* = \arg\max_y [\log P(y|x) + \beta \log P_{LM}(y)] \tag{5.5}$$

where $\log P(y|x)$ is the beam search output, $\log P_{LM}(y)$ is the language model output and β is the weighting factor for the language model.

The deep fusion approach, like shallow fusion, assumes that the ASR system and the external language model are pre-trained. The difference between the two is that in deep fusion the external language model is integrated by fusing the hidden states of the decoder and the language model.

The cold fusion approach is an early training integration approach, where the ASR system is trained together with the pre-trained language model.

6 Methods

To do E2E named entity extraction from spoken Finnish, we will explore two approaches. In the first approach, we will build an attention-based encoder-decoder (AED) model for ASR by augmenting the labels with NER tags. In the second approach, we will explore multi-task learning where the model simultaneously learns to transcribe speech and annotate it with named entity tags. The pipelines of both approaches are explained in Figures 12 and 13 respectively.

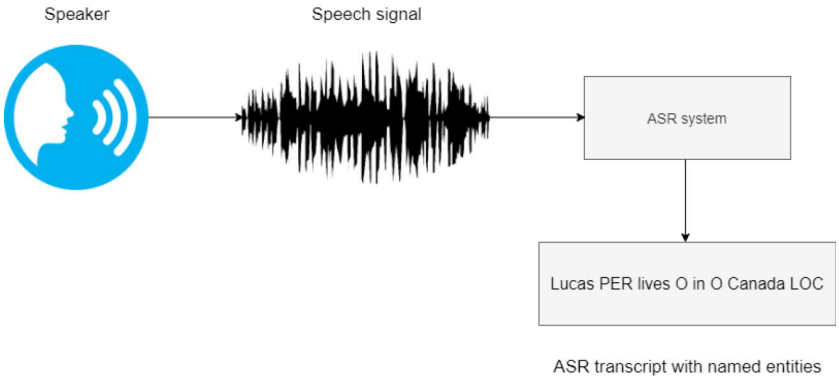

Figure 12: E2E NER system with augmented labels.

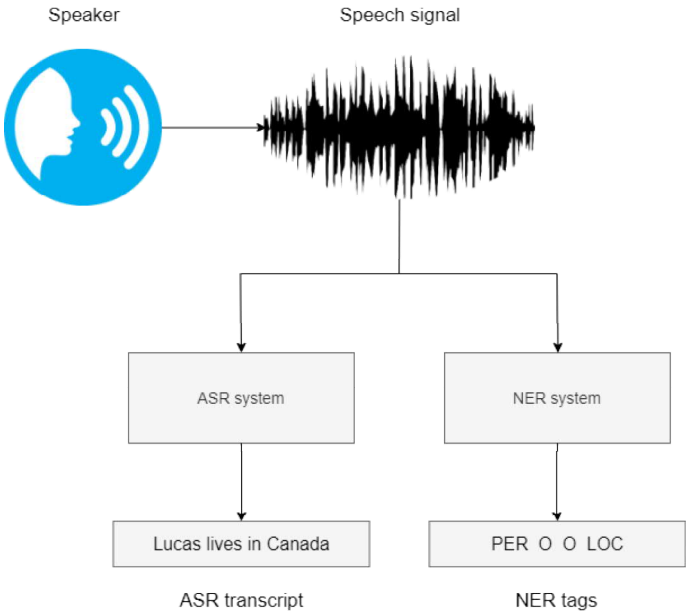

Figure 13: E2E NER system with multi-task learning.

6.1 Baseline NER system

The baseline system that we used to annotate the data with named entity tags is a BLSTM neural network with a CRF layer on top. This architecture utilizes morph,

character, and word embeddings, where the word embeddings are pre-trained and the morph and character ones are learned during training. Each of the embeddings goes into a separate BLSTM and the outputs are then concatenated together. The concatenated output goes through a highway layer, followed by a fully-connected layer. The output of the fully-connected layer is passed to a CRF layer, which produces tag probabilities. The baseline architecture is explained in mode detail in [72]. The architecture is depicted in figure 14.

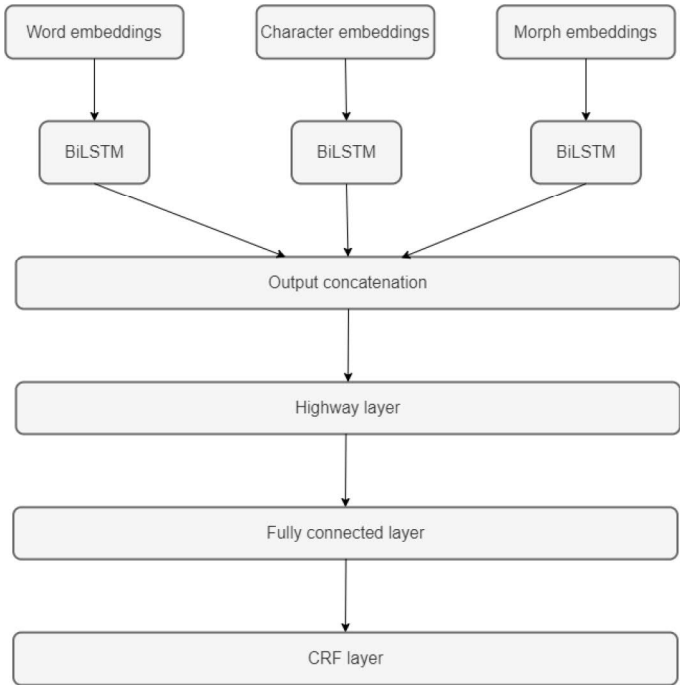

Figure 14: Baseline NER system architecture.

6.2 ASR with augmented labels

For this approach we developed an attention-based encoder-decoder architecture that takes audio features as input and produces transcripts with named entity tags. As a starting point, we can consider the standard encoder-decoder architecture, without attention. Let $X = (x_1, x_2, ..., x_T)$ be the audio features, where each feature is represented as x_i and i is the order of the feature. Additionally, we define the output character set $Y = (y_1, y_2, ..., y_T)$, where y consists of all the characters plus the special tokens: $< UNK >$, $< sos >$, $< eos >$, O, PER, LOC and ORG. The goal is to model the conditional probability:

$$P(Y|X) = \prod_i P(y_i|Y_{<i}, X) \tag{6.1}$$

It predicts the i-th output character, given the previous characters and the input features X. It does this using an encoder and a decoder.

The encoder is a recurrent neural network that computes the hidden states h_i for each timestep t as follows:

$$h_{enc_t} = f(W^{hh}h_{t-1} + W^{hx}x_t) \tag{6.2}$$

where W^{hh} and W^{hx} are learnable parameters, h_{t-1} is the hidden state of the previous timestep and x_t is the input feature of the current timestep. The final hidden state contains information from all the previous timesteps. The encoder compresses the input features in a single hidden representation. This hidden representation is then used to initialize the decoder.

In the standard encoder-decoder architectures, the decoder consists of another recurrent neural network, which computes the hidden states h_t as:

$$h_{dec_t} = f(W^{hh}h_{t-1}) \tag{6.3}$$

From equation 6.3 we can see that to compute the current hidden state h_t, the decoder uses only the previous hidden state h_{t-1}. To get the output label y_t for the timestep t, a softmax function is applied to create output distribution of the labels:

$$y_t = softmax(W^s h_t) \tag{6.4}$$

where, W^s is a learnable weight.

The drawback of the standard encoder-decoder architecture is that the encoder network needs to compress the entire input sequence into a single vector representation. This can affect the performance of the model, especially when dealing with long sequences, which is the case for speech features. The attention mechanism tries to mitigate this problem. The attention-based encoder-decoder architecture is presented in figure 15.

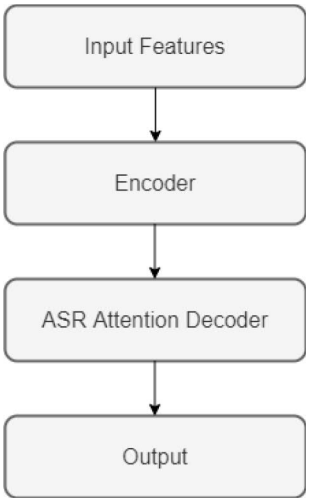

Figure 15: Model architecture for ASR with augmented labels.

6.3 Attention mechanism

The issue with the encoder compressing the entire input sequence into a single vector is addressed by the attention mechanism. The attention mechanism allows the decoder to "focus" on different parts of the input sequence at each step of the decoding. There are many different types of attention mechanisms and they are divided into two major groups: Luong attention [73] and Bahdanau attention [33].

6.3.1 Luong attention

To add Luong attention to the decoder, we first need to calculate the decoder hidden states. The hidden states are calculated using the previous decoder hidden state and the output of the embedding layer, similar to Equation 6.2:

$$h_{dec} = f(W^{hh}h_{t-1} + W^{hx}x_t) \tag{6.5}$$

where h_{t-1} is the previous hidden state and x_t is the output of the embedding layer. To apply attention to the decoder, we need to calculate the alignment vector α_{ts} as:

$$\alpha_{ts} = softmax(score(h_{enc}, h_{dec})) \tag{6.6}$$

where h_{enc} is the encoder hidden state and h_{dec} is the decoder hidden state. The Luong attention mechanisms are divided into several categories, depending on the scoring function that they use.

Dot attention computes the scores by doing a dot product between the hidden states of the encoder and the decoder. The scoring function for the dot attention is defined as:

$$score(h_{enc}, h_{enc}) = \langle h_{enc}, h_{dec} \rangle \tag{6.7}$$

General attention is similar to the dot attention with the only difference being that the result from multiplying the hidden states of the encoder and the decoder is additionally multiplied by a weight matrix. The weight matrix W is calculated by passing the decoder hidden states through a fully connected layer. The scoring function for general attention is defined as:

$$score(h_{enc}, h_{dec}) = W(h_{enc} * h_{dec}) \tag{6.8}$$

Concat attention works in a way that first the hidden states of the encoder and the decoder are summed up and then passed through a fully connected layer. The fully connected layer is then passed through a *tanh* nonlinearity and finally multiplied by a weight vector v. The concat scoring function is defined as:

$$score(h_{enc}, h_{dec}) = v * tanh(W_{combined}(h_{enc} + h_{dec})) \tag{6.9}$$

Hybrid + location-aware attention works in a way that it also takes into account the location of the elements, which is not the case for the previous scoring functions. By taking the location into account, different scores can be produced for the same

element appearing in different positions. The scoring function of this type of attention is calculated as:

$$score(h_{enc}, h_{dec}) = v * tanh(W^e * h_{enc} + W^d * h_{dec} + W^c * conv + b) \tag{6.10}$$

where v and b are learnable weights, together with the W matrices and $conv$ is the location-aware element, which is a convolution defined as:

$$conv = F * \alpha_{ts} \tag{6.11}$$

where, F is a learnable matrix.

After the scores and the alignment vector are computed, the context vector c_t is then calculated as:

$$c_t = \sum_s \alpha_{ts} h_{enc} \tag{6.12}$$

To get the attention vector att_t, we concatenate the context vector c_t and the decoder hidden state h_{dec} and pass them through a fully connected layer:

$$\text{att}_t = W^c[c_t; h_{dec}] \tag{6.13}$$

6.3.2 Bahdanau attention

To apply the Bahdanau attention, we first need to calculate the alignment score α_{ts}, just like in the Luong attention:

$$\alpha_{ts} = softmax(score(h_{enc}, h_{dec})) \tag{6.14}$$

The scoring function is defined as:

$$score(h_{enc}, h_{dec}) = v * tanh(W_t * h_{enc} + W_s * h_{dec}) \tag{6.15}$$

where, the vector v and the matrices W are learnable parameters.

With this scoring function, the encoder and the decoder hidden states get passed through separate fully connected layers and then added together, after which a tanh nonlinearity is applied. In the end, the score is multiplied by weight vector v.

After the alignment is computed, the context vector c_t is calculated as:

$$c_t = \sum_s \alpha_{ts} h_{enc} \tag{6.16}$$

and is concatenated with the embedding as follows:

$$k = [c_t; x_t] \tag{6.17}$$

where x_t is the output of the embedding layer. In the end, the concatenated result and the decoder hidden states are passed through a recurrent neural network, followed by a fully connected layer:

$$h_{dec} = W^c * f(W^{hh} k_{t-1} + W^{hx} h_{dec}) \tag{6.18}$$

6.4 Multi-task learning

The multi-task approach uses attention-based encoder-decoder architecture, like the ASR with augmented labels. In comparison to the previous approach, in addition to the ASR decoder, this method incorporates another decoder branch for doing named entity recognition. The architecture is depicted in figure 16.

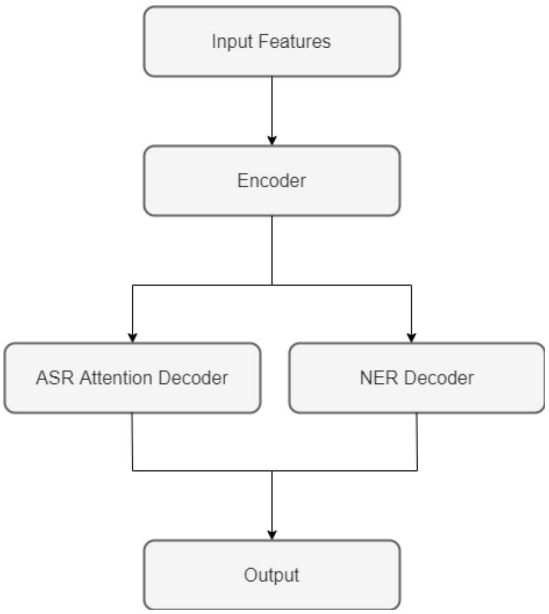

Figure 16: Model architecture for multi task learning.

As we can see from the figure, both decoders share the same encoder. The encoder and the ASR attention decoder are the same as in the ASR with augmented labels model. The NER decoder consists of a bidirectional LSTM with a CRF layer on top.

CRF is a discriminative model suited for sequential tasks where the neighboring states can affect the current prediction. It is a graphical probabilistic model that uses state transition matrix as parameters. The output of the model for an input sentence $[X]_n$ is the matrix of scores $f_\theta([X]_n)$, where θ represents the learnable parameters. Each element of the matrix $[f_\theta]_{i,j}$ is the score for the i-th tag at the j-th word. Besides the matrix of scores, there is also a transition matrix $[A]_{i,j}$, storing the transition scores from i-th to the j-th state. By introducing the transition matrix, additional learnable parameters need to be added. We will denote the whole parameter set as θ'. The score for each sentence $[X]_n$ with the tag sequence $[i]_n$ can be calculated by summing the matrix of scores and the transition scores:

$$s([X]_n, [i]_n, \theta') = \sum_{t=1}^{T} [f_\theta]_{[i]_t,t} + [A]_{[i]_{t-1},[i]_t} \tag{6.19}$$

Since this is a multi-task approach, we have two losses: ASR loss and NER loss. To get the final loss, we do a weighted sum between the two losses as follows:

$$L = \lambda L_{asr} + (1 - \lambda) L_{ner} \tag{6.20}$$

where L_{asr} is the loss from the ASR decoder, L_{ner} is the loss from the NER decoder, and λ is a weighting factor that determines the contribution of both loss functions.

6.5 Decoding

As discussed earlier in this chapter, the encoder-decoder architecture produces a probability distribution of the character set, for each timestep. Having a probability distribution is not sufficient because we need to pick one character that is most probable for each timestep. To solve that, we are going to discuss two decoding approaches for doing that: greedy decoding and beam search decoding.

6.5.1 Greedy decoding

One of the easiest and most straightforward decoding strategies is greedy decoding. This strategy takes the output of the decoder, which is a probability distribution over all the characters, and picks the most probable one at each timestep. The advantage of this approach is that it is easy to implement and it works fast. Even though this approach can produce good results, the quality of the output is still worse than using other decoding strategies. The greedy decoder can pick the most probable character, which might be incorrect and that will influence the rest of the decoding. This issue is solved with the beam search decoder.

6.5.2 Beam search decoding

Beam search decoding is an improved decoding strategy, in comparison to the greedy search. It works in a way that instead of considering only the best character at each timestep, we consider n different ones at the same time. In the end, we can select n different hypotheses. The score for each hypothesis is calculated as the sum of the log-probabilities of the selected characters. This approach solves the problem that greedy decoding has, which is selecting a wrong character and influencing the rest of the decoding from that character. The beam search decoding algorithm usually produces better results in comparison to greedy search. The downside of this decoding strategy is that the computational time can get high, especially when the number of considered characters at each timestep is high. In our experiments, we are going to use this decoding strategy because it is superior to greedy decoding.

7 Experiments

This section covers the data that was used in the experiments. Furthermore, it explains the setup that was used for the experiments, including the choice of audio features, architecture type, and various hyper-parameters. The last part of the section covers the most important part of the thesis, which is the results. We will compare both the augmented labels and multi-task approaches with the standard pipeline and observe the advantages and drawbacks of using an end-to-end model. In the multi-task approach, we will see how the NER branch performs when trained individually, in comparison to training it jointly with the ASR branch. Lastly, we will compare the two proposed approaches for E2E named entity extraction from speech and see which one performs better.

7.1 Data

As training data, we used Finnish parliament sessions [74]. The dataset consists of approximately 1500 hours of recordings. To investigate the impact of the amount of data on the results, we also used a subset of the dataset, containing approximately 600 hours.

The whole data consists of approximately 7.3 million tokens, from which 337423 are unique. The dataset is annotated with a baseline named entity recognition system, consisting of three named entity tags: person (PER), location (LOC), and organization (ORG). The number of person tags is 44984, the number of location tags is 73860 and the number of organization tags is 65463.

The subset data contains approximately 3.6 million tokens, where the number of unique ones is 227763. The number of person tags is 23018, the number of location tags is 37835 and the number of organization tags is 34288.

The data distribution for both the whole and the subset datasets is shown in table 1

Table 1: Data distribution for the whole and the subset datasets.

Parameters	Whole data	Subset data
Audio length	1500 h	600 h
Total tokens	7.3 M	3.6 M
Unique tokens	337423	227763
PER tags	44984	23018
LOC tags	73860	37835
ORG tags	65463	34288

To train the baseline NER system, we combined two datasets: Digitoday and Turku NER.

The Digitoday dataset [1] was collected and provided by [75]. It consists of online Finnish technological news articles. There are 953 articles and 193,742 word tokens in the dataset. Since the articles are from one domain, the authors also provided a

[1]The dataset is publicly available at: https://github.com/mpsilfve/finer-data

Wikipedia test for evaluating the system on out-of-domain data. Both datasets are annotated using the BIO annotation scheme [76]. The Wikipedia test set consists of 83 articles and 49,752 word tokens. The class distribution in the Digitoday dataset is shown in table 2.

Table 2: Class distribution in Digitoday and Wikipedia.

Class	Count Digitoday	Count Wikipedia
ORG (organization)	15445 (36.74%)	1821 (18.03%)
LOC (location)	4159 (9.89%)	1427 (14.13%)
PER (person)	6517 (15.50%)	2492 (24.67%)
DATE (date)	3685 (8.76%)	1862 (18.43%)
PRO (product)	11655 (27.73%)	2135 (21.14%)
EVENT (event)	569 (1.35%)	362 (3.58%)
TOTAL	42030	10099

The Turku NER corpus [2] was collected and provided by [77]. This corpus was also annotated using the BIO scheme, just like the Digitoday. The class distribution for the Turku NER dataset is presented in table 3.

Table 3: Class distribution in Turku NER.

Class	Count Turku NER
ORG (organization)	2601 (22.72%)
LOC (location)	3269 (28.55%)
PER (person)	3085 (26.94%)
DATE (date)	1332 (11.63%)
PRO (product)	980 (8.56%)
EVENT (event)	181 (1.58%)
TOTAL	11448

7.2 Experimental setup

As discussed in the previous section, we used the Digitoday and Turku NER datasets to train the baseline NER system. Since our goal is to annotate the parliament transcripts, which are in lowercase and without punctuation, we pre-processed the training data and converted it to lowercase.

To make a distinction between the first and the last word in a sentence and the rest of the words, we added "<*start*>" and "<*end*>" tokens to each sentence. For the morph-based sub-word modeling, we added boundary markers to enforce restrictions on the generated output. Different ways of adding markers enforce different restrictions. Some common types of markers are: "<*w*>", "<*m+*>", "<*+m*>", "<*+m+*>". In our experiments, we used the "<*+m+*>" style marker since it is shown to give the best results for Finnish language modeling in ASR [78]. For example, the word "mobiilikäyttöjärjestelmä" would be segmented as "mobiili+ +käyttö+ +järjestelmä".

[2]The dataset is publicly available at: https://github.com/TurkuNLP/turku-ner-corpus

The architecture has 2 BLSTM layers and 4 highway layers. The embedding dimensions of words, chars, and morphs are 300, 100, 100 respectively for the BLSTM networks. The hidden sizes are 300, 75, 75 for words, chars, and morphs. A dropout of 0.5 is added to the final BLSTM outputs and 0.2 for each layer except for the last. After the highway layer, we added a dropout probability of 0.7. For training, the model we used a batch size of 128 and RAdam optimizer [79] with a learning rate of 0.001. All of the hyperparameters were chosen based on internal experiments that we did on the development set. The model parameters are presented in table 4.

Table 4: Model parameters for the baseline NER system.

Parameters	Value
BLSTM layers	2
Highway layers	4
Word embedding dim	300
Character embedding dim	100
Morph embedding dim	100
Word hidden size	300
Character hidden size	75
Morph hidden size	75
Optimizer	RAdam
Learning rate	0.001

As discussed in section 6, both ASR with augmented labels and multi-task learning approaches use attention-based encoder-decoder architecture. As input features, we used logarithmic filter banks with 40 filters. Speech features consist of a large number of timesteps, so processing them using a standard BLSTM network is computationally expensive. To deal with that we use a pyramidal BLSTM network. The pyramidal structure reduces the computational time by concatenating every two consecutive timesteps in each layer except the first. Our encoder consists of 5 BLSTM layers and in each layer, the time resolution is reduced by half. The hidden size of the encoder is 300. A dropout with a probability of 0.1 is applied after the last BLSTM layer. As an optimizer, we used Adam [80] with a learning rate of 0.0005.

In the ASR with augmented labels approach, the decoder consists of a character embedding layer with a dimension of 150, a single layer LSTM network with a hidden size of 300, and a hybrid + attention-aware attention size of 300. The number of filters in the convolutional layer is 100. A dropout of 0.1 is applied after the attention mechanism. The decoder uses Adam optimizer with 0.0005 learning rate.

In the multi-task approach, we have two decoders, one for ASR and one for NER. The ASR decoder is identical to the decoder in the augmented labels approach. The NER decoder uses pre-trained 300 dimensional fastText word embeddings as input to the one-layer BLSTM network with hidden size of 300. The output of the BLSTM network is followed by a fully-connected layer with a hidden size of 300 and a dropout probability of 0.1. In the end, the output is passed through a CRF layer which produces tag probabilities. The NER decoder uses Adam optimizer with a learning rate of 0.0005. After multiple experiments, we decided to use λ weighting factor of 0.8 for combining the separate losses as in equation 6.20.

In both of the approaches, we used negative log-likelihood loss. The parameters are given in table 5 and table 6.

As a baseline speech recognition system, we are using the augmented labels model but without augmenting the transcripts with named entities.

Table 5: Model parameters for augmented labels approach.

Parameters	Encoder	Decoder
Layers	5	1
Hidden size	300	300
Learning rate	0.0005	0.0005
Char embedding size	150	
Attention type	hybrid + location-aware	
Attention size	300	
Location-aware convolution size	100	
Dropout	0.1	
Optimizer	Adam	
Batch size	10	

Table 6: Model parameters for multi-task approach.

Parameters	Encoder	ASR Decoder	NER Decoder
Layers	5	1	1
Hidden size	300	300	300
Learning rate	0.0005	0.0005	0.0005
Word embedding dim	300		
Char embedding dim	150		
Attention type	hybrid + location-aware		
Attention size	300		
Location-aware convolution size	100		
Dropout	0.1		
Optimizer	Adam		
Batch size	16		
λ Weighting factor	0.8		

7.3 Results

This subsection covers the most interesting part of the thesis, that is, the results. We will start by presenting the results for the baseline NER architecture.

We evaluated the baseline NER system on the Digitoday and Wikipedia test sets, converting them in lowercase and removing the punctuation. The results are presented in table 7.

Next, we can observe how both E2E approaches performed in terms of WER on the subset parliament dataset. The results are shown in table 8.

Similarly, in table 9, we can observe how both approaches performed on the whole parliament dataset, but this time, in comparison to the baseline ASR system. The speech recognition baseline system is the same as the augmented labels architecture but using the original transcripts. It is a standard attention-based encoder-decoder architecture.

Table 7: F1 score for the Digitoday and Wikipedia test sets, evaluated using the baseline NER system trained on lowercase data.

	Digitoday			Wikipedia		
Entity tag	Precision	Recall	F1	Precision	Recall	F1
ORG	68.45	83.89	75.39	53.44	55.10	52.24
PER	73.81	82.66	77.98	82.47	70.40	75.96
LOC	86.53	70.38	77.62	72.31	64.62	68.25
DATE	88.22	99.46	93.50	86.70	96.86	91.50
PRO	69.74	53.62	60.63	67.10	46.33	54.81
EVENT	93.33	48.28	63.64	34.64	14.72	20.66
Micro average	72.22	73.05	72.63	72.83	63.74	67.98

Table 8: WER on the parliament subset data.

Model	WER
Augmented labels	39.49
Multi-task	39.66

Table 9: WER on the parliament whole data.

Model	WER
Baseline ASR	34.52
Augmented labels	35.16
Multi-task	35.76

Next, we wanted to see how well the multi-task model performs in terms of precision, recall, and F1 score, across all the named entities. The results are presented in table 10.

Table 10: Precision, recall and F1 score for the subset and whole parliament data, using the multi-task approach.

	parliament subset			parliament whole		
Entity tag	Precision	Recall	F1	Precision	Recall	F1
PER	86.16	80.90	83.45	89.05	81.96	85.36
ORG	84.44	76.22	80.12	85.40	78.92	81.98
LOC	95.53	94.50	95.01	95.12	96.57	95.84
Micro average	93.25	91.05	92.13	93.69	92.87	93.27

To see if the ASR branch in the multi-task learning approach had any impact on learning the named entities, we disabled it and trained only the NER branch. The results are shown in table 11.

Table 11: F1 score for the whole and subset parliament dataset using the multi-task approach where the ASR branch is disabled.

	parliament subset			parliament whole		
Entity tag	Precision	Recall	F1	Precision	Recall	F1
PER	89.80	84.08	85.85	91.25	83.02	86.94
ORG	88.63	76.78	82.28	90.91	79.28	84.69
LOC	96.79	95.2	96.00	96.32	96.75	96.54
Micro average	95.08	92.40	93.72	95.13	93.27	94.19

So far, we were using the original transcripts (the gold standard) to do named entity recognition. In the next experiment, we did NER on the transcripts generated by the multi-task model. First, we generated the transcripts with the ASR branch and then got the named entities using the baseline NER system. Next, we generated named entities using the NER branch and compared it against the ones generated by the baseline model. The results are shown in table 12.

Table 12: F1 score for the whole parliament dataset using the multi-task approach, where NER is done on the transcripts generated by the ASR branch.

Entity tag	Precision	Recall	F1
PER	83.82	64.43	72.86
ORG	82.21	76.03	79.00
LOC	95.19	93.24	94.21
Micro average	92.70	85.66	89.04

We did a similar experiment for the augmented labels approach. We first generated the transcripts and then used the baseline NER system on those transcripts. In the end, we compared the named entities of the baseline system and the E2E augmented labels approach. The results are shown in table 13.

Table 13: F1 score for the whole parliament dataset using the augmented labels approach, where NER is done on the transcripts generated by the E2E system.

Entity tag	Precision	Recall	F1
PER	83.51	55.34	66.57
ORG	90.18	62.22	73.64
LOC	95.07	89.78	92.35
Micro average	92.65	80.35	86.06

7.4 Analysis of the results

From the baseline NER system results presented in table 7, we can see that the system performs better on the Digitoday dataset because that data is more in-domain, in comparison to the Wikipedia. The F1 scores for both datasets are not very high because the system was trained and evaluated on lowercase data, which causes difficulties for the system, as discussed in the previous sections. Nevertheless, this system serves as a good baseline.

In table 8, we can see the WER for both approaches, when evaluated on the subset parliament data. From the table, we can see that the augmented labels approach achieves 39.49% WER, which is a small improvement over the multi-task approach, which got 39.66% WER.

Similar results can be seen in table 9, where we evaluated the WER for both approaches, but this time, in comparison to the baseline ASR. Here, the models are evaluated on the whole parliament data, which gives an improvement. From the table, we can see that the augmented labels approach achieves WER of 35.16%, which is an 0.6% improvement over the multi-task approach, which achieves 35.76% WER. When we compare the results to the baseline ASR, we can see that both approaches perform similarly in terms of WER but still fall slightly behind the 34.52% WER that the baseline model achieves.

From table 10, we can see that using the whole dataset improved the NER results for the multi-task approach, compared to using the subset data. Using the whole data, we got an absolute improvement of 1.14%. From the table, we can also see that the multi-task approach does a really good job of learning the named entities, achieving a high F1 score of 93.27% on the whole parliament data. In both the subset and the whole dataset, we can see that the model mostly struggles with the ORG entity, especially in terms of recall.

When training the multi-task approach only with the NER branch and disabling the ASR, we can see that the model is still able to learn the named entities. The results can be seen in table 11. If we compare these results with the ones in table 10, we can see that the system is able to learn better the named entities by relying only on the NER branch, which gives an improvement of 0.92%. This also shows that the named entities can be learned directly from the acoustic features. Even though by using only the NER branch, the system achieves slightly better results, this approach is not very useful because we don't have the transcripts. Without the transcripts, it is hard to interpret the text just from the named entities.

In table 12, we can see that even when we did a NER on the transcripts generated by the ASR branch, instead of using the original ones, the system still achieves good results. We can notice that the system does a pretty good job with the LOC entity and mostly struggles with detecting the PER entity, thus the lower recall score.

In table 13, we did similar experiment but with the augmented labels approach. Similarly, like in the multi-task approach, we can observe that the system struggles to recognize the PER entity and achieves lower recall. Nevertheless, the augmented labels approach does a pretty good job of doing NER.

If we compare the overall F1 score in tables 12 and 13, we can see that the multi-task approach achieves higher F1 score in comparison to the augmented labels, with an absolute improvement of 2.98%.

A list of sample transcriptions and their named entity annotations, done using the multi-task approach is presented in table 14. From the table, we can see that the system does a pretty good job of transcribing speech, as well as doing named entity recognition. It is also important to mention that the named entity recognition in this case is done on the original transcripts and not on the predictions.

Similarly, in table 15, we see a list of samples and their named entities, done

using the augmented labels approach. From the sample sentences, we can see that the augmented labels approach also is able to do a good job at transcribing speech and detecting named entities, just like the multi-task approach.

Table 14: WER and F1 scores for sample sentences using the multi-task approach.

True sentence	Predicted šentence	True NER	Predicted NER	WER %	F1 %
eukin on tehnyt sinänsä hyviä päätöksiä näistä ikään kuin taakanjaoista	eukki on tehnyt sinänsä hyviä päätöksiä näistä ikään kuin taakajaojaosta	'PER', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O'	'PER', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O'	20.00	100
on täysin selvää että eduskunta ja hallitus kokonaisuudessaan ajavat suomen etua eivätkä suostu	on täysin selvää että eduskunta ja hallitus kokonaisuudessaan ajaa suomen etua eikä suostu	'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'LOC', 'O', 'O', 'O'	'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'LOC', 'O', 'O', 'O'	15.38	100
mutta samaan aikaan koko eurooppa mietti kuinka pörssipelureita ja joukkovelkakirjoihin sijoittaneita pelastetaan kun pitäisi miettiä kuinka luodaan uutta kasvua ja uutta	mutta samaan aikaan koko eurooppa mietti kuinka pörssipelureita ja joukkovelkakirjoihin sijoittaneita pelastetaan kun pitäisi miettiä kuinka luodaan uutta kasvua ja uhta	'O', 'O', 'O', 'O', 'LOC', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O'	'O', 'O', 'O', 'O', 'LOC', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O'	4.76	100
valiokunta katsoo että suomen keskeinen tavoite on kansallisten ulkopoliittisten intressien toteutuminen joko kahdenvälisesti tai eu tasolla	valiokunta katsoo että suomen keskeinen tavoite on kansallisten ulkopoliittisesti ja intressien toteutuminen joko kahdenvälisesti tai eu tasolla	'O', 'O', 'O', 'LOC', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'ORG', 'O'	'O', 'O', 'O', 'LOC', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'ORG', 'O'	12.50	100
<UNK> ei käy ilmi onko eussa arvioitu keski afrikan tasavallan aikaisempaan kehitykseen ja vakauteen vaikuttaneita tekijöitä	teosta ei käy ilmi onko eussa arvioitu keski afrikan tasavallan aikaisempaan kehitykseen ja vakauteen vaikuttaneita tekijöitä	'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'LOC', 'LOC', 'O', 'O', 'O', 'O', 'O', 'O'	'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'LOC', 'LOC', 'O', 'O', 'O', 'O', 'O', 'O'	6.25	100

Table 15: WER for sample sentences using the augmented labels approach. Here, the named entity tags are included in the WER calculation.

True sentence	Predicted sentence	WER %
matti PER vanhasen PER kakkoshallitus O on O sosiaalisesti O oikeudenmukainen O ja O se O myös O näkyy O tehdyissä O päätöksissä O	matti PER vanhasen PER kakkoshallitus O on O sosiaalisesti O oikeudenmukainen O ja O se O myös O näkyy O tehdyissä O päätöksessä O	4.16
keskustan ORG eduskuntaryhmä ORG antaa O hallitukselle O ja O erityisesti O elinkeinoministeri O pekkariselle O tunnustuksen O energiapaketin O määrätietoisesta O perusteellisesta O ja O ripeästä O valmistelusta O	keskustan ORG eduskuntaryhmä ORG antaa O hallitukselle O ja O erityisesti O elinkeinoministeri O pekkariselle O tunnustuksen O energiapaketin O määrätietoisesta O perusteellisesta O ja O ripeästä O valmistelusta O	0.00
on O hyvä O että O monista O muista O maista O poiketen O suomen LOC rikkaista O luonnonvaroista O löytyvät O lääkkeet O vastata O	on O hyvä O että O monista O muista O maista O poiketen O suomen LOC rikkaissa O luonnonvaroissa O löytyvät O lääkkeet O vastata O	7.69
keskustan ORG eduskuntaryhmä ORG pitää O hyvänä O että O vanhasen PER kakkoshallitus O jatkaa O vanhasen O ykköshallituksen O keskustalaisella O kannustavalla O	keskustan ORG eduskuntaryhmä ORG pitää O hyvänä O että O vanhasen PER kakkoshallitus O jatkaa O vanhasen PER yksi O hallituksen O keskustalaisella O kannustavalla O	12.50
puolitoista O vuotta O sitten O suomen LOC pienenä O maana O piti O kantaa O vastuuta O antaa O näille O pienille O maille O kreikalle LOC irlannille LOC ja O <UNK> O	puolitoista O vuotta O sitten O suomen LOC pienenä O maana O piti O kantaa O vastuuta O antaa O näille O pienille O maille O kreikalle LOC irlannille LOC ja O <UNK> O	0.00
keskustalainen O suomi LOC ei O ole O tasaverojen O suomi LOC vaan O maa O jossa O jokaisen O on O osallistuttava O yhteiskunnan O rakentamiseen O kykyjensä O	keskustalainen O suomi LOC ei O ole O tasaverojen O suomi LOC vaan O maa O jossa O jokaisen O on O osallistettava O yhteiskunnan O rakentamiseen O kykyjensä O	3.33
päinvastoin O kuin O oppositiolla O suomen ORG hallituksella ORG ei O ole O käytössään O politikoinnin O mahdollisuutta O	päinvastoin O kun O oppositiolla O suomen ORG hallituksella ORG ei O ole O käytössään O politikoinnin O mahdollisuutta O	5.00

8 Conclusion

This chapter covers what are the main takeaways from this thesis. It shows what we learned from experimenting with doing named entity recognition for spoken Finnish in an end-to-end manner. Furthermore, it covers the future directions in which this research can be extended.

8.1 Conclusion

Doing named entity recognition on spoken data is a challenging task that requires careful design choices. In the standard pipeline approach, the spoken data is first transcribed using an automatic speech recognition system and then the named entity recognition system annotates the transcripts. The speech recognition output is usually without capitalization and punctuation, which causes the named entity recognition system to miss or misclassify the entities. Usually, the speech recognition systems are not perfect and they make mistakes, which are propagated to the named entity recognition system. Having two separate systems also means that the ASR system is not optimized for the NER task and vice-versa.

To overcome the above issues that arise when doing named entity recognition on spoken data, we developed two approaches for extracting named entities from speech in an end-to-end manner. These two approaches are the main focus of this thesis.

The augmented labels approach is attention-based encoder-decoder architecture, that instead of using the normal transcripts during training, it uses transcripts augmented with named entities. This way, the model can learn both tasks simultaneously.

The second approach is the multi-task learning approach, which is also attention-based encoder-decoder architecture. The difference between this and the augmented labels approach is that in this approach we have two separate decoder branches. One branch for doing NER and one for ASR. This allows both decoder branches to share the encoder by doing hard parameter sharing, which reduces the number of parameters.

We showed that both approaches do a good job of transcribing speech and doing named entity recognition. In the thorough experiments that we did, we showed that the augmented labels approach is slightly better at transcribing the speech, whereas the multi-task approach achieves better results on annotating the transcripts with named entities. In comparison to the baseline ASR model, both proposed approaches perform slightly worse in terms of WER. This means that the performance of the ASR does not suffer too much from learning to annotate the named entities.

To explore the impact of the ASR branch in the multi-task learning approach, we disabled it during training. From those experiments, we showed that the ASR branch does not have that much impact on learning the named entities but is still important to have if we want to analyze what has been spoken. We also showed that the named entities can be learned directly from the acoustic features and even achieve slightly better results than training the system together with the ASR branch.

Another thing that we can conclude is that adding more data helps in achieving

better results. In all the experiments that we did, the models trained on the whole parliament data outperform the ones trained on the subset of the data.

From the experiments in this thesis, we can conclude that doing named entity recognition from speech in an end-to-end manner is a good approach and performs very good in comparison to the standard pipeline approach. They keep in pair with the baseline ASR system, while also achieving high F1 score on the NER task.

The end-to-end approaches do the transcription and the named entity annotation, using just one system, optimized to do both tasks, instead of using two separate systems, like in the pipeline approach. This reduces the total number of parameters that need to be learned, which can speed up the computation.

In the future, we plan to investigate the multi-task learning approach even more. We plan to adaptively adjust the weight of both losses so that we can give more advantage to the ASR loss later in the training. We also plan to experiment more with scheduled sampling during training.

The experiments that we did in this thesis are on in-domain parliament data. In the future, we plan to investigate how both approaches perform on out-of-domain data. Furthermore, we plan to test the models on manually annotated data and see how well they perform against the pipeline approach.

References

[1] M. Hassel, "Exploitation of named entities in automatic text summarization for swedish," in *NODALIDA'03–14th Nordic Conferenceon Computational Linguistics, Reykjavik, Iceland, May 30–31 2003*, p. 9, 2003.

[2] B. Babych and A. Hartley, "Improving machine translation quality with automatic named entity recognition," in *Proceedings of the 7th International EAMT workshop on MT and other language technology tools, Improving MT through other language technology tools, Resource and tools for building MT at EACL 2003*, 2003.

[3] J. Jiang, "Information extraction from text," in *Mining text data*, pp. 11–41, Springer, 2012.

[4] R. Leaman, C.-H. Wei, and Z. Lu, "tmchem: a high performance approach for chemical named entity recognition and normalization," *Journal of cheminformatics*, vol. 7, no. S1, p. S3, 2015.

[5] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, pp. 1310–1318, 2013.

[6] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[7] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] M. Gerosa, D. Giuliani, and F. Brugnara, "Acoustic variability and automatic recognition of children's speech," *Speech Communication*, vol. 49, no. 10-11, pp. 847–860, 2007.

[10] T. L. Perry, R. N. Ohde, and D. H. Ashmead, "The acoustic bases for gender identification from children's voices," *The Journal of the Acoustical Society of America*, vol. 109, no. 6, pp. 2988–2998, 2001.

[11] V. Zue, S. Seneff, J. R. Glass, J. Polifroni, C. Pao, T. J. Hazen, and L. Hetherington, "Juplter: a telephone-based conversational interface for weather information," *IEEE Transactions on speech and audio processing*, vol. 8, no. 1, pp. 85–96, 2000.

[12] M. Bates, R. Bobrow, P. Fung, R. Ingria, F. Kubala, J. Makhoul, L. Nguyen, R. Schwartz, and D. Stallard, "The bbn/harc spoken language understanding system," in *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp. 111–114, IEEE, 1993.

[13] 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*, pp. 369–376, 2006.

[14] E. Pépiot, "Voice, speech and gender:. male-female acoustic differences and cross-language variation in english and french speakers," *Corela. Cognition, représentation, langage*, no. HS-16, 2015.

[15] C. Andrade and O. V. Martins, "Speech fluency variation in elderly.," *Pro-fono: revista de atualizacao cientifica*, vol. 22, no. 1, pp. 13–18, 2010.

[16] M. Gerosa, S. Lee, D. Giuliani, and S. Narayanan, "Analyzing children's speech: An acoustic study of consonants and consonant-vowel transition," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 1, pp. I–I, IEEE, 2006.

[17] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, pp. 707–710, 1966.

[18] K. H. Davis, R. Biddulph, and S. Balashek, "Automatic recognition of spoken digits," *The Journal of the Acoustical Society of America*, vol. 24, no. 6, pp. 637–642, 1952.

[19] J. G. Wilpon, L. R. Rabiner, C.-H. Lee, and E. Goldman, "Automatic recognition of keywords in unconstrained speech using hidden markov models," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, no. 11, pp. 1870–1878, 1990.

[20] A. Varga and R. Moore, "Hidden markov model decomposition of speech and noise," in *International Conference on Acoustics, Speech, and Signal Processing*, pp. 845–848, IEEE, 1990.

[21] R. Moore, "Signal decomposition using markov modelling techniques," *MEMORANDUM-ROYAL SIGNALS AND RADAR ESTABLISHMENT RSRE MEMO*, 1986.

[22] I. M. El-Emary, M. Fezari, and H. Attoui, "Hidden markov model/gaussian mixture models (hmm/gmm) based voice command system: A way to improve the control of remotely operated robot arm tr45," *Scientific Research and Essays*, vol. 6, no. 2, pp. 341–350, 2011.

[23] P. Bansal, A. Kant, S. Kumar, A. Sharda, and S. Gupta, "Improved hybrid model of hmm/gmm for speech recognition," *Intelligent Information and Engineering Systems*, 2008.

[24] L. Deng, P. Kenny, M. Lennig, V. Gupta, F. Seitz, and P. Mermelstein, "Phonemic hidden markov models with continuous mixture output densities for large vocabulary word recognition," *IEEE Transactions on Signal Processing*, vol. 39, no. 7, pp. 1677–1681, 1991.

[25] M. O. Khelifa, Y. M. Elhadj, Y. Abdellah, and M. Belkasmi, "Constructing accurate and robust hmm/gmm models for an arabic speech recognition system," *International Journal of Speech Technology*, vol. 20, no. 4, pp. 937–949, 2017.

[26] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on audio, speech, and language processing*, vol. 20, no. 1, pp. 30–42, 2011.

[27] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[28] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649, IEEE, 2013.

[29] J. S. Garofolo, "Timit acoustic phonetic continuous speech corpus," *Linguistic Data Consortium, 1993*, 1993.

[30] K. Rao, H. Sak, and R. Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 193–199, IEEE, 2017.

[31] 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)*, pp. 4960–4964, IEEE, 2016.

[32] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[33] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[34] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid ctc/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

[35] S. Petridis, T. Stafylakis, P. Ma, G. Tzimiropoulos, and M. Pantic, "Audio-visual speech recognition with a hybrid ctc/attention architecture," in *2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 513–520, IEEE, 2018.

[36] R. Grishman and B. M. Sundheim, "Message understanding conference-6: A brief history," in *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, 1996.

[37] D. Farmakiotou, V. Karkaletsis, J. Koutsias, G. Sigletos, C. D. Spyropoulos, and P. Stamatopoulos, "Rule-based named entity recognition for greek financial texts," in *Proceedings of the Workshop on Computational lexicography and Multimedia Dictionaries (COMLEX 2000)*, pp. 75–78, 2000.

[38] K. Torisawa *et al.*, "Inducing gazetteers for named entity recognition by large-scale clustering of dependency relations," in *proceedings of ACL-08: HLT*, pp. 407–415, 2008.

[39] F. Jahangir, W. Anwar, U. I. Bajwa, and X. Wang, "N-gram and gazetteer list based named entity recognition for urdu: A scarce resourced language," in *Proceedings of the 10th Workshop on Asian Language Resources*, pp. 95–104, 2012.

[40] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the 18th International Conference on Machine Learning 2001 (ICML 2001)*, pp. 282–289, 2001.

[41] A. Borthwick and R. Grishman, *A maximum entropy approach to named entity recognition*. PhD thesis, Citeseer, 1999.

[42] H. L. Chieu and H. T. Ng, "Named entity recognition: a maximum entropy approach using global information," in *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pp. 1–7, Association for Computational Linguistics, 2002.

[43] A. McCallum and W. Li, "Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons," in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pp. 188–191, Association for Computational Linguistics, 2003.

[44] B. Settles, "Biomedical named entity recognition using conditional random fields and rich feature sets," in *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*, pp. 107–110, 2004.

[45] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.

[46] Z. Liu, M. Yang, X. Wang, Q. Chen, B. Tang, Z. Wang, and H. Xu, "Entity recognition from clinical texts via recurrent neural network," *BMC medical informatics and decision making*, vol. 17, no. 2, p. 67, 2017.

[47] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[48] E. F. Sang and F. De Meulder, "Introduction to the conll-2003 shared task: Language-independent named entity recognition," *arXiv preprint cs/0306050*, 2003.

[49] T. Hirsimaki, J. Pylkkonen, and M. Kurimo, "Importance of high-order n-gram models in morph-based speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 4, pp. 724–732, 2009.

[50] O. Kuru, O. A. Can, and D. Yuret, "Charner: Character-level named entity recognition," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 911–921, 2016.

[51] M. Gridach, "Character-level neural network for biomedical named entity recognition," *Journal of biomedical informatics*, vol. 70, pp. 85–91, 2017.

[52] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional lstm-cnns-crf," *arXiv preprint arXiv:1603.01354*, 2016.

[53] C. N. d. Santos and V. Guimaraes, "Boosting named entity recognition with neural character embeddings," *arXiv preprint arXiv:1505.05008*, 2015.

[54] M. Creutz, T. Hirsimäki, M. Kurimo, A. Puurula, J. Pylkkönen, V. Siivola, M. Varjokallio, E. Arisoy, M. Saraçlar, and A. Stolcke, "Morph-based speech recognition and modeling of out-of-vocabulary words across languages," *ACM Transactions on Speech and Language Processing (TSLP)*, vol. 5, no. 1, pp. 1–29, 2007.

[55] V. Siivola, T. Hirsimaki, M. Creutz, and M. Kurimo, "Unlimited vocabulary speech recognition based on morphs discovered in an unsupervised manner," in *Eighth European Conference on Speech Communication and Technology*, 2003.

[56] R. Yeniterzi, "Exploiting morphology in turkish named entity recognition system," in *Proceedings of the ACL 2011 Student Session*, pp. 105–110, Association for Computational Linguistics, 2011.

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

[58] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

[59] X. Feng, X. Feng, B. Qin, Z. Feng, and T. Liu, "Improving low resource named entity recognition using cross-lingual knowledge transfer.," in *IJCAI*, pp. 4071–4077, 2018.

[60] J. Xie, Z. Yang, G. Neubig, N. A. Smith, and J. Carbonell, "Neural cross-lingual named entity recognition with minimal resources," *arXiv preprint arXiv:1808.09861*, 2018.

[61] A. Gravano, M. Jansche, and M. Bacchiani, "Restoring punctuation and capitalization in transcribed speech," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4741–4744, IEEE, 2009.

[62] L. Zhai, P. Fung, R. Schwartz, M. Carpuat, and D. Wu, "Using n-best lists for named entity recognition from chinese speech," in *Proceedings of HLT-NAACL 2004: Short Papers*, pp. 37–40, 2004.

[63] S. Ghannay, A. Caubrière, Y. Estève, N. Camelin, E. Simonnet, A. Laurent, and E. Morin, "End-to-end named entity and semantic concept extraction from speech," in *2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 692–699, IEEE, 2018.

[64] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in *1995 International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. 181–184, IEEE, 1995.

[65] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[66] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning word vectors for 157 languages," in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[67] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[68] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," *arXiv preprint arXiv:1901.02860*, 2019.

[69] S. Toshniwal, A. Kannan, C.-C. Chiu, Y. Wu, T. N. Sainath, and K. Livescu, "A comparison of techniques for language model integration in encoder-decoder speech recognition," in *2018 IEEE spoken language technology workshop (SLT)*, pp. 369–375, IEEE, 2018.

[70] C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.-C. Lin, F. Bougares, H. Schwenk, and Y. Bengio, "On using monolingual corpora in neural machine translation," *arXiv preprint arXiv:1503.03535*, 2015.

[71] A. Sriram, H. Jun, S. Satheesh, and A. Coates, "Cold fusion: Training seq2seq models together with language models," *arXiv preprint arXiv:1708.06426*, 2017.

[72] D. Porjazovski, J. Leinonen, and M. Kurimo, "Named entity recognition for spoken finnish," in *2nd International Workshop on AI for Smart TV Content Production: Affiliation; Access and Delivery*, (New York, NY, USA), ACM, 2020.

[73] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[74] A. Mansikkaniemi, P. Smit, M. Kurimo, *et al.*, "Automatic construction of the finnish parliament speech corpus.," in *INTERSPEECH*, vol. 8, pp. 3762–3766, 2017.

[75] T. Ruokolainen, P. Kauppinen, M. Silfverberg, and K. Lindén, "A finnish news corpus for named entity recognition," *Language Resources and Evaluation*, pp. 1–26, 2019.

[76] L. A. Ramshaw and M. P. Marcus, "Text chunking using transformation-based learning," in *Natural language processing using very large corpora*, pp. 157–176, Springer, 1999.

[77] J. Luoma, M. Oinonen, M. Pyykönen, V. Laippala, and S. Pyysalo, "A broad-coverage corpus for finnish named entity recognition," in *Proceedings of The 12th Language Resources and Evaluation Conference*, pp. 4615–4624, 2020.

[78] P. Smit, S. Virpioja, M. Kurimo, *et al.*, "Improved subword modeling for wfst-based speech recognition.," in *INTERSPEECH*, pp. 2551–2555, 2017.

[79] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," *arXiv preprint arXiv:1908.03265*, 2019.

[80] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.