NTNU
Norwegian University of
Science and Technology

A Normative Study on Applying Deep Learning to Native Language Identification

Iselin Bjørnsgaard Erikssen

Master of Science in Informatics
Submission date: June 2018
Supervisor: Björn Gambäck, IDI

Norwegian University of Science and Technology
Department of Computer Science

Abstract

This thesis is a normative study on various approaches within native language identification (NLI), with the intention of highlighting the shortcomings and strong points of implementing deep neural networks for this task. NLI is the task of identifying a person's first language (L1) based solely on written and/or spoken output produced in a learned language (L2). The research is mainly based around the NLI shared tasks, which are workshops where different teams participate to produce solutions that aims at bettering NLI performance. The dataset *TOEFL11: A Corpus of Non-Native English*, which was distributed in the context of these tasks, will also be used for the scope of this thesis. Deep neural networks, also commonly referred to as *deep learning*, have proven useful in many applications, including other related fields in natural language processing (NLP). In the most recent NLI shared task, there proved to still be many unanswered questions regarding the usefulness of deep neural networks in the field, and how to better utilise the available data. Through experiments and by studying related work, this publication aims to bring light to these questions using variations of recurrent neural networks as the classification models, specifically *long short-term memory (LSTM)* and *gated recurrent units (GRU)*.

Sammendrag

Denne masteroppgaven er en normativ studie på forskjellige tilnærminger innenfor feltet *native language identification (NLI)*, med den hensikt å belyse og kartlegge de temaene som er mer og mindre brukbare i forhold til effektiv utnyttelse av dype nevrale nettverk. NLI tar for seg identifisering av en persons morsmål (L1) bare ved å bruke deres egenproduserte skrifter og/eller tale i et sekundært språk (L2). Oppgaven er spesielt inspirert av forskning gjort i sammenheng med *NLI shared task*, som er en workshop hvor forskjellige forskningsteam deltar med sine egne løsninger for å forsøke å bedre forskningen innen NLI. Datasettet *TOEFL11: A Corpus of Non-Native English* som ble distribuert i sammenheng med disse workshoppene er den samme som brukes i denne masteroppgaven. Dype nevrale nettverk, også ofte kalt for *deep learning*, har vist seg nyttige i nærliggende felt slik som naturlig språkprosessering, og det har nylig vært optimisme for å undersøke hvorvidt dette også kan gjelde for NLI. I NLI shared task i 2017 var det et par løsninger som baserte seg på nevrale nettverk, men det gjenstod fortsatt mange spørsmål rundt hva som skal til for å øke prestasjonen. Gjennom eksperimenter, og studie av tidligere verk, skal denne oppgaven forsøke å belyse dette ved å implementere variasjoner av *recurrent neural networks*, spesifikt *long short term memory (LSTM)* og *gated recurrent units (GRU)*.

Preface

Identifying the origins of an unknown author of a text, or automatic evaluation of second language acquisition works and other similar problem areas are all theoretically achievable by native language identification. While it is a relatively young area of research, there has been a big growth of interest in recent years, and thanks to the NLI shared tasks the field has advanced considerably. The driving force behind this thesis is to help improving the field, by focusing on a so far little documented approach; deep neural networks in the context of NLI. The most recent contributions to the NLI shared task 2017 has shown that it is still the traditional approaches that perform best, and this thesis wants to explore how deep neural networks may perform better and why they are currently insufficient.

Acknowledgements

A big thanks to Björn Gambäck who kindly agreed to supervise this thesis, and his continuous help and support throughout the process. A thank you to the organisers of the NLI shared tasks for creating an arena for this research field to thrive and improve. And last but not least a thank you to ETS, the creators of the TOEFL11 dataset, for making this valuable resource for native language identification research. It is important to note that the opinions set forth in this publication are those of the author and not ETS.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Though processing and analysis of text is nothing new, the research field of native language identification (NLI) is still young. NLI is the task of identifying the native language (L1) of a person based solely on their texts, and speech, produced in a learned language (L2). Being able to identify this trait of an author is of great interest in several fields, such as authorship identification, second language acquisition (SLA) research, and recognition of possible multi-authored texts. This master's thesis aims to improve the results in this field by researching the possibilities of applying deep neural network models, otherwise known as deep learning.

1.1 Background and motivation

When trying to distinguish the difference between texts produced by different people, there are several traits to one's choice of expression that separates one individual from another. The idea that you are able to identify the native language of an author is based on the belief that speakers of the same language will lean towards similar habits, and make the same types of mistakes when expressing themselves in a L2 language. Fortunately, the advances and research in and around the field has shown results indicating that this theory holds truth. There has been shown that there are several traits and patterns that emerge when people of the same, or similar languages, produce texts in another language, according to Malmasi and Dras [2017]. For example that speakers of L1s that have words similar, but not identical, to those of an L2 might be more prone to misspell these as mentioned by Ircing et al. [2017].

Or other telling features such as leaving out certain pronouns, or erroneous word conjugations.

1.1.1 The NLI shared task

The NLI shared task is a collaborative event where, based on a common dataset, different teams with different backgrounds submit their solutions to contribute to improving research and results in the field. It has so far been held twice, in 2013 and 2017, with a smaller speech only contribution as a part of the 2016 Interspeech conference. In the latest 2017 shared task there were three available tracks to choose from; essay, speech and fusion. By making more data available, the organisers of the 2017 NLI shared task had hopes for experimentation with deep learning, which recently has shown great results in many other areas of natural language processing (NLP) [Malmasi et al., 2017]. The best performing solutions so far in the shared tasks have been based on traditional feature based machine learning, such as *support vector machines (SVM)* that has been the overall winner. There were also some deep neural network approaches, however, none of these performed particularly well compared to the more conventional machine learning solutions. There has, as far as related papers go, not been pinpointed an exact reason for these results. Below are descriptions of the three different tracks.

Essay track

The essay track is the task of identifying a candidate's L1 based only on their *written* English essays. In the 2017 shared task the training data was based on the training and development data used in the 2013 shared task, while the test data used new and previously unreleased data from the TOEFL11 dataset [Malmasi et al., 2017].

Speech track

The speech track was based on solely utilizing spoken 45-second English responses of the candidates in order to identify their L1. Since the raw data of the voice recordings could not be obtained, other means of data representations had to be used. This was distributed as texts transcribed from the speech responses, created manually by humans. To enhance the

accuracy of the speech systems there were also *i-vectors* provided in place of the raw recordings, which was available for the teams who requested them [Malmasi et al., 2017].

Fusion track

The fusion track is, as the name suggests, a combination of the two earlier tracks, which means that one is free to use both speech- and essay data.

Since the more data the better, resulting in the possibility to make more accurate systems, this thesis will focus on the fusion track, which allows the use of all the shared task data, and has shown great results in the most recent shared task [Malmasi et al., 2017]. The L1s in question for this classification task are French, Spanish, German, Turkish, Italian, Arabic, Hindi, Telugu, Chinese, Korean and Japanese. All the data is collected from the same data source, the *Test of English as a Foreign Language (TOEFL)* test, which evaluates a candidates written and spoken level of English. This data was released as what is known as the TOEFL11 dataset, where eleven represents the number of different L1s, and was available to all researchers participating in the 2017 NLI shared task.

1.2 Goals and research questions

Goal: Improve the performance of deep learning in NLI.

The goal of this thesis is to improve the accuracy of deep learning solutions in NLI by first locating the reasons for the poor performance until now, and then try to accommodate these. The following research questions are the main topics that will be discussed and solved in this work.

Research question 1: What can we learn from the previous shared tasks about the performance differences between neural networks and conventional machine learning?

To properly understand the state-of-the-art and the current state of deep neural nets in NLI it would be wise to utilise the findings of the earlier shared tasks and their proposed solutions.

Research question 2: How can we expand the dataset in a meaningful way to avoid overfitting?

All effective deep learning applications require a huge amount of data in order to train properly. The bigger the dataset, the better neural network can be produced. Many have posed questions especially regarding the usefulness of transcribed speech, which some mean is limited due to the short length. But in a small field like NLI with limited data available, how can a larger data collection be acquired?

Research question 3: What features are the most valuable when applying deep learning to NLI?

All models are different, and there might be some differences in which feature types and data processing is best suited for each. This thesis will try to identify some of these, based on the results from the previous shared tasks and the experiments performed in relation to this thesis.

1.2.1 Research method

To achieve these goals the following research will take an experimental approach to the effectiveness of expanding the dataset to see if it can boost the performance of deep learning in NLI. Further, there will be experimentation with different data representations, models and feature types. The 2017 NLI shared task has essay, speech and mixed tracks available, with appropriate data for all of the categories. For the scope of this thesis it is deemed appropriate to utilise data from all of the categories in order to ensure as good performance as possible. Thus this thesis will fall under the fusion category, and experiment with all of the available data in an ensemble system.

1.2.2 Thesis structure

Chapter 2 will explain and clarify different concepts, theories and technologies relevant to this thesis.

Chapter 3 is an introduction to other solutions and works related to our topic, and will also explain the current state of the art.

Chapter 4 presents a description of the experiment environment, setup and planning.

Chapter 5 is a description of the results of the experiments.

Chapter 6 will present a discussion of the experimental results in relation to the research questions.

Chapter 7 concludes the thesis, presenting the final outcome of the research and what still lies ahead for future work.

Chapter 2

Background Theory

In order to fully understand and grasp the concepts and technologies related to the thesis, this chapter will briefly explain the most important ones. The topics brought up are the surrounding problem fields, the methods that are used to solve these, and last but not least a presentation of the dataset.

2.1 Natural language processing

Natural Language Processing (NLP) is a broad field in computer science in the crossing of artificial intelligence and computational linguistics with a mixture of different disciplines. This field is concerned with the task of making computers *understand* human language, also known as *natural language*. This is backed with the desire to achieve speech recognition, text analysis, text generation and many more tasks that traditionally were believed to be a mastery limited to humans. Native Language Identification is a sub-task within this field, and therefore share many similarities and techniques with other NLP fields.

NLP and NLI should, however, not be mistakenly interchanged as being the one and the same thing. NLP is the general field as a whole, whereas NLI is a small sub-scope which requires its own techniques and methods in order to work. For example, comparing native English texts to one another, and non-native to non-native require different mindsets. But to be able to make meaningful results of an NLI solution, knowledge about NLP and its conventions is required.

2.2 Native language identification

Native Language Identification (NLI) is a field within the language processing realm that aims to solve the problem of identifying a person's first language (L1) based solely on their expressions in a learned language (L2). The first known work addressing the task of NLI was performed by Tomokiyo and Jones [2001], which focused on identifying whether utterances in English were made by native or non-native speakers by using a naïve Bayes classifier.

The main principle which this field builds upon is the assumption that a person's L1 will dispose them towards certain patterns in a learned L2 language, which can be identified generally for people of the same L1 [Ircing et al., 2017]. This phenomenon is broadly discussed as *language transfer*, and has been a controversial topic for a long time in fields such as second language acquisition (SLA), due to lack of quantitative research on a wide range of different L1s in regard to practices on the same L2 [Odlin, 2013]. However, based on the success of NLI, and other related work, the theory seems to hold. While at the same time showing the difficulties when dealing with languages of similar linguistic origins, and distinguishing between these in an NLI setting.

A workshop known as the "NLI shared task" is the biggest recent contributor to this field of research and has been held twice; April 2013 and September 2017 [Tetrault et al., 2013, Malmasi et al., 2017]. There was also a sub-challenge in the INTERSPEECH 2016 Computational Paralinguistics Challenge dedicated to NLI, but covered only that which equals scope of the speech track [Schuller et al., 2016]. Different datasets have been used for this task, however, with the release of the new dataset known as TOEFL11, the task of NLI has become easier to carry out. The dataset will be discussed more thoroughly in Section 2.8.

It is worth to mention that even though the shared task and many other works mentioned here are made with consideration to English as the L2 in question, there has recently been several studies on other languages as well. Examples of these are Norwegian by Malmasi et al. [2015], as well as three other works covering Arabic, Chinese and Finnish [Malmasi and Dras, 2014a,b,c].

2.3 Machine learning

Giving computers the ability to learn by themselves, and achieve artificial intelligence has been a desirable achievement among humans for quite some time. This dream resulted in the birth of what we today know as *machine learning*. A field which has a long history, which strictly speaking has roots back to 1763 when the famous statistician Thomas Bayes described what we today know as Bayes' Theorem [Bayes, 1763], a probability theorem which deals with the problem of making optimal decisions. Today this theorem is widely used in applications for machine learning, especially intelligent learning systems [Zoubin, 2004].

However, the kick off for machine learning as we know it today started around 1943 when McCulloch and Walter Pitts proposed a model of artificial neurons, and Marvin Minsky and Dean Emonds in 1951 built the first Neural Network Machine that was capable of learning [Russel and Norvig, 2016]. The field is therefore not an entirely new invention, but it has grown drastically in the most recent years. Machine learning is based upon statistical techniques and algorithms, giving computers the ability to *learn* from data. This data can for example be text, numbers or images, and is useful for reducing cost and optimising tasks that before could only be achieved through human labour. Examples of such tasks are *automatic classification* and *labelling*, which are important for fields such as for example NLP, and this very thesis. Traditionally slow and expensive tasks performed by humans, can and has been made more efficient by utilising machine learning methods, resulting in quick and precise possibilities for classification in a wide range of different fields [Gupta, 2018].

Machine learning methods can generally be classified as belonging to either one of two categories; supervised or unsupervised.

Supervised learning

Supervised learning regards the situations where a model is presented with a set of examples, the training set, along with a corresponding set of labels or tags to help the model decide how to classify the data. The model is able to connect observations in the examples to the tags, associating certain kinds of input with specific outputs. Therefore these models are best suited for

classification applications with expected and pre-defined labels.

Unsupervised learning

The unsupervised learning models are instructed to analyse and classify data without any given prior knowledge of potential describing labels or tags. This way the model has to decide by itself, only based on the data and its features, how to classify and interpret the information in the dataset. These kinds of models are good for when the goal is to extract information from unclassified data, for example when searching for features and patterns that are hard for humans to detect otherwise.

2.3.1 Artificial neural networks (ANN)

To be able to understand the concept of deep neural networks, one should first understand the workings of artificial neural networks; an application of machine learning, which by utilizing a network of neurons that are connected to one another by weights of importance can be used to analyze and classify data. The acquired data is *stored* in the connections in the network based on a biologically motivated idea which implies that it is possible to learn data *implicitly*. The weights of importance are shifted as the system acquires more data, deciding what is important or not by itself. That is one of the big differences between ANNs and the traditional machine learning, where the knowledge is modelled *explicitly* according to a set of rules. Though solid and efficient input encoding applied by the developer is crucial for a functioning ANN, the remarkable ability of an ANN is that it is, like a human, able to find its own solutions through analysing patterns [Deco and Obradovic, 1996]. This means that unlike conventional machine learning, they can to some extent make features by themselves.

Though the neural networks have been inspired by the functions of the human brain, it is important to understand that they are usually not meant to represent a realistic model of the brain itself. Rather, as described by Goodfellow [2016] it is more reverse engineering of the brain in order to attain desirable functionality of the brain to use in applications. Humans are very gifted at certain tasks, such as classification and image recognition, which traditionally have been very difficult to achieve in computers. Next we will

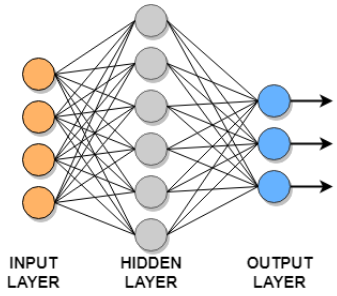

Figure 2.1: A simple *shallow* artificial neural network with one hidden layer.

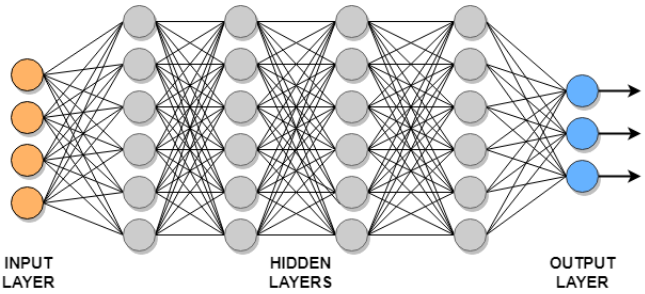

Figure 2.2: A *deep* neural network with four hidden layers.

describe an extension of ANNs which is becoming an increasing hot topic; deep neural networks, also known as deep learning.

2.3.2 Deep neural networks (DNN)

Deep learning is a hot topic among not only experts, researchers and computer enthusiasts, but also increasingly among the general public as a buzz word of sorts. They are, simply put, ANNs literally expanded a few *steps* deeper, and are also therefore known as deep neural networks (DNN). Where ANNs traditionally consist of an input layer, one or two hidden layers and an output layer such as shown in Figure 2.1, DNNs expand this by further applying even more hidden layers as illustrated in Figure 2.2. The idea of expanding the number of hidden layers is that it will be possible to achieve more accurate classification by deepening the network of information, which is why DNNs often are referred to as *deep learning*. Feature extraction and transformation is performed, and each successive layer uses the information from the previous ones as input. As a result you have achieved a method which works somewhat like the human brain, and is in theory able to learn and distinguish data by its own *reasoning*.

Since the learning and processing of the data is updated between the hidden layers of the neural network as it adapts and learns, a neural network is for the most part a *black box method*. This means that except for the input, and the resulting output, the researchers know very little of what is going on during the learning process. The result is that after using a deep learning

method, there is left little knowledge of exactly what was actually useful for the model, which is one of the main criticisms of this approach. However, on the beneficial side, the ability that deep neural networks have of finding patterns and to some extent automatically finding features in data without specific human output has shown great results [Trivedi, 2016].

But, since this approach requires huge amounts of data in order to achieve any good results, DNNs were put off for a long time, since adding even just a few layers to the traditional ANNs did not produce worthwhile results, while requiring more computational power than what was available. For each layer added to a neural network, the complexity and cost of running such a model increase along with the need of even larger datasets. Now that there is more data and computational power is available, deep learning has caught wind and has been seen to be put to good use in several applications, including the NLP field. And as described by Andrew Ng, the co-founder of Google Brain project, the performance of deep learning applications scale with increasing data amounts much better than older machine learning algorithms [Ng, 2015]. When it comes to NLP related tasks, there are in general two main types of models that are widely used today; convolutional neural networks (CNN), and recurrent neural networks (RNN). The model types of choice within this thesis are variations of the RNN methodology.

2.3.3 Recurrent neural networks (RNN)

The novel application of RNNs is deep learning based on sequential data, and presumes that these inputs are dependent on each other in some way. This is fundamentally different from vanilla RNNs that assume the inputs to be independent, which might be a disadvantage if you want to build understanding on sentence and language structures. The RNNs operate by utilising loops, allowing memory to persist when analysing new data. By doing this the network can use understanding of previous sequence patterns to understand new ones.

This makes this deep learning method a go-to for many NLP tasks, since it excels at sequential tasks that benefit from keeping track of previous information, such as for example text interpretation. But, even though RNNs in theory can work on as long sequences as one would like, they can in reality only hold memory of a few steps back in time. Known as the *van-*

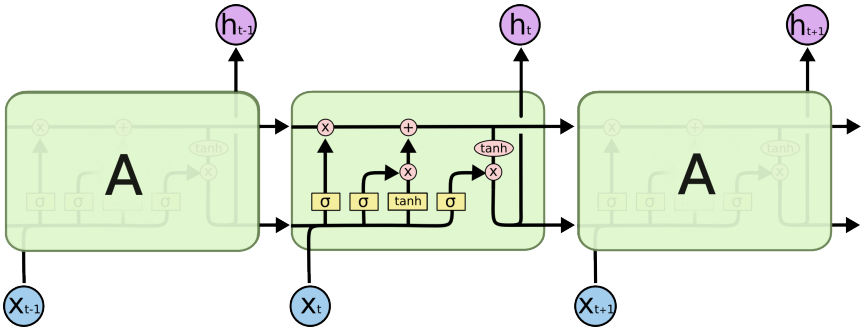

Figure 2.3: An example of the inner architecture of an LSTM node. The figure is taken from `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`, and used by permission from Crisopher Colah.

ishing/exploding gradient problem, this is an inconvenience when the goal is to make strong and robust language models, and has been solved by long short-term memory (LSTM) models, and more recently with gated recurrent units (GRU) models.

2.3.4 Long short-term memory (LSTM)

LSTMs are an extension of vanilla RNNs, that are able to capture long-term dependencies and were for the first time introduced by Hochreiter and Schmidhuber [1997]. According to their paper, the LSTMs can bridge much longer time intervals, without loss of short time lag capabilities, which is achieved by using a gradient-based algorithm. It works by utilising *gate units*, which are called *input gate*, *forget gate* and *output gate*. The process starts with the cell state, which content will be determined by those gates. The input gate controls what information flows from the input to the cell state, and what should be blocked. The forget gates' job is to decide which information should stay and which should go, by assigning 0s and 1s to each number in the cell state. The output gate is the last to come into play, and is responsible for filtering the now updated cell state before releasing it to the network. The network has to learn which error signals to keep, and which to get rid of by learning from previous layers and iterations by running several times. In Figure 2.3 the general architecture of a normal LSTM node is presented. The horizontal line at the top is the cell state, which is controlled and eventually updated by the gates below before it is released into the next

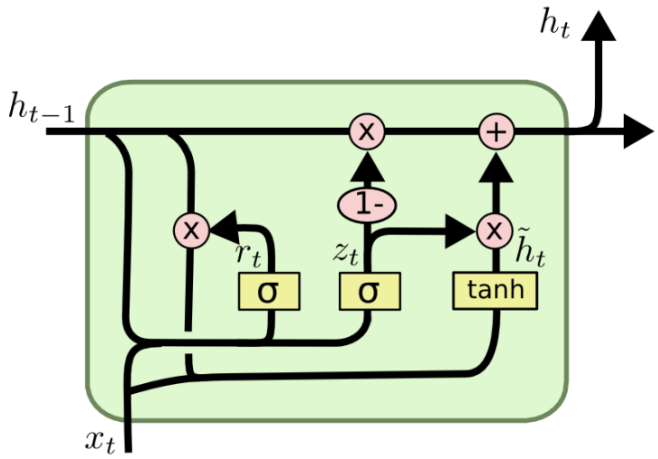

Figure 2.4: An example of the inner architecture of a GRU node. It is slightly different than that of an LSTM, as one can see here. The figure is taken from `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`, and used by permission from Crisopher Colah.

part of the network through the output gate.

2.3.5 Gated recurrent units (GRU)

A GRU model is a variation of the LSTM presented by Cho et al. [2014], where instead of three gates, it has two. This is because it merges the forget and input gates into a single *update gate*. This model has grown increasingly popular, much because it is much quicker to train than the LSTM, and because it performs well in cases working with less data.

As can be seen in Figure 2.4 the structure of a general GRU node is very similar to the LSTM node presented in the previous section. In Figure 2.3 there are two separate gates on the left side, the input and forget gate, which in the GRU has been simplified into one pipeline. The resulting structure is two gates, and a simpler structure which has lead to faster training times.

2.3.6 Support vector machines (SVM)

Support vector machines (SVM) are a popular approach in NLI, and also the main ingredient in the best performing systems in the 2013 and 2017 NLI shared tasks. SVMs are supervised learning models especially made for classification and regression analysis. The general inner workings is that the model assigns examples to each category, creating a mapping space where the gap between each category is as wide as possible. Usually these models operate by classifying data as one of two categories, otherwise known as binary, based on whether the data falls on one side or the other of a hyperplane. However, it is also possible, which has also been shown in the shared task submissions, to use SVMs to classify data into multiple classes.

This classifier is effective when it comes to text classification and is quite reliable thanks to the simple decision boundary, making it little prone to overfitting. Additionally the low computational cost, rigidness, and ease of use has made this a popular approach in many classification tasks.

2.4 Text preparation

Usually when working with raw text it is common to analyze and process the text in order to extract the most useful features to use for later application. This is because depending on the use case for the data, there are usually many aspects of the text that are less useful and can be removed in exchange for better performance. In addition to this, there are some steps that are required in order to be able to use the data in a machine learning setting. Some of these methods will be described in the following section.

2.4.1 Tokenisation

When analysing texts, it is often of interest to split the longer texts into sentences or words. This way it becomes possible to look at inter level word relations, vocabulary, grammar and much more. Tokenisation is one such method that segments raw text into what is referred to as *tokens*. This is a very common pre-processing step in natural language processing that takes place early on in the information extraction process.

There are many different ways to create these tokens, with different rules for how and where to split text depending on the later intended usage of the tokens. The most common standard approach is to split text based on white-space, thus usually producing word tokens. Though it is important to keep in mind that this the tokenization approach can be fundamentally different depending on the target text language. For example Chinese, Japanese and Thai text splitting on white-spaces would be meaningless as these languages have no such feature [Trim, 2013].

2.4.2 Stop words

In most NLP or information retrieval (IR) applications stop words are a part of sentences that is considered disposable, since they are not very useful for distinguishing between documents. Removing them makes the data less computationally expensive, and reduces the dimensionality, making the analysis faster [Vijayarani et al., 2015].

2.4.3 Stemming

Stemming is another pre-processing method which is popular in NLP and IR, which reduces the words to the *stem* form in order to reduce the time complexity and save memory space. This means that the sentence "The flowers bloomed in different colours" would become "The flower bloom in differ colour". Thus the system does not need to consider as many different word variations, since all variations of a word will be represented as the same word stem.

2.4.4 Word embeddings

The meaning of the term *word embeddings* is simply to represent words according to the environment they are in. In the context of texts in computer science, this is usually the act of transforming raw strings into representations of numbers. Many learning machines are incapable of reading strings when analysing natural language, which means that they have to be translated to something it understands better; numbers or binary codes. Therefore creating word embeddings is a crucial pre-processing stage, which in some cases has to be done in order to get a learning model running at all. These representations of the words will be used for the computer to capture the

semantic relationships and contexts in the sequences they are given. There are in general two different approaches to word vectorization; frequency- and prediction based.

Frequency based

Frequency based word embeddings are usually pretty simple, where the length of each vector equals the number of unique tokens in the vocabulary extracted from the data. The idea behind these methods is generally to compute the frequency of terms in a dataset, and then use this data to determine the importance and/or relations between said terms. Examples of such methods are count vectorisation and tf-idf vectorisation, where the latter is largely the first concept taken a step further and the one used in the experiments of this thesis. It stands for *term frequency-inverse document frequency*, and uses the occurrences of words in the entire corpus to determine the importance of these. It penalises the most common words in order to pick up on the more unique and informative words, which is expected to better interpret the content of a text.

Prediction based

A newer approach to creating word vectors, which has risen in popularity, is prediction based vectors. Instead of having vectors that are as long as the vocabulary, these methods calculate the probabilities of the words and transform the sequences to vectors representing only the words in that exact sequence. This produces shorter vectors than the frequency based approach, however, they are much more computationally expensive because they work by using unsupervised shallow neural networks. Some of these methods are, for example, revolutionary in the way that it now is possible to make computations on the words, such as *woman + king - man = queen*. There are two different methods within prediction based vectors; Continuous bag of words (CBOW) and skip grams.

CBOW aims to predict the probability of a given word or term based on its context. This means that given the other surrounding words you can predict which word is most likely to come next. These are in general quite quick to train compared to skip grams, and perform well on finding frequent words.

Unit	Sequence sample:	1-gram (unigram)	2-gram (bigram)	3-gram (trigram)
Character	The_flowers_bloomed_in_the_colours_of_the_rainbow	T, h, e, _, f, l, o, w, e, r, s, _, b, l, o, o, m, e, d, _, ...	Th, he, e_, _f, fl, lo, ow, we, er, rs, s_, _b, ...	The, he_, e_f, _fl, flo, low, owe, wer, ers, rs_, ...
Word	The flowers bloomed in the colours of the rainbow	The, flowers, bloomed, in, the, colours, of, ...	The flowers, flowers bloomed, bloomed in, ...	The flowers bloomed, flowers bloomed in, ...

Figure 2.5: Character and word n-grams between the values of 1 to 3.

Skip grams are the opposite of CBOW, which means that provided a word it will predict the context. These are slower to train, and in general represent rare words and phrases well if you are working with small amounts of data Mikolov [2013].

2.4.5 Sequence padding

Sequence padding is performed to ensure that all sequence vectors are of the same length. When using machine learning, many models require that all sequences fed into the machine are of the same length in order to make comparisons and calculations. One of the usual methods of padding is to add 0s in to the end or the start of the sequences until they reache the target length specified by the program. It is also possible to pad sequences by using other symbols, or words, instead.

2.5 Features

Before feeding the data to the learning model, it is common practice to apply some feature extraction to the data beforehand. This is the first step in text-mining; the basis of successful information retrieval systems.

2.5.1 N-grams

An n-gram is a sequence of tokens, where n is the number of tokens combined in each unit. These are usually extracted from text or speech data, and are useful for identifying sentence structures and grammar. They were the overall best performing feature type in the 2017 shared task [Malmasi et al., 2017], and is a central approach in natural language processing. Among the n-grams, there are two types that are particularly popular; character based and word based.

Example sequence	The	flowers	bloomed	beautifully	in	spring.
Tags	DET	N	V	ADV	P	**N**
Representation	(The, DET)	(flowers, N)	(bloomed, V)	(beautifully, ADV)	(in, P)	(spring, N)

Figure 2.6: An example of part-of-speech tagging of a short sequence. Notice that the last word *spring* can have several possible tags depending on context. An automatic POS tagger could classify this as either a verb, adjective or a noun with each tag indicating quite different meanings.

Character n-grams

Character n-grams are made by splitting the input string into individual characters, spaces included, and then collecting these in sequences of length n. Usually these take up much less storage space than word n-grams, and in general also hold less information. But they have proven to be especially useful for anomaly detection, such as spelling differences, errors and L1 specific sequence patterns. Look to Figure 2.5 for an example of character n-grams from 1 to 3.

Word n-grams

The word n-grams are made similarly to the character n-grams, but instead of splitting the strings into individual characters they are split into individual words. They are useful for capturing sentence structures, relations between words and context. L1 specific words and sequences are also traits that could be revealed by using these features. An example of word n-grams between 1 to 3 can be seen in Figure 2.5.

2.5.2 Part of speech tagging

Part of speech tagging, commonly referred to as POS tagging, is the process of classifying words into their specific part of speech and labelling them thereafter. A words part of speech is based on the basic word classes present in languages, such as for example verbs and nouns, and therefore signifies the syntactic role of a word. By giving the words their grammatical tags, the idea is to capture the grammatical structure of sentences, which can be very useful for fields such as language identification. An example of a POS tagged sequence can be seen in Figure 2.6. It is important to keep in mind that since there are several words that can have different meanings depending on

context. Even though a sentence has been POS tagged it is not necessarily semantically correct. But despite this weakness, POS features have proven to produce good results, given that they are still considered an important and widely used method in NLP research.

2.6 Tools

This section will explain the main tools that have been used for realising the experiments in this thesis.

2.6.1 Python

As a simple and straightforward programming language, Python is high-level, general-purpose, very adaptive and easy to work with. Python is very popular, and widely used in both NLP and NLI, as it is a fairly fast and very convenient method to apply both string processing and machine learning algorithms. The integrated development environment (IDE) of choice for this thesis has been Spyder run in the Anaconda environment [Raybaut, 2009, Anaconda, 2012]. This platform allows for convenient analysis of variables and functions, with a built-in console, detailed variable overview, and editor in one.

2.6.2 TensorFlow

The DNN models will be made using TensorFlow, which is a framework built by Google [2015], which facilitates development of machine learning and deep learning research. It is a very popular option for many companies and researchers, and makes it easier for everyone to create machine learning applications.

The technology is based on numerical computation by using data-flow graphs, and serves as a framework that simplifies the handling of these computations. TensorFlow is compatible with Python, and can therefore fit comfortably in a Python project.

2.6.3 Keras

Keras is a high-level application programming interface (API) written in Python for creating neural networks, especially for those focusing on fast implementation and running experiments [Chollet, 2015]. It is able to run on top of TensorFlow, and is also supported by Google's TensorFlow team. Since Keras is made for Python, it is quite straightforward to use and very extensible. This is very convenient for building the deep learning models for carrying out the experiments.

2.7 Evaluation

In order to measure the performance of an application, it is important to evaluate it based on the criteria used in the respective field. This section will describe the methods used to evaluate applications within NLI, that also apply to the models in this thesis.

2.7.1 F1-score

Though accuracy is a very common metric used to measure a system's performance, the 2017 NLI shared task decided to use f1-scores as their official evaluation metric, with accuracy as an addition for completeness [Malmasi et al., 2017]. The F1-score is a harmonised mean of precision and recall, which takes both false negatives and false positives into account. Where accuracy only considers the amount of correct results compared to the total number of objects, f1 scores give a more complete picture of the performance of a classifier system. The formula for calculating the f1-score is as follows:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

2.7.2 K-fold cross validation

K-fold cross validation is a popular model evaluation method, that divides the training data into k smaller subsets, and withhold parts of the dataset for testing purposes. It repeats this process k times, imitating the seen vs. unseen data scenario several times [Schneider, 1997]. This is because one of the problems with training machine learning models, is the uncertainty of

how it will perform on unseen data. The k-fold cross validation method allows for the model to repeat this process, giving the developer the possibility to make a more realistic measure of the models true performance in the wild.

When it comes to deep learning models, the k-fold method might be slightly inconvenient as an evaluation method since these models are notoriously time consuming. K-fold cross validation is useful when dealing with relatively small datasets, but deep learning uses a lot of data and computational power and therefore tend to avoid using the k-fold method if possible.

2.8 Dataset

This section will describe the TOEFL11 dataset and its origins. This is the only dataset that will be used in this thesis.

2.8.1 TOEFL11 dataset

The dataset which was recently used by the 2017 NLI shared task, as well as the earlier 2013 shared task, leading to great results as seen in the report by Malmasi et al. [2017], will also be used in the solution of this thesis. The official name is ETS Corpus of Non-Native Written English, also known as *TOEFL11*, and was developed by *Educational Testing Service* with native language identification being its intended use [Blanchard et al., 2014]. This is the only corpus to date which is specifically made for NLI research, and also the largest available dataset suitable for this task. Before this corpus came to be, the most common dataset used in earlier NLI research relied on the *International Corpus of Learner English (ICLE)*. However this dataset was not made with NLI in mind, and is therefore quite topic biased, in addition to only having seven different L1s [Blanchard et al., 2013].

TOEFL11 offers eleven different L1s, and has been distributed evenly among all the different languages as much as the dataset allows in order to prevent bias. The different languages are Korean (KOR), Chinese (CHI), Japanese (JPN), Telugu (TEL), Hindi (HIN), Arabic (ARA), Turkish (TUR), French (FRE), German (GER), Spanish (SPA) and Italian (ITA). The data was collected between 2006 and 2007 from a high-stakes college-entrance test known as the *Test of English as a Foreign Language (TOEFL)*, which is a standard-

ised test used to measure a candidate's English proficiency [Blanchard et al., 2013]. These standardised tests consist of one part which involves writing essays based on a given task, and another which requires the participant to answer a questions orally in English. It was released in 2013, however, shortly after the initial release, an additional 1100 candidates were added with corresponding essays and speech to the corpus, bringing the total up to 13,200 different elements.

In the 2013 shared task, each row in the provided dataset labels file consisted of the candidate test takers id, essay prompt, their L1 and the overall performance on the test. However, in the 2017 version, the one used in this thesis, the candidate performance data has been removed, and speech prompts have been added instead. The distributors of the dataset have split the dataset into training, development and test sets with a ratio of 83.3%, 8.3% and 8.3% respectively. This means that there are 1100 data elements from each language in the training data, and 100 of each in the development and testing data. All the texts, both essays and speech transcriptions, are presented in UTF-8 format, with both tokenised and original raw texts available. The test data was not available for use in this thesis.

2.8.2 Essay data

The essay data has been used in both of the previous NLI shared tasks, and was the only available dataset in 2013 where the shared task focused solely on using essay data.

The length of each essay text is quite long compared to the speech ones, and the average sequence lengths also vary between the languages. However, according to the dataset creators, they represented the data as evenly as the available data allowed. The average essay length can be seen in Figure 2.7. The longest text in the essay dataset is 799 words long and the shortest is 2, with an average of 315 words per essay.

2.8.3 Speech data

Speech data was first used in the context of the NLI shared task at the Interspeech conference in 2016 [Schuller et al., 2016], and later at the 2017 NLI shared task. In the 2016 Interspeech there were raw audio files available

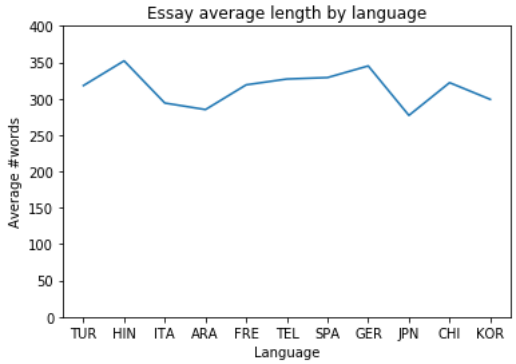

Figure 2.7: The average length of essays by language.

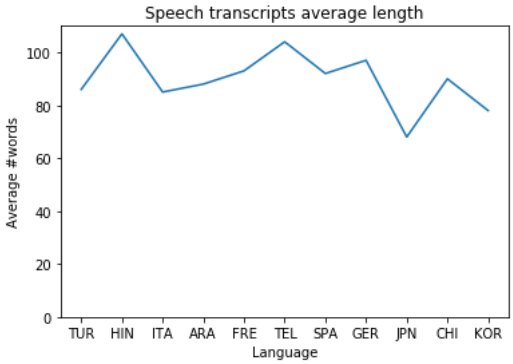

Figure 2.8: The average length of speech transcripts by language.

for use, but this was not the case for the 2017 NLI shared task [Malmasi et al., 2017]. The speech data available for use consists of manually written transcripts of 45 second long audio recordings, with associated i-vectors. Participants of the TOEFL exam had to perform both written and oral examinations, and the speech data is therefore related to the same participants as the essay data.

As mentioned briefly previously, the speech transcriptions are on average very short compared to the essay ones, which can be seen in Figure 2.8. This is allegedly caused by the short answering time of this test, and the longest speech transcript is 202 words long while the shortest is 0. The average length across all the transcripts is 90 words.

I-vectors

Due to privacy concerns, i-vector acoustic features were distributed to make up for the absence of the raw audio files. The term i-vector is an abbreviation of *identity vector*, and represents speech signals. The i-vectors provided by the NLI shared task are 800-dimensional, and they each represent frame-level acoustic measurements extracted from the speech responses [Malmasi et al., 2017].

Chapter 3

Related Work

In the field of Native Language Identification there are already several solutions available. This section will present some of the works most relevant to the scope of this thesis. The relevant works described here were all made by teams participating in the 2017 NLI shared task, and the overview of all the contributions and complete results can be found in Malmasi et al. [2017]. Based on the results achieved during the shared task of 2017, the teams were placed in a ranked group according to their performance with rank 1 as the best and rank 5 as the worst. There were a total of five ranks, and the number of teams in each group was based on the proximity of performance rather than a predetermined amount. Each track had their respective list of rankings. Results of shared tasks prior to the one in 2017 can be found in Tetrault et al. [2013] and Schuller et al. [2016].

3.1 CEMI (Ircing et al., 2017

CEMI participated in all three different tracks, and achieved rank 1 in the speech and fusion tracks with F1 scores of 0.8607 and 0.9257, respectively. For the written essay task, their F1 score of 0.8536 landed them in the rank 2 category, which means that this team performed well overall in the shared task. They accredit their best result to a feed forward neural network, using mainly i-vector features for speech and a large variety of n-grams for the essays. For both essays and transcripts they extracted unigrams, bigrams and trigrams, however, ended up only using the n-grams extracted from the essays. Character n-grams ranging from 3 to 5, and POS n-grams from 1 to

5 [Ircing et al., 2017].

In the CEMI paper published in the 2017 shared task it was recommended to leave out the speech transcripts, and rather focus more on i-vectors that seemed to overshadow both POS and character n-grams based on transcripts. Further they reported that speech and text features compliment each other well, which is also reflected by the results.

3.2 ETRI-SLP (Oh et al., 2017)

ETRI-SLP also participated in all three tracks, with their best performance in the fusion track. In essay-only they placed in rank two with an F1 score of 0.8601, speech-only as rank 1 with 0.8664 and fusion as rank 1 with 0.9220. The features used in the system were latent semantic analysis (LSA) and linear discriminant analysis (LDA) features based solely on word 1- to 3-grams and character 4- to 6-grams for both the essay data and the speech transcript data, with an addition of normalised i-vectors with LDA feature extraction for the fusion track. They experimented with two distinct approaches, one performing late fusion between the textual features and the i-vectors and the other with early fusion before feeding the data into a DNN [Oh et al., 2017].

3.3 L2F (Kepler et al., 2017)

The L2F team participated solely in the fusion track, where their system placed in the rank 3 category with an F1 score of 0.8377. Wishing to avoid heavy use of hand-crafted features, the focus was on using simple word and sub word features. The conference paper (Kepler et al. [2017]) reported that by utilizing ANN and DNN methods for learning, in their case GRU, one can avoid having to rely on specially engineered features to achieve good results. Further, both n-grams and i-vectors were considered great contributors to the systems' performance.

However, like most of the deep learning approaches, this system did not perform particularly well in the shared task, reflected by the fact that it ended up as the seventh best out of eight teams above the baseline in the fusion track. The team comments in the paper that while i-vectors without

a doubt is a state-of-the-art approach to NLI, the simple approach in this solution did not work in their favour when using neural networks.

3.4 ItaliaNLPLab (Cimino & Dell'Orletta, 2017)

The best performing essay track team in the 2017 NLI shared task was the ItaliaNLPLAB team with an F1 score of 0.8818 thanks to a 2-stacked sentence-document architecture. In the paper [Cimino and Dell'Orletta, 2017] the main novelty of this systems approach is described to be that it is able to exploit both local sentence information and a rich amount of features from the documents themselves by combining sentence and document classifiers. The features used were linguistic description, lexical, morpho-syntactic, and syntactic information.

The team advocates for the importance of sentence level classifiers and utilizing local information, which is also reflected in the work submitted for the task. This might be useful results showing that in NLI it is important to focus on the smaller details which can be found in the sentences, since these can hold useful information in identifying a specific L1.

3.5 UnibucKernel (Ionescu & Popescu, 2017)

The only team to participate in all three tracks and reaching a rank 1 level in each was UnibucKernels multiple kernel learning system. Achieving an F1 score of 0.8695 in the essay track which put this system as the 6th best submission out of 17. In the speech and fusion tracks it placed as number one in both, with F1 scores of 0.8755 and 0.9319 respectively. This makes UnibucKernel the overall best performing team in the 2017 NLI shared task, mainly basing the solution on character n-grams and i-vectors. The speech transcripts did not provide enough information to accurately distinguish the L1s, and speech did not seem to benefit from long n-grams. On the other hand the essay data responded best to n-gram values between 5-9, and the best performing solution which was submitted to the shared task combined 5-9 n-gram essay data with i-vectors and 5-7 n-gram speech data [Ionescu and Popescu, 2017].

These findings could prove useful for identifying that speech transcripts might be much too short compared to the essays to be as useful to extract information, which was also mentioned earlier by CEMI. Character n-grams, which this system was based the most upon, might also be less useful in speech transcripts because they are written by more proficient, if not native, English speakers. The comments in UnibucKernels' paper show how important it is to include the i-vectors in order to properly utilise the speech data.

Chapter 4

Methodology

Description of the experiment environment setup, components and methods, and the plan for these experiments.

4.1 Classifiers models

NLI is a discipline which deals with classification of user input, as well as being a supervised training problem. This means that we have the luxury of labels describing the dataset elements, which can be used to instruct the model how to classify a language. Supervised learning methods are therefore very fitting for the problem task, and the classifiers of choice for the experiments are long short term memory (LSTM) and gated recurrent unit (GRU). As mentioned in Section 2.3.3, these are popular neural networks in NLI and NLP, and they were also used in the 2017 NLI shared task in for example the solution of the L2F team presented in Section 3.3 [Kepler et al., 2017].

By looking at the research conducted by Yin et al. [2017], it seems that GRUs outperform LSTMs at almost every NLP task they tested, even if just by a small margin. And even though as mentioned by Kepler et al. [2017], GRUs are in general faster to train, the experiments will still be tested with both LSTMs and GRUs since they are almost identical to implement.

Parameters	*Values*
Layer type	Dense, LSTM, GRU, Dropout
Input shape	batch size, sequence length, input size
Number of hidden layers	0-many
Number of units per layer	1-many
Output shape	1-many

Table 4.1: Parameters that define the classification environment for LSTM and GRU. The values provided are the standard values and ranges.

4.1.1 Model environment

There are many different parameters and features that are essential for how the model trains on the dataset. This goes from, for example, the number of layers, number of units in each of these layers, and the layer types. Different tasks require different tweaking, combinations and setup of these. Table 4.1 shows the parameters deemed the most important to keep in mind for creating the classification models for this thesis.

Layer types

The *LSTM* and *GRU* layers, as described in Chapter 2 in Section 2.3.4 & 2.3.5, are the ones the model uses to perform the core operations on the data. *Dropout* layers are used for regularisation, where you randomly deactivate certain units in order to avoid overfitting. This might be an important step, especially since LSTMs are known for being prone to overfitting. Last, but not least, the *dense* layer changes the vector dimensions. This layer is connected to all the units in the previous layer, and is important especially to create the output layer, which in the case of this task is of the size of the number of L1s in the dataset.

Hyperparameters

The *input shape* will vary between the input types, for example, that the i-vectors are 800 dimensional, while maybe the transcriptions would be 100 dimensional. How deep the model is depends on the number of *hidden layers* that make up the black box between the visible input and output layers as described in Chapter 2 Section 2.3.1. Each hidden layer has a number of *units*, also known as neurons, that decide how complex models can be made

from the data. This amount is reliant on the dimensionality of the input shape, and therefore may change drastically between different tasks. The amount of neurons will be the same for each layer in the respective models.

4.2 Data preparation

Before converting the data to vectors or performing other feature extraction methods, there are a few precautions that have to be made that can have an impact on the efficiency of the classification later on. These, along with observations of the data itself will be presented here.

4.2.1 Essay data

Out of all the text data used in the NLI shared tasks, the essays have shown to give the most significant improvements to good language classification. This may be because they hold many important features describing different L1s, and the following will describe some hypotheses on what these might be and should be used for in the experiments.

Misspellings

When inspecting the essay data, it becomes clear that there are many misspellings, as well as semantic errors, that are important for choosing the text pre-processing methods. Because of these misspellings, no stemming will be applied, even though this is a common step as mentioned in Section 2.4.3, since these misspellings by themselves are features of different language backgrounds.

Lowercase vs. uppercase

Transforming all words to lowercase is another common pre-processing step, which is done automatically by many off-the-shelf solutions, and is usually when dealing with text processing a smart move in terms of memory efficiency. When working with normal text processing systems, the capitalisation of a certain word may not be of much significance. But for native language identification, this could be another feature one can use to identify specific languages.

Punctuation and stop words

There will be no removal of punctuation symbols, especially commas, since some L1s have very different rules from one another. Stop words will not be removed either, for the same reasons as punctuation.

The experiments will not make much pre-processing on the texts, since they hold many features that may be important for the model. But there will be experiments on preserving uppercase words, to see if this has as much to say in the identification of a language as previously hypothesised or not. Tokenisation, as described in Section 2.4.1, will also be applied.

4.2.2 Speech transcription data

The speech data is quite different from the essay data, in more ways than just document lengths as pointed out in Section 2.8.3. By inspecting the data, there were several strange occurrences that will be pruned from the texts. Like the essay texts, tokenisation will be applied.

Missing tokens

These occurrences are all similar, in the format of numbers, percentage symbols or combinations of both between inequality signs, and seem to represent missing tokens. Probably, these are traces of some sort of automatic system that has handled the files, and they were present in all of the files that were empty as well as in place of seemingly missing words mid-sentence. These will be removed before any further processing of the raw texts.

Stammering and hesitation

Since the transcriptions are directly reflecting the spoken responses, there are occurrences of words such as “um” and “uh”, and unfinished words that are also present represented by a single single hyphen at the end of the word. There will be made two versions of the texts, one with the “um” and “uh” and one without. The stammering will be left out this time, to limit the scope of the thesis.

High level of English

While the speech has very distinct vocabulary and grammar usage, the transcriptions themselves are clearly transcribed by someone with high level knowledge of English, which was also pointed out by Ionescu and Popescu [2017]. The words are mostly correctly written, in contrast to the essay data where the grammar might be tidier but the words are written strangely. How to write a word based on a speech transcription is harder to determine, unless there is some clear mispronunciation. Because of this, preservation of capitalised words will be overlooked, however, stemming will not be performed since the transcriptions the utterances are still sometimes strangely conjugated.

4.2.3 i-vector data

Given the good results achieved by utilising i-vectors, as mentioned in the related work in Chapter 3, it would be wise to include these in the experiments. The data consists of 800-dimensional vectors, which when read by humans do not make much sense. There will be no specific pre-processing performed on these, other than testing feature extraction with LDA, inspired by the work of Oh et al. [2017] and Ircing et al. [2017], and feature selection with a tree classifier.

4.3 Features

The features used for the experiments will be presented here, and further reading on these can be found in the previous Chapter 2 in Section 2.5.

4.3.1 N-grams

Word- and character n-gram features will be extracted from both the essay texts and the speech transcriptions. These have shown to be the most informative and useful features, and have in some manner been used by most of the participants in the 2017 NLI shared task [Malmasi et al., 2017]. There will be extracted n-grams of range 1 to 4 from both text data types, and these will be used in different combinations in the experiments.

4.3.2 POS tag

Initially it was planned to extract POS tags as an additional feature to be used. This will be dropped in favour to the n-grams, because they have proven to be less useful than both word and character n-grams as described by Ircing et al. [2017]. Ionescu and Popescu [2017] also pointed out that POS tags are inferior to character and word n-grams, especially for the transcription data. For the sake of limiting the scope, the POS tags were therefore dropped in favour of other features.

4.4 Feature extraction

Since the data is the core which gives meaning to the whole system, it is important to handle it with care. How to process the data before training, extract the appropriate features, and how to feed it into the model are all important steps that must be considered when using machine learning. Feature extraction is the process of preparing the data for further processing at later stages.

4.4.1 Word embeddings

The word embedding methods that will be used in these experiments will be from both frequency based and prediction based approaches, explained in Section 2.4.4, with one from each. For the frequency based solution the tf-idf vectoriser method will be applied, while Word2Vec will represent the prediction based approach. Due to limited time, the experiments will be limited to these two methods.

Word2Vec

The Word2Vec method is a popular approach to prediction based word vectors, and is either built by using CBOW or skip gram methods. There will be experimentation with Word2Vec using both skip gram and CBOW, to see which method has the greatest impact on this task.

tf-idf vectoriser

Explained briefly in Section 2.4.4, the tf-idf approach is a frequency based word embedding method that intends to reflect how important a word is to a document. It achieves this by measuring the frequency of all the terms in the documents, and then penalising words that are too common while trying to find the words that are more distinctive. Because of this, the tf-idf vectoriser was chosen as the frequency based method for the scope of this thesis, over the simpler word count vectoriser.

4.4.2 i-vector

The i-vector data is very different from the raw texts provided in the dataset. However, they have proven useful, and many of the participants in the 2017 shared task found ways of utilising these. As mentioned earlier in 4.2.3, a seemingly successful approach has been applying LDA to the i-vectors. A tree classifier known as the *ExtraTreesClassifier* will be applied to evaluate the effects of using other methods on the i-vector data.

4.5 Experiment plan

The experimental plan will be presented, along with the underlying arguments for these choices.

4.5.1 Experiment flow

When working with supervised learning, there is a common setup for how to run from training to the final performance result. In Figure 4.1 this flow of operations is described. For all models tested during these experiments, this is the step-by-step process that shall be followed. All models train from scratch in order to clearly distinguish the effectiveness of the different models. The experiments will be created by using the *Python* programming language, utilising *TensorFlow* and *Keras* to build the models. More information on these can be found in Section 2.6.

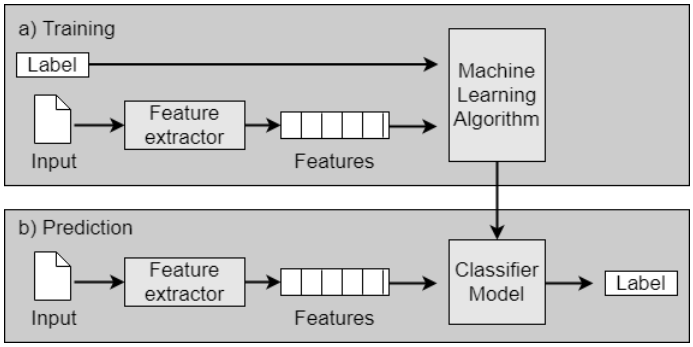

Figure 4.1: The general step by step process of supervised machine learning. The original diagram was made by Steven Bird et al., distributed under the *creative commons* licence, and can be found at `http://www.nltk.org/book/ch06.html#fig-supervised-classification`.

4.5.2 Separate testing

Before experimenting on the full fusion system, experiments will be conducted individually on each type of data to find the model environments that work best. This is because the data types are fundamentally different in many aspects, and will require different preparation and training environments. In addition, training deep neural networks is notorious to be a time consuming task, so locating the individual optimal parameters ahead should be favourable. In Table 4.2 the different features, and the n-range they will be extracted in, is described. Experimenting with tweaking the parameters presented in Table 4.1 for each respective data type model is done in order to examine what parameters are the most useful when creating the fusion model.

4.5.3 Fusion testing

The scope of the thesis is to evaluate the neural network methods against as much data as possible, and since the data is of different dimensions and types, the data shall first be input separately, trained separately, and then be concatenated to produce the end result which will be referred to as *fusion testing*.

Category	*n-gram*
Essay	
Character	1-4
Word	1-4
Speech	
Character	1-4
Word	1-4
i-vector	
LDA	
ExtraTreesClassifier	

Table 4.2: Features for use in the experiments per data type.

4.5.4 Word embeddings

During the testing process the best method between frequency or prediction methods that produce the best results is decided. After this decision has been made, the most effective method will be applied to further tuning and changes. The earlier mentioned concerns about capitalisation and removal of certain parts of the transcription data shall also be determined after the choice of embedding method.

Further there will be tests with the parameters within the word embeddings themselves as well. For the Word2Vec approaches this means tweaking vector length, minimum word occurrences and whether or not it is using skip gram or CBOW. In the case of tf-idf vectorisation this is related to deciding the maximum number of features.

4.5.5 Sequence padding

Researching methods for enriching or increasing the data in some way has been described as one of the topics of further work as described by Malmasi et. al. in the conclusion of the 2017 shared task. This has also been described in the second research question in Section 1.2, and will for the experiments of this thesis be conducted through sequence padding experiments. Naïve padding of the essay and speech datasets will be performed, which means appending parts of the same sequence to the end in order to bring shorter sequences to the target sequence length.

4.5.6 Evaluation

For evaluating the models there will be two methods used; recall/accuracy and f1 score. As described in Section 2.7.1, the f1 score is the official metric that was used in the 2017 NLI shared task, and therefore also relevant to the experiments of this thesis. The *macro averaging method*, that was used by the shared task participants, is also the one to be used in this thesis. Accuracy is included for clarity, and because it is the standard result metric when using Keras to build neural network models.

The second evaluation method is *k-fold cross validation*, presented in Section 2.7.2, and will be used at the end of the experiment phase to measure the performance. This method is often avoided in deep learning, however, since it is relatively small dataset the k-fold method could still be useful.

Chapter 5

Experiment Results

Following the methodology chapter, this chapter will describe in detail the actual execution and results of the experiments. In depth discussion, and reflection on these results will be presented in the next chapter.

5.1 Fused Classifier

After testing each of the data types individually, they were concatenated and trained together in a stacked classifier. First the fused classifier is presented, followed by the other experiments that led up to these results.

5.2 Structure

After testing each data type individually, the most optimal model structure that was found is presented in Figure 5.1. The system which proved to give the best results was a GRU network with 3 hidden layers with 100 nodes each for the essay data, 2 hidden layers with 60 each for the speech data, and last but not least 1 hidden layer with 60 nodes each for the non-processed i-vector data. After performing 4 fold cross validation on the best model measuring 2 and 3 hidden layers, it appears that the difference in performance with using 3 hidden layers instead of 2 is an increase by approximately 3%. Regarding the number of nodes, the desirable number for the essays seem to be between 100 to 200 nodes, while the speech data responded best to between 40 to 60 nodes.

RNN model comparisons Word2Vec										
acc	*f1*	*data*	*classifier*	*hidden layers*	*nodes per layer*	*sg / cbow*	*embed. length*	*sequence length*	*features*
42.09%	-	e,s,i	LSTM	2/1/2	40/10/10	sg	200/100	200/100	w1
42.36%	41.43%	e,s,i	GRU	2/2/1	100/60/10	cbow	100/100	350/150	w1
35.54%	-	e,s,i	GRU	3/2/2	100/60/10	cbow	200/100	450/150	w1
57.90%	57.67%	e,s,i	GRU	3/2/1	100/60/60	cbow	200/100	350/150	w1
34.09%	-	e,s	LSTM	3/2	100/60	sg	100/100	350/150	w1
49.00%	48.41%	e,s	GRU	3/2	100/60	sg	100/100	350/150	w1
55.09%	55.49%	e,s	GRU	3/2	100/60	cbow	100/100	350/150	w1
45.99%	-	e	LSTM	3	200	cbow	100	200	w123
46.09%	-	e	LSTM	2	200	cbow	100	200	w123
44.09%	-	e	LSTM	2	200	cbow	100	400	w1234
30.67%	-	s	GRU	2	40	cbow	100	150	w1
36.14%	-	s	GRU	2	40	cbow	100	150	w1c6
tf-idf									
acc	*f1*	*data*	*classifier*	*hidden layers*	*nodes per layer*	*max features*	*select*	*sequence length*	*features*
11.27%	-	e,s	GRU	3/3	100/30	200/100	T	400/100	w1
10.00%	-	e	GRU	3	90	300	F	350	w1
9.72%	-	e	GRU	3	150	600	F	350	w1
9.81%	-	e	GRU	3	150	600	T	350	w1

Table 5.1: The most prominent results from the different models' performance on the development set. The marked feature cells indicate lowercase essay features.

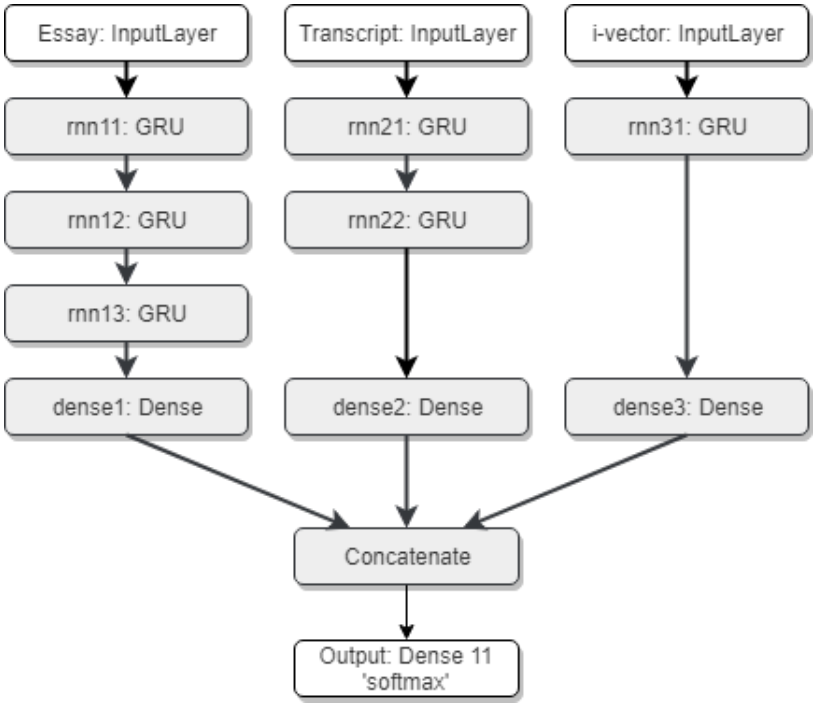

Figure 5.1: Architecture of the fusion model that had the best score among the experiments. The grey layers represent the hidden layers, while the white ones are the visible layers; input and output. It was trained for 150 epochs with a batch size of 60.

For the i-vector data there did not seem to be much improvement when increasing the number of layers, and therefore final model for the i-vector was chosen to be a shallow 1 hidden layer structure with 60 nodes.

Since all the data required different dimensions in order to produce worthwhile results, the last layer of all the individual models had to be connected to dense layers before concatenating. There were several attempts to apply RNN or LSTM on the concatenated data, however there was no solution found during this time due to the dense layers producing two dimensional vectors, while RNNs require 3 dimensional ones. The output layer has been the same through all of the experiments, with 11 different classification outcomes with a *softmax* activation function. Most of the single feature experiments were trained for 150 epochs, using *early stopping* as a regulariser to prevent

overfitting.

5.2.1 GRU vs. LSTM

There were experiments made using both GRU and LSTM classifiers, where the GRU models tended to be faster, and the LSTM models seemed to perform slightly worse across most of the experiments. Since the GRUs were much faster to train, and with seemingly no sign of loss in performance, most of the experiments were made with GRUs. The best performing system was also made up of a GRU, as can be seen in Table 5.1.

5.2.2 Early stopping

Machine learning models that train for too long will start memorising the data by learning all the small noisy features instead, losing the ability to generalise. The experiments showed clear signs of overfitting, by reporting good loss and accuracy on the training set, but poorly on the development set. To battle this, *early stopping* with patience two was applied, which purpose was to make the model stop training should the loss not improve for more than two epochs. Doing this increased the performance on the development data by 8%, as well as the overall training speed, but the model was still overfitting since the training accuracy increased with about the same amount.

5.2.3 K-fold cross validation

At the end of the experiments, k-fold cross validation was performed in order to evaluate the performance of the best performing model, since no test set was available. Performing cross validation is quite time consuming, and therefore 4-fold cross validation was executed on the model with the best score.

The best performing model had Word2Vec word embeddings of length 200, which resulted in a memory error when the k-fold cross validation was performed. Therefore the word embedding length was set to 150 instead, and the number of epochs per run was set to 60. The results were worse than the non cross classified model by about 10%, but on the other hand the results

showed that the validation score during k-fold training, and prediction of the development data, only differed with about 3%.

5.3 Essay data

In the previous 2017 shared task, the essay track was the one with the best results between essay and speech standalone models. The experiments conducted in this thesis were no exception. This section will describe the results of the essay standalone model, which has been important for the model selection for the fusion system. The most significant results from these experiments are presented in Table 5.1.

5.3.1 Preparation is key

Except for cutting the raw text to specific lengths, the only pre-processing step which differed between experiments was whether or not to preserve capital letters in words. As expected, the lowercase versions had a smaller vocabulary, and were also quicker to train and later transform the data. But, contrary to the initial theory there was no profitable results in preserving the uppercase words.

The order for cutting the sequences and training the Word2Vec also had a seemed to play a significant role in the performance of the models. When training the Word2Vec on the full sequences, and then cutting the sequences to the desired length afterwards, increased the performance by 2%. The reason might be that training the Word2Vec not only lets it learn more word dependencies, but also increases the vocabulary, which helps when later classifying new data.

5.3.2 Word n-gram comparison

When comparing the efficiency of the different word n-grams to one another, the word uni-grams won by quite the margin as can be seen in Table 5.2. The results indicate that when using systems that only use normal word uni-grams are the better option, if one decided to settle for using just one type of word n-grams. The standalone essay systems in Table 5.1 that used combinations of different word n-grams were very slow to train, and did not

N-gram comparisons							
f1	*data*	*hidden layers*	*nodes per layer*	*sg / cbow*	*embed. length*	*sequence length*	*feature*
53.62%	e,s,i	3/2/1	100/60/100	cbow	100/100	350/150	w1
46.05%	e,s,i	3/2/1	100/60/100	cbow	100/100	350/150	w2
38.07%	e,s,i	3/2/1	100/60/100	cbow	100/100	350/150	w3

Table 5.2: A comparison of different word n-grams made from the essay data. The systems tested were GRU fusion models where the speech data was normal word uni-grams, and LDA was applied to the i-vectors.

serve very useful compared to single word n-gram features.

When comparing single word n-grams, it was the word unigram feature that performed best by a large margin as shown in Table 5.2. As the n increased, the score decreased by almost 8% for each experiment. Based on these results, it was assumed that the word unigram feature was the superior choice for the final model.

5.4 Speech data

For the speech data there were two separate types to consider, the transcriptions and i-vectors, and therefore tested individually in the early phases. The most significant results are documented in Table 5.1.

5.4.1 Speech transcriptions

The speech transcription sequences are quite a bit shorter than the essay data, and was therefore trained separately from the essay data in all the experiments. For the speech transcriptions there were tests with values between 50 to 200, which is the maximum length of the training transcription data. The best performing transcription length proved to be 150 words, a bit longer than the average transcription length.

5.4.2 Abnormality preservation

Concerning preservation of abnormal occurrences in the transcription data, there were two textual features that were especially experimented on. The removal of stuttering words 'uh' and 'um', and both complete removal and replacement of foreign tokens. By removing the stuttering there was a decrease in the performance by approximately 2%, and another 2% for the removal of the foreign tokens described in Section 4.2.2. When replacing these same tokens with 0s, there was no significant increase in performance. Therefore the best performing speech system preserves both features, leaving no additional pre-processing to the speech other than transforming into lowercase.

5.4.3 Characters and words

Though the amount of n-grams that were tested were less complex than the initial experiment plan, just adding a character 6-gram to the speech standalone model increased the accuracy by about 5.5%, while also increased the training time by several minutes for each epoch. Comparing this to the results of increasing the number of n-grams in the essay standalone shows that adding different features could be more fruitful when the computational power is limited and only a few features can be chosen. The combination of character- and word n-grams did, however, not make it to the last ensemble classifier, since these results were discovered by the end of the project deadline.

5.4.4 i-vectors

Following the earlier works in NLI, especially focusing on the work of ETRI-SLP (Oh et al. [2017]) and CEMI (Ircing et al. [2017]), it was decided to keep the original vector length of 800. For extracting features the *linear discriminant analysis (LDA)* was mainly used, which has been used by for example the ETRI-SLP team among others. The i-vectors seemed to do well during the training phase, but the prediction on the development set told a different story. In an attempt to improve the i-vector performance, an *extra tree classifier* was used for feature selection, with no better performance results.

The i-vectors gave the best results when they were used without any pre-processing or feature selection, increasing the model f1 score from 55.49% to

57.67%, which resulted in the ultimate best solution among the experiments. This is illustrated in Table 5.1, where the first and second best performing models have been highlighted in grey.

5.5 Word embeddings

As described in Section 4.4.1, there were two different main approaches to performing vectorisation of the raw text data. This section will describe the results from these in the experiments, first taking a look at the frequency based tf-idf method, and then the prediction based Word2Vec results.

5.5.1 tf-idf vectorisation

By looking at Table 5.1, the tf-idf weighted embeddings performed sub par in the experiments. Both experiments applying feature selection, and not applying feature selection were carried out to measure the significance of the feature selection method. For the feature selection, a tree classifier known as the *extra trees classifier* was used since it is known for being able to estimate the importance of different features. *Recursive feature elimination (RFE)* was also used, but did not improve the performance. Feature selection using the tree classifier did speed up the training, however, it did not do much to change the outcome of the model predictions on the development data. During the training, the loss was decreasing steadily, but the results remained on par with random selection of an L1.

There were several experiments performed to locate the reason for the low performance of the tf-idf model, tuning and rewriting the model and data settings. However, since the tf-idf approach did not seem to improve through these efforts, most of the experimentation was carried out using Word2Vec embeddings.

5.5.2 Word2Vec

These embeddings were the ones that produced the best results in all of the models. And between CBOW and skip grams there were some noteworthy differences as well. The CBOW method showed the overall best results, and with the features used for these experiments there were not much of a

difference in training speed. However, when working with 1-4 word n-grams the training of the skip gram method was very slow and even resulted in memory errors.

Minimum word appearance

For the Word2Vec embedding models it is standard to only count a word or term into the vocabulary if it has appeared a minimum number of times. In an attempt to see if rare words, that even only appear a few times, would help distinguishing between the languages there was no requirement of minimum appearance in some experiments. Though not a vast difference in end performance, keeping a minimum of at least 5 occurrences did increase the prediction accuracy by approximately 0.7% in addition to speeding up the training time of the embedding model. The final model used a minimum appearance count of 5 for the essays, and 2 for the transcriptions.

Embedding length

When experimenting with different lengths of the embedded vectors, the values varied between 50 and 200. Too small vectors did not give better results, though it did make training much quicker. Pushing the length up to 200 resulted in memory errors in the early experiments when handling many sequences, especially the essay data, however, when they were successfully run they did not seem to be significantly much more useful than those of length 100. The training time of the classifier increases significantly along with the dimensionality of the input sequences, and long embeddings did not seem to make up for the time and resources they required.

Train before you cut

For the RNNs to run, they require all the vectors to be of the same length. Though this is not a problem for the tf-idf approach, the Word2Vec method does require some form of padding. During the experiments there were different sequence lengths used throughout, which meant that most of the time the sequences would be cut to a certain size before used for the classifier training. Training the Word2Vec on the full sequences, and then cutting these sequences to size before transforming them, increased the classifier accuracy with by 1.36%, without sacrificing significant time. The results are reported in Table 5.3.

Cutting sequences before or after training							
f1	*padded*	*hidden layers*	*nodes per layer*	*sg / cbow*	*embed. length*	*sequence length*	*cutting*
53.29%	GRU	3	100	cbow	100	350	before
54.65%	GRU	3	100	cbow	100	350	after

Table 5.3: The difference between cutting the sequences before or after training the Word2Vec model. Both models were trained and tested on essay *only* data, with lowercase word unigram features.

Zero padding vs. word padding							
f1	*data*	*padded*	*hidden layers*	*nodes per layer*	*sg / cbow*	*embed. length*	*sequence length*
54.52%	e,s,i	0	3/2/1	100/60/60	cbow	100/100	350/150
52.38%	e,s,i	e	3/2/1	100/60/60	cbow	100/100	350/150
55.67%	e,s,i	s	3/2/1	100/60/60	cbow	100/100	350/150

Table 5.4: Results of word padding essay and speech data, compared to zero padding. The models were trained using GRU models using single word unigrams of transcriptions and essays cut after embedding training. The i-vectors were unprocessed.

5.6 Word padding

One of the research questions was how one could potentially enrich the limited data available, which in these experiments was performed by naively adding words to the end of shorter sequences until they reach the desired length. Word padding on essay sequences of length 350, the best found length for the essays, resulted in a worse performance than the zero padded vectors by 2.14%, as seen in Table 5.4. Padding the speech transcriptions resulted in a 1.15% increase in performance.

5.7 Language performance

As can be seen by the results in Table 5.1, the best performing system consisted of essay, speech transcription and unprocessed i-vector data classified by a 3, 2 and 1 hidden layer GRU, respectively. The resulting confusion matrix in Figure 5.2 shows the performance of the final classifier for each

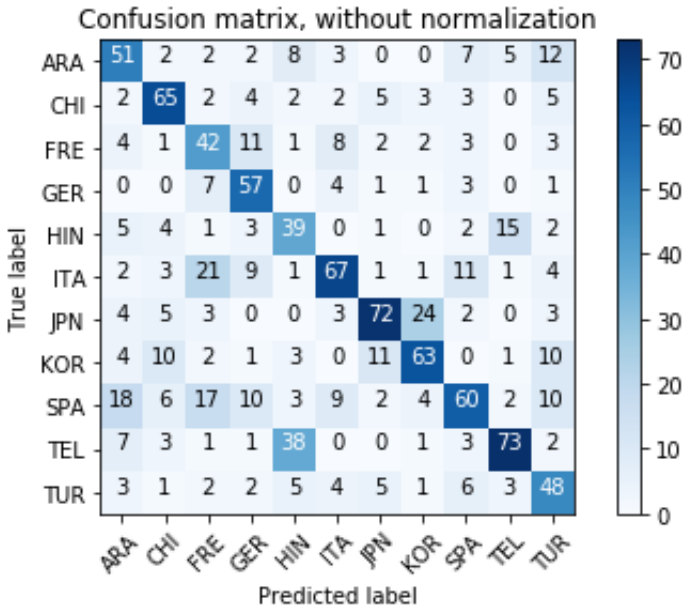

Figure 5.2: The resulting confusion matrix from the best fusion system, which is the highlighted model in Table 5.1.

language. There are 100 candidates of each represented L1, and most of them were right more than 50% of the time. Throughout the experiments there has been a repeating pattern of which languages the model had the most trouble to distinguish between.

Especially Japanese vs. Korean and Hindi vs. Telugu seem to be mistaken for each other the most. For the first group, Japanese has been classified right the highest number of times, and for the right side it is Telugu. Other languages that often were confused was the Italian vs. Arabic vs. Spanish group.

Chapter 6

Discussion

Here the experiment results will be interpreted, while discussing these results in relation to similar works, and the research questions.

6.1 LSTM vs. GRU

There seems, like stated earlier, to be very little difference between the two methods. They have been dubbed to be almost identical in accuracy, but in the experiments the GRUs generally performed significantly better than the LSTMs. This might be because during the experiments there was ultimately very little data used compared to what was planned, and that GRUs perform better with less complicated and smaller networks. Because of the short training times and comparably similar results of the GRU, it could prove to be one of the go-to methods for future deep learning with RNN in NLI and NLP. This is also supported by the research done by Yin et al. [2017], which showed that the GRUs in general performed better at most tasks. However, both models must be tested on larger NLI systems as well to be able to make a proper evaluation of the two models.

6.2 Importance of good word embeddings

Word2Vec proved to be more efficient than the tf-idf vectoriser in all the experiments, and within the Word2Vec experiments there was also differences depending on which method was used. The CBOW approach showed much greater results in the standalone experiments, and also in the fusion task.

CBOW specialise in predicting words based on the context, and based on the overall dominion over the skip grams, the most useful word embeddings for distinguishing different languages might lie in the sentence context rather than specific, rare words.

Looking at the performance of the implementation of Word2Vec vs. tf-idf shows how important it is to use a good word embedding method. Not only is it important to choose an appropriate embedding method, it also proved to be important to consider the data which they are trained on. You do not speak like you write, and neither do you write like you speak, which was also shown when attempting to predict speech data with an essay classifier. Therefore embedding the essays and transcripts separately might be the best way to go, considering that the these text data types are quite different in nature. Among the related works, both the Ionescu and Popescu [2017], and Ircing et al. [2017] teams reported that essay and transcriptions data require different types of n-grams, and that ultimately the speech transcriptions were better left out. Treating the essays and transcriptions as data types of different natures with different features could increase the usability of the speech transcriptions, but this is a task left for future works.

Tf-idf were the chosen word embedding method for several of the 2017 NLI shared task, including Ircing et al. [2017] and Oh et al. [2017], that were the second and third best fusion models respectively. The success of tf-idf in the related tasks and fields means that the tf-idf method is viable. Therefore the cause for the failure of tf-idf for this thesis might be related with particularities of the implementation. On the other hand, the tf-idf vectors could also be less suited for RNN models than Word2Vec or other similar prediction based methods.

6.3 Transcriptions not informative enough

As seen in the previous chapter, there is a large dissonance between the training results and the results from development data predictions. This might be caused by factors such as very short sequences as mentioned by other works [Malmasi et al., 2017], but since essays at the same length as speech transcripts gave better results it also indicates that the data itself might be a bit tricky. There were some abnormal occurrences found in the training

data, which might improve the transcript data performance if fixed.

On the other hand, as all neural networks are known for being constantly hungry for more data, it could be the model choice or the chosen hyper-parameters that are at fault for the poor results. But the gap between the essay track in the shared task, and the inferiority compared to the i-vectors in the related works speak otherwise.

6.4 Naïve word padding works

By applying a naïve word padding method to the speech transcriptions, the model performance improved by 1.15%, which means that enriching the transcription data is possible. It is uncertain why the difference between padding the transcriptions and essays would be as much as 3.29%, considering that the transcriptions on average would be padded more than the essays. As stated in Section 2.8.1, the average length of a transcript is 90 words long, while the essays are on average 315 words long.

When word padding the transcripts to a length of 150, it means an average of 60 words are padded per sequence. For the essays this would be 35 words in order to reach a length of 350. Yet, even though the transcripts were padded with more redundant information than the essays, the end result was significantly better. This could mean that the transcriptions would respond better to word specific features, and that the essays are more suited for context features. It has been reported by several teams in the 2017 NLI shared task that there are concerns regarding language bias when it comes to the prompt topics. Because of the environment the data was captured, many participants mention key words regarding their country of origin when answering the questions. Repetition of these in a naïve word padding experiment like performed in this thesis could have enforced these features on the short sequences, and therefore made that model perform better. For future works, it seems like padding of sequences can artificially enrich the data, and experimentation with more sophisticated word padding methods could be an interesting approach.

6.5 Sequence length is important

During the standalone essay and transcription experiments, the length of the sequences was a central parameter regarding the overall results of the RNN models. Using the full length of the sequences ended with less good results than shorter sequences. The probable cause of this is that even though there are essays with around 600-800 words, the majority of all essays lay between 300-400 words. Meaning that if the sequence length is longer than 500, around 90% of the essays have to be padded in some manner. And among these 90%, approximately 50% of these are padded to the double of their length or more.

Because of the extensive padding, the model ended up underfitting on the data, probably due to the sequences being mostly zeros. Word padding the full length sequences ended in horrible overfitting, which most probably was caused by the large amounts of redundant information. Therefore, unless some more sophisticated word padding method can be utilised, the sequence lengths should be kept around the average length across the data set.

6.6 Fusion

Here the results of the attempt at fusing the different data will be discussed. This track was the most successful at the 2017 NLI shared task, and was also deemed at the beginning of this research to be the one to produce the most fruitful results.

6.6.1 Essay + Transcript + i-vector

Even though the system in the end only used word unigram features for classification, the performance was better than the standalone models. Compared to the 2017 shared task, however, there was a strange development connected to the i-vectors. According to the earlier tasks, it seems to be the most popular conclusion that the i-vectors are what brought the speech systems to new levels. Yet when used in the systems produced through the experiments, they did not give any significant improvement to the results.

The best performing model among the experiments did not perform any preprocessing or feature selection on the i-vectors. In the related works using i-vectors, there were no systems that utilised only RNNs to perform classification on this data, so it is difficult to locate the exact reason for the low performance. Though the error might be in poor handling of the i-vector data itself causing poor results, it could indicate that RNNs are not well suited for this data type. Therefore for RNNs used in future work, it could prove fruitful to classify the i-vector data using a different classifier in an ensemble system.

Only lowercase word unigram features, with zero padded Word2Vec cbow embedding models trained before cutting the sequence lengths were used for the final best performing model. But, the optimal last experiment would have been to use word padded speech data on top of the current best performing model, however, that will be something that has to be left for the future.

6.6.2 Overfitting

In almost all of the models there has been a problem with overfitting. This is most probably because of not feeding the model enough data, however, just having a lot of data does not do the trick either. When attempting to increase the amount of data, by appending the remainder of sliced sequences to the data, the training took a lot of time while producing no better results than the best single word unigram system. The model did overfit less, but the growth stagnated at the same accuracy as the simple word unigram system. Because of this observation, it seems it would be more fruitful to include several features of short sequence lengths, rather than just increasing the amount of data.

6.7 Language confusion

The confusion matrix that resulted from the best fusion model is very similar to those of related works. Many have speculated that the difficulty of distinguishing Hindi and Telugu is caused by the fact that they have received the same English education through the Indian school system. Korean was often misinterpreted as Japanese, but not so much the other way around. Confu-

sion between these two is also not uncommon across related works, and has been credited to the fact that the languages are quite close to one another.

It is interesting that Japanese scored the best among all the L1s, considering that it is the language with the lowest average length of both speech transcriptions and essays as described in Section 2.8.1. Italian is the language with the third lowest transcription and essay average length, but also ranked as the third best classified L1. This might indicate that these languages have some very prominent features, given the results showing that they can classify quite well based on a below average amount of data.

6.8 Comparison to the related works

By taking a look at Table 6.1, it is clear that the best performing model from this thesis has much lower performance than the related works presented in Chapter 3. Even the baseline, that was trained by using a linear SVM on word unigrams, had achieved much better results. One of the first major differences between the thesis model and the related works would be that, except for the baseline, they all used several features. For the thesis model, only word unigrams and raw i-vectors were used, but the related works presented in this thesis all used several features. The thesis model did not use any sophisticated features, and neither were there enough tests evaluating the efficiency of diverse features, which is probably one of the main reasons why the results were not particularly good.

Combining several different classifiers in ensemble systems also seem to be a solid approach. The L2F team also used GRUs, combining them with feed-forward layers, and achieved much better results. Further, shallow networks seem to be the overall best performing on the data and among the top systems. Oh et al. [2017] and Kepler et al. [2017] used more than three hidden layers for their models, but they used different layer types than the ones in this thesis. Most of the experiments showed little improvement of using deep models with three hidden layers compared to just using one or two. But, since the experiments used relatively little data, this should be explored in a deep GRU network with more input data.

The related works were ultimately tested on the test set provided by the

Comparison to the related works essay + transcript + i-vector			
team	*acc.*	*f1*	*approach*
Unibuckernel	93.18%	93.19%	Kernel-based learning
CEMI	92.55%	92.57%	Feed-forward NN
ETRI-SLP	92.18%	92.20%	Vanilla DNN with early fusion
L2F	83.91%	83.77%	Feed-forward + GRU
Baseline e,s,i	79.09%	79.01%	Linear SVM on word unigrams + i-vectors
Thesis solution	57.90%	57.67%	GRU
Baseline random	9.10%	9.10%	Randomly selects a L1

Table 6.1: A comparison of the experiments' best system and the related works. The baselines are taken from Malmasi et al. [2017], which was used to evaluate the team solutions in the 2017 NLI shared task.

2017 NLI shared task, but where unfortunately not available for this thesis. Therefore the results of this thesis cannot be directly compared to the related works. But, since the cross validation showed that there were similar behaviour in the validation data and the development data, the results should still be relevant for highlighting some of the important features to keep in mind when working with RNNs in NLI.

Chapter 7

Conclusion and Future Work

To wrap it all up, this chapter will present the conclusion and what should be considered for future works.

7.1 Conclusion

The experiments conducted in this thesis were not on par with the previous NLI research, especially with regards to the previous shared task. Compared to the baselines described in Malmasi et al. [2017], neither the LSTM nor the GRU models reached these goals. However, there have been discovered some methods that might prove useful to increasing the performance of future deep learning approaches in NLI, and some methods that might be better left out.

It seems that prediction based word embedding methods overall perform better than frequency based ones. And among the prediction based word embedding approaches in Word2Vec, the choice between CBOW and skip grams are also important to take into consideration when building a deep learning RNN model.

The models benefited from going from shallow networks of 1 or 2 hidden layers, also showing good results at 3. Going past 3 layers did not contribute to noticeably better results for any of the data, just causing longer training times. An environment that uses more variables and more data is more probable to benefit from deep neural networks compared to the environment of this thesis, and perhaps even combinations of several neural network models

could be a good approach.

The relatively good results of the fusion model on only word unigrams could prove that if future works could input more data into a similar model, the results would improve. Having considered the related work and the experiments conducted in this thesis, it does seem that there is a bright future for deep neural networks in NLI.

7.2 Future work

Based on the results from the experiments conducted in this thesis, and knowledge from related works, the problems that are left for future works will be presented.

7.2.1 Speech transcription re-evaluation

Even though the speech transcriptions in theory could give valuable information, especially in the form of semantic features, such as word placement and choice of vocabulary, they have been hard to use effectively. Both from conducting the experiments, reading related papers, and inspecting the dataset, it is clear that transcriptions does not perform well compared to the essay text data. Since the texts are short, they seem especially ill fitted for deep learning approaches, and it also seems that lack of useful content might be an underlying serious issue. Padding the transcription sequences with words showed some improvement in performance, meaning that this is a topic worth considering further with more sophisticated word padding in future works.

Stammering and repetition of words sometimes make up a lot of the speech transcriptions, and might be a sign of stress due to the examination setting, instead of an actual L1 bias. For future work it could be a good idea to either find ways to use this stammering for something useful, or find a way to extract the L1 features by pre-processing them so that they can become more useful. Concatenation, as also mentioned by Malmasi et al. [2017], will probably also be an important addition to the transcriptions since they are very short, especially if stammering, or the word repetitions, would be removed.

7.2.2 Other classifier models

based on the sub-par results produced by the RNNs in the experiments, it might be that that other deep neural network models might fare better. In other related works the i-vector data proved to be very useful, which makes it strange that they would be so little useful in this thesis if the system was optimal. Experimenting with convolutional neural networks, or other deep neural networks, while also taking into consideration the differences between speech and essay handling could be an interesting task for future work.

7.3 Encountered problems

One of the biggest, and most expensive, problems faced was the grave error in time estimation. Planning the finish of the experiments by the middle of March, when in reality the basic environment itself was not in place before that very self set deadline.

Following that, another problem which deep learning is famous for, RNNs require a lot of computational power. The RNN models cost much more at earlier stages than expected in the planning process. Such as memory constraints, and several hours required for one training run. This was overcome by time as the implementation became more efficient, but the optimisation was too late for all the planned experiments to be executed in time. Training with more than one n-gram unit per data type proved to take a great toll on the computer, with using up to four features increasing the training time to a whole day.

7.4 Learning outcome

As the peak of my education so far, it has gone past my expectations in many ways. Through the last year I have achieved things that I had never dreamt of at the start of my masters, and I have tasted some of the hardships and perks of doing a one man academic research project for a year.

I did not have any experience with native language identification or implementing deep learning prior to choosing this specific thesis. But, what is the

point of a masters thesis if not to challenge oneself and one's abilities? After this year, I am eager to continue working with something similar. Even if not as a job, I have already started to plan my next hobby project on language identification.

The scope of the project became maybe a little broader than what would be possible for me in that time frame, which was much due to panic and sudden impulses on something else that could be interesting to add or examine. This has made me very aware of how important it is to stick to one's schedule, not to underestimate time constraints, and *always* make sure to keep frequent backup of your work.

Bibliography

I. Anaconda. Anaconda Documentation. `https://docs.anaconda.com/`, 2012.

T. Bayes. An Essay Towards Solving a Problem in the Doctrine of Chances. *Philosophical Transactions*, pages 370–418, 1763. doi: 10.1098/rstl.1763.0053. Accessed on 8. December 2017.

D. Blanchard, J. Tetrault, D. Higgins, A. Cahill, and M. Chodorow. TOEFL11: A Corpus of Non-Native English. 2013. Retrieved May 7th, 2018.

D. Blanchard, J. Tetrault, D. Higgins, A. Cahill, and M. Chodorow. ETS Corpus of Non-Native Written English. `https://catalog.ldc.upenn.edu/LDC2014T06`, 2014. Retrieved December 19th, 2017.

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*, pages 1–15, 2014.

F. Chollet. Keras. `https://keras.io`, 2015.

A. Cimino and F. Dell'Orletta. Stacked Sentence-Document Classifier Approach for Improving Native Language Identification. In *Proceedings of the 12th Workshop on Building Educational Applications Using NLP*, pages 430–437, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

G. Deco and D. Obradovic. Preliminaries of Information Theory and Neural Networks. *An Information-Theoretic Approach to Neural Computing*, pages 7–27, 1996.

I. Goodfellow. *Deep Learning*, chapter Introduction, pages 1–12. Adaptive computation and machine learning. MIT Press, 2016.

Google. TensorFlow. `https://www.tensorflow.org/`, 2015. Last visited March 12th, 2018.

S. Gupta. Automated Text Classification Using Machine Learning. `https://towardsdatascience.com/automated-text-classification-using-machine-learning-3df4f4f9570b`, 2018. Retrieved April 22nd, 2018.

S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780, 1997.

R. T. Ionescu and M. Popescu. Can string kernels pass the test of time in Native Language Identification? In *Proceedings of the 12th Workshop on Building Educational Applications Using NLP*, pages 224–234, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

P. Ircing, J. Švec, Z. Zajíc, B. Hladká, and M. Holub. Combining Textual and Speech Features in the NLI Task Using State-of-the-Art Machine Learning Techniques. In *Proceedings of the 12th Workshop on Building Educational Applications Using NLP*, pages 198–209, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

F. Kepler, R. F. Astudillo, and A. Abad. Fusion of Simple Models for Native Language Identification. In *Proceedings of the 12th Workshop on Building Educational Applications Using NLP*, pages 423–429, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

S. Malmasi and M. Dras. Arabic Native Language Identification. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Langauge Processing (ANLP)*, pages 180–186, Doha, Qatar, October 2014a. Association for Computational Linguistics.

S. Malmasi and M. Dras. Chinese Native Language Identification. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 95–99, Gothenburg, Sweden, April 2014b. Association for Computational Linguistics.

S. Malmasi and M. Dras. Finnish Native Language Identification. In *Proceedings of Australasian Language Technology Association Workshop*, pages 139–144, Melbourne, Australia, November 2014c.

S. Malmasi and M. Dras. Native language identification using stacked generalization. *arXiv preprint arXiv:1703.06541*, pages 1–33, 2017.

S. Malmasi, M. Dras, and I. Temnikova. Norwegian Native Language Identification. In *Proceedings of Recent Advances in Natural Language Processing*, pages 404–412, Hissar, Bulgaria, September 2015.

S. Malmasi, K. Evanini, A. Cahill, J. Tetrault, R. Pugh, C. Hamill, D. Napolitano, and Y. Qian. A Report on the 2017 Native Language Identification Shared Task. In *Proceedings of the 12th Workshop on Building Educational Applications Using NLP*, pages 62–75, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

T. Mikolov. word2vec. `https://code.google.com/archive/p/word2vec/`, 2013. Last visited April 23rd, 2018.

A. Ng. What data scientists should know about deep learning. `https://www.slideshare.net/ExtractConf/andrew-ng-chief-scientist-at-baidu`, 2015. Retrieved April 11th, 2018.

T. Odlin. Crosslinguistic Influence in Second Language Acquisition. *The Encyclopedia of Applied Linguistics*, pages 1–6, 2013.

Y. R. Oh, H.-B. Jeon, H. J. Song, Y.-K. Lee, J.-G. Park, and Y.-K. Lee. A deep-learning based native-language classification by using a latent semantic analysis for the NLI Shared Task 2017. In *Proceedings of the 12th Workshop on Building Educational Applications Using NLP*, pages 413–422, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

P. Raybaut. Spyder Documentation. `https://pythonhosted.org/spyder/#`, 2009.

S. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach, Global Edition*, chapter The History of Artificial Intelligence, pages 16–28. Pearson Education Limited, 2016.

J. Schneider. Cross Validation. https://www.cs.cmu.edu/~schneide/tut5/node42.html, 1997.

B. Schuller, S. Steidl, A. Batliner, J. Hirschberg, J. K. Burgoon, A. Baird, A. Elkins, Y. Zhang, E. Coutinho, and K. Evanini. The INTERSPEECH 2016 Computational Paralinguistics Challenge: Deception, Sincerity & Native Language. In *Interspeech 2016*, pages 2001–2005, San Francisco, USA, September 2016.

J. Tetrault, D. Blanchard, and A. Cahill. A Report on the First Native Language Identification Shared Task. In *Proceedings of the Eight Workshop Innovative Use of NLP for Building Educational Applications*, pages 48–57, Atlanta, USA, June 2013.

L. M. Tomokiyo and R. Jones. You're Not From 'Round Here, Are You?: Naïve Bayes Detection of Non-Native Utterance Text. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8, Pittsburgh, USA, June 2001. Association for Computational Linguistics.

C. Trim. The Art of Tokenization. https://www.ibm.com/developerworks/community/blogs/nlp/entry/tokenization?lang=en, January 2013. Retrieved March 9th, 2018.

A. Trivedi. Transfer Learning and Fine-tuning Deep Neural Networks. https://www.slideshare.net/PyData/py-datasf, 2016. Retrieved April 11th, 2018.

S. Vijayarani, J. Ilamathi, and M. Nithya. Preprocessing Techniques for Text Mining - An Overview. *International Journal of Computer Science & Communication Networks*, 5(1):7–16, 2015.

W. Yin, K. Kann, M. Yu, and H. Schütze. Comparative Study of CNN and RNN for Natural Language Processing. *arXiv:1702.01923*, pages 1–7, 2017.

G. Zoubin. Bayesian Machine Learning. http://mlg.eng.cam.ac.uk/zoubin/bayesian.html, 2004. Retrieved December 12th, 2017.