A Study of Recurrent and Convolutional Neural Networks in the Native Language Identification Task

Thesis by

Robert Werfelmann

In Partial Fulfillment of the Requirements

For the Degree of

Masters of Science

King Abdullah University of Science and Technology

Thuwal, Kingdom of Saudi Arabia

April, 2018

EXAMINATION COMMITTEE PAGE

The thesis of Robert Werfelmann is approved by the examination committee

Committee Chairperson: Xiangliang Zhang
Committee Members: Mikhail Moshkov, Xin Gao

ABSTRACT

A Study of Recurrent and Convolutional Neural Networks in the Native Language Identification Task

Robert Werfelmann

Native Language Identification (NLI) is the task of predicting the native language of an author from their text written in a second language. The idea is to find writing habits that transfer from an author's native language to their second language. Many approaches to this task have been studied, from simple word frequency analysis, to analyzing grammatical and spelling mistakes to find patterns and traits that are common between different authors of the same native language. This can be a very complex task, depending on the native language and the proficiency of the author's second language. The most common approach that has seen very good results is based on the usage of n-gram features of words and characters.

In this thesis, we attempt to extract lexical, grammatical, and semantic features from the sentences of non-native English essays using neural networks. The training and testing data was obtained from a large corpus of publicly available essays written by authors of several countries around the world. The neural network models consisted of Long Short-Term Memory and Convolutional networks using the sentences of each document as the input. Additional statistical features were generated from the text to complement the predictions of the neural networks, which were then used as feature inputs to a Support Vector Machine, making the final prediction.

Results show that Long Short-Term Memory neural network can improve performance over a naive bag of words approach, but with a much smaller feature set. With more fine-tuning of neural network hyperparamters, these results will likely improve significantly.

ACKNOWLEDGEMENTS

I would like to give a big thanks to my friend and colleague, Uchenna Akujuobi. Without his help, I would never have been able to get as far as I have in such a short amount of time. In addition to Uchenna, I would like to include Michal Mankowski for making my stay at KAUST such a wonderful experience filled with pleasant memories. I want to thank my fiancee Maha, for giving me so much support and strength to keep going, especially on days with discouragement. I would like to thank King Abdullah University of Science and Technology for providing me with the opportunity to learn and improve my skills. As well, I'd like to thank the Computer Science department, whose professors have given me the knowledge and skills I can use to make a positive impact on the world. Last, but not least, I would like to give a tremendous thanks to my professor, Xiangliang Zhang, for giving me this opportunity to study under her wing, giving me guidance, and providing me with the inspiration for this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

Since the introduction of the word wide web, internet usages across the globe have been increasing at a tremendous rate each year. With this increasing usage, more and more data is being allocated, usually in the form of usage statistics (e.g. search and shopping trends), communications, and user-based knowledge (e.g. Wikipedia). Information coming from people is not always in the form of numbers. People speak different languages, and have different understanding of what different words mean. But the world is still advancing toward a completely global community where any two people from different backgrounds and cultures can communicate. This is all due to the technology of being able to translate messages, filter and define words, and assist in searching for the right information.

1.1 Natural Language Processing

One may wonder how humans are able to interact and communicate with each other. We can communicate through forms such as speech, writing, and sign language. Often, the way we speak, or the way we write also communicate meaning and attitude. If we are to be able to communicate to a machine, we'll need to understand how to interpret words, how they are written, and how they are presented. Natural Language Processing (NLP) is the field that attempts to understand this. It is the field of study for interactions between a human and a machine. The field originated from the idea of the Turing Test, which Alan Turing described as the "Imitation Game"

[1]. In his original paper, Turing considered the question “Can machines think?” But to define *machine* and *think* is very complex, so instead he proposes the question “Can a machine convince a human that it is not a machine?” In order for any machine to convince a human that it is a human and not a machine, it has to be able to communicate like a human. This is where strong applications of NLP are crucial. The machine has to be able understand what a human is communicating, and more importantly, how to respond.

NLP has many applications. For example, an NLP system can analyze text to identify mentions of people, places, and organizations. This is also called Named-Entity Recognition [2]. Other examples of NLP are stemming and tokenizing of text. A document can be converted so that words are simplified to their root word, such as changing instances of “running”, “runs”, and “ran” to simply “run”. Tokenizing simply breaks down a document into tokens that make up words, characters, or other user-defined boundaries.

1.2 Native Language Identification

Native Language Identification (NLI) is the task of classifying the native language of an author whose text is written in a second language. For example, if an author whose first language is Spanish, and has written a text in English, then a good NLI system will be able to predict this texts authors native language to be Spanish. This task extends the task of author profiling, which includes other tasks such as determining an authors gender, location, and age. NLI has a few, but impactful applications. It can be used to study typical habits of certain second languages that can be used to develop teaching material [3], or to help further develop grammar correction tools [4].

NLI is a relatively new task that began in earnest in 2005 with Koppel’s [5] analysis of errors and stylistic idiosyncrasies. NLI is a task that is filled with complexity, as there are many factors to consider when making a prediction.

Several approaches have been developed in the NLI task, such as using spelling and grammar errors as features for a classifier. Other approaches include using various n-grams of words and characters within the text. As of this writing, only few publications have approached this task using neural networks, so more analysis can be done.

1.3 Problem Setting

The problem can be described as follows: Given a set of documents $D_1 ... D_N$ written in some language L2, and without any knowledge of the author's origin or background, we want to predict the native language L1 of the authors for each document $D_1 ... D_N$. This involves extracting features and patterns from the text which will give clues to what the native language of the author might be. For this thesis, we have chosen to extract an ensemble of different features, with three being distinct:

• Lexical features of the text. This involves understanding how someone writes at the character level. For example, the word *color* can also be written as *colour* and still hold the same meaning. Another example is where some authors spell possessives such as *Charles'* instead of *Charles's*, and vice versa.

• Grammatical structures. There might be grammatical structures that are common to authors of a certain L1. For example, one L1 might write a determiner followed by a noun, and another L1 might commonly omit the determiner before nouns.

• Semantic structures. The content of the writing itself may also hold clues to the author. One simple example would be a sentence saying *"Hello, my name is Diego and I'm from Spain"* might indicate that the author is from Spain.

• Additional statistical features. Other features might help separate L2 writings based on statistical occurrences such as usages of punctuation, sentence lengths, specific spelling errors, and other idiosyncrasies.

Considering these features, we have developed models using two neural network approaches. One is using a bi-directional Long Short-Term Memory (LSTM) model, and another using a Convolutional Neural Network (CNN) model. The following objectives were achieved in this thesis:

- Design a method for extracting statistical features from raw text.
- Build an LSTM model that learns lexical, grammatical, semantic structures, and combinations thereof from text.
- Build a CNN model that learn the same structures.
- Compare various configurations of both LSTM and CNN models.

This Thesis is structured in the following way: Chapter 2 will discuss prior work that has been developed in the area of Native Language Identification. Chapter 3 will describe and analyze the dataset used in the experiments. Chapter 4 will describe the methodology of this Thesis and the approach to the problem. Each algorithm used will be discussed in detail. Chapter 5 will report the results from this Thesis' approach and models, showing a comparison of the complete ensemble and subsets of the feature set. In Chapter 6, conclusive statements and potential future work will be mentioned.

Chapter 2

Previous Work

Native Language Identification has been an area of research since 2005. Since that time, only a handful of research papers had been published, as the bar had been set fairly high early on. However, when the 2013 Native Language Identification Shared Task began, much more research and approaches had been developed. This chapter will describe some of the most effective approaches as well as an approach that is similar to this Thesis' work.

2.1 Native Language Identification

Among the first known attempts to apply Native Language Identification was Koppel et al. [5] in 2005. Their work showed that stylistic idiosyncrasies could be used to identify the native language of an author from text written in another language. Three feature categories were used: function words, Letter n-grams, and errors and idiosyncrasies. Function words are words that hold little meaning by themselves, but rather help structure grammar. The example they use is the word *the*, which is used less frequently in some native languages such as Russian. Koppel et al. explains letter n-grams as useful for possibly showing orthographic conventions used in some native languages. Errors and idiosyncrasy features were extracted as it was expected that many authors would transfer orthographic or syntactic conventions from their native language to their second language. These were broken down into four categories: orthography errors, syntax errors, neologisms, and part-of-speech bi-grams. Koppel et al. considers orthography errors to include instances of repeated letters, missing letters, conflated words and other spelling errors. Syntax errors include sentence fragmentations, run-on sentences, missing words, and misuse of singular and plural. Neologisms are new words or expressions that have not seen common usage. Koppel et al. states that some writers create neologisitc adjectives, such as *fantabulous*, while others create verbs and nouns. The part of speech bi-grams considered are ones that occur rarely in text. The intuition behind this is that certain part-of-speech pairs, even if they are in error, might occur more commonly among a specific native language than others. In their experiments, they used the International Corpus of Learner English V1. From the corpus, they use a subset consisting of essays from 258 authors from five languages, with each essay containing between 579-846 words. Using the features extracted, they use a linear Support Vector Machine over a ten-fold cross-validation experiment. From their experiments, they achieved an accuracy score of 80.2%.

By 2013, interest in Native Language Identification had grown enough to start a shared task on the topic. In the First Native Language Identification Shared Task [6], 29 teams participated to improve accuracy results on the then newly-created TOEFL11 dataset [7]. Accuracy scores as high as 84.6% were obtained. The report does not detail the approaches each team used, but does state that the most used classifier was Support Vector Machines. Most teams used word, character and part-of-speech n-grams as features.

In 2017, the shared task was hosted again [8]. This shared task reports NLI approaches and results of three tracks, essay-only, speech-only, and a combination of the two. The dataset used here was the TOEFL11 [7]. 19 teams participated in the task, with 17 competing in the essay-only track. Most teams used Support Vector Machine classifiers with various n-gram features. Here, we'll describe two notable team's approaches. One that used neural networks, and one that achieved the highest accuracy and F1.

Cimino [9] used a Stacked Sentence-Document Classifier approach. Their model consists of two SVM classifiers, one for classifying the native language of each sentence in a document, and the other for classifying the entire document. At the sentence level, they use several n-gram features of characters, words, lemmas, and dependencies to make predictions of the sentence. At the document classification, they use the results of the sentence predictions plus a few additional features such as average sentence length. The best configuration scored an F1 of 88.18% on the TOEFL11 dataset, which was the highest of the 2017 NLI Shared Task.

2.2 Neural Networks

Neural networks have recently been a popular choice with NLP tasks. Collobert [10] attempted to address several NLP tasks using a single neural network architecture. They considered six NLP tasks: Part-Of-Speech tagging, Chunking, Named-Entity Recognition, Semantic Role Labeling, Language Models, and Semantically Related Words. The common approach for these tasks is to extract features that are given to a linear Support Vector Machine. These features often task dependant, with Semantically Related Words requiring a complex feature set. Collobert presents a deep neural network broken into several steps. Each word from the input is embedding into a d-dimensional vector space using a lookup table. From the lookup table, a Time-Delay Neural Network (TDNN) is used to perform a convolution on the input sequence. These layers can be stacked, extracting local features in lower layers, and extracting more global features in subsequent layers. In order to solve complex tasks like Semantically Related Words, more classical neural network layers can be added. Collobert states that by sharing the features learned in this deep neural network architecture, the tasks mentioned can be solved. In their experiments, they used sections of the PropBank dataset for training and testing, as well as a database of 631 million words from Wikipedia. They tested their architecture on solving a combination of tasks, all inclusive of Semantically Related Words. They reported word error rates between 14.30 and 18.40 across different lookup table dimension sizes of 15, 50 and 100.

In a more specific study, Radford used LSTMs to perform sentiment analysis on Amazon product reviews [11]. Sentiment analysis is the task of understanding subjective information, typically from text. An example of this would be predicting the rating a user would give a product on amazon based on the review text. In Radford's papers, the Amazon product review dataset was used for training and testing. The dataset contains over 82 million product reviews, and Radford split the data into 1000 shards of equal size, with 2 shards set aside for validation and testing. Their model chosen was a multiplicative LSTM that uses 4,096 units for reason of converging faster than a traditional LSTM. Their model was trained for one epoch on batches of 128 which contained lengths of 256. An Adam optimizer was used with an initial learning rate of 5e-4 that decayed linearly to zero. Each review was inputted into the LSTM as sequences of UTF-8 encoded bytes. After training, Radford tested their model on four datasets; movie reviews from Rotten Tomatoes, a different but overlapping dataset from Amazon product reviews, a dataset from subjectivitiy and objectivity detection, and MPQA's opinion polarity. Radford's model achieved the highest accuracies with the movie and product reviews datasets (86.9 and 91.4 vs Adasent's 83.1 and 86.3). The product reviews comparison is likely biased however, as even Radford mentions that the dataset overlaps with what the model was trained with.

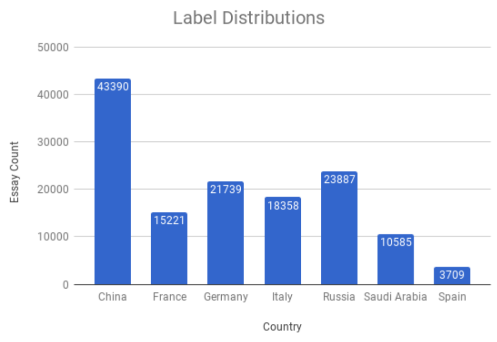
Bjerva et al. [12] used neural networks to build features for their SVM classifier. The main component of their system was using deep residual networks, also known as resnets. They applied resnets with variations of word unigrams, and 4-6 character n-grams. Another feature uses bidirectional LSTMs on tokens combined with their corresponding part-of-speech tag. They included bigrams of spelling errors, ranging from one to three character sequences in a word where the spelling error occured. Lastly, they include features from a continuous bag-of-words that represents each essay as the average embedding of all its words. For the systems they reported results, all included their resnet features. Their best system consisted of a resnet with word unigrams and character 5-grams, a separate resnet with character 6-grams, a multi-layer perceptron, and a continuous bag-of-words. This system obtained an F1 score of 83.23% on the TOFEL11 dataset.

Using various combinations of n-gram features of words of characters and Support Vector Machines has become very popular since NLI's introduction. But occasionally, different approaches are applied. Bjerva et al. [12] uses an LSTM for capturing syntactic patterns, but is only used to complement the resnet configurations. It is not clear what patterns or features are extracted from the resnets used in [12]. In this thesis, we will primarily use LSTMs for capturing lexical and grammatical features.

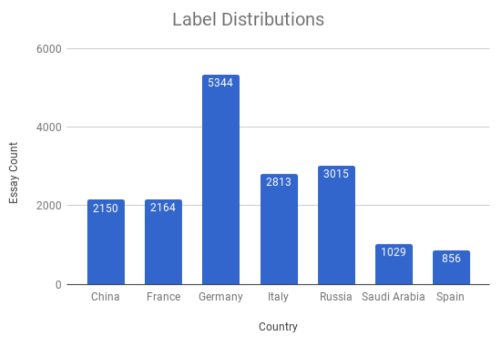
Chapter 3

Data Analysis

The dataset used here consists of short essays written in English from the EF-Cambridge Open Language Database (EFCAMDAT) corpus. From the official website [13], *The EF-Cambridge Open Language Database (EFCAMDAT) is a publicly available resource to facilitate second language research and teaching. It contains written samples from thousands of adult learners of English as a second language, world wide. EFCAMDAT currently contains over 83 million words from 1 million assignments written by 174,000 learners, across a wide range of levels (CEFR stages A1-C2).* The corpus contains 1,180,309 essays from 198 nationalities across 16 different levels of teaching units. Many of the nationalities are comprised of only a handful of authors, and some countries have more writers in lower or higher levels. To get a sizable sample of several countries with clear native languages, the following countries were selected: China, France, Germany, Italy, Russia, Saudi Arabia, and Spain, which are assumed to have authors whose native language is Chinese, French, German, Italian, Russian, Arabic, and Spanish, respectively. The corpus was divided into two sub samples, each with the aforementioned groups; essays which are part of the unit levels 5 through 10, and unit levels 11 through 16. It is expected that an NLI model will perform better on lower levels than higher levels. Higher levels indicate more experience in writing English, and would be more difficult to find patterns that transfer from their native language to English. Conversely, lower levels indicate less experience in writing English, and are more likely to contain grammar and spelling errors, including writing structures carried over from their native language.

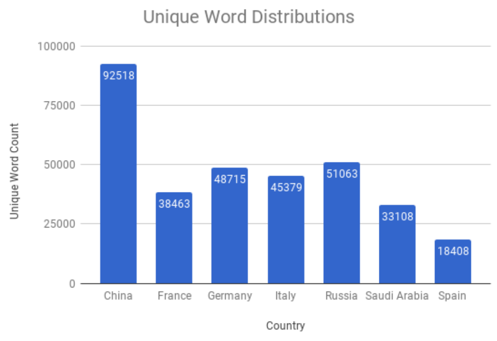


(a) Units 5-10

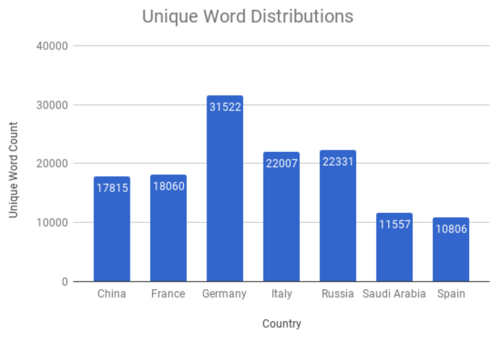


(b) Units 11-16

Figure 3.1: Label distributions



(a) Units 5-10



(b) Units 11-16

Figure 3.2: Unique word distributions

Figures 3.1(a) and 3.1(b) show the number of essays for each label. Group 1 (units 5-10) has 136,889 essays, with most having authors from China. Group 2 (units 11-16) has 17,371 essays, most having German authors. In both groups, Spain has the fewest authors. Figures 3.2(a) and 3.2(b) show the number of unique words in each label. Both groups have the same proportions of essay counts to unique word counts. For instance, the label with the highest number of essays also has the highest number of unique words. We can also see that although the total number of unique words in the higher level units is fewer than in the lower level units, the ratio of unique words to number of essays is much higher. For example, the unique words to number of essays ratio for Germany in the higher level units is about 5.9, compared to China's 2.1 in the lower level units. This supports the expectation that higher level units authors make use of a larger vocabulary.

Chapter 4

Methodology

In this chapter, we will discuss the approaches used in solving the Native Language Identification problem.

4.1 Problem Definition

Native Language Identification can be defined as follows: Given a list of documents, $D_1, D_2, ..., D_N$ where N is the number of documents, each written in language L_y, and where each document has a principle author whose native language is L_x, predict L_x for each document. For example, D_1 is written by an author who's L_x is Spanish, and L_y is English. The NLI task is to predict that D_1, written in English, is written by an author who's native language L_x is Spanish.

4.2 Supervised Learning

Here, we consider this problem to be a classification problem. In order to make predictions, each document D_i must have a label L_j which represents the native language of the principle author. In order to learn how to make accurate predictions, features must be extracted from each document and related to the corresponding L_j language. We use a Long Short-Term Memory (LSTM) model, a type of Recurrent Neural Network (RNN), to extract lexical, grammatical, and semantic features from the text. We also use a Convolutional Neural Network (CNN) as an alternative approach to using LSTMs. Using the prediction results of the neural network in

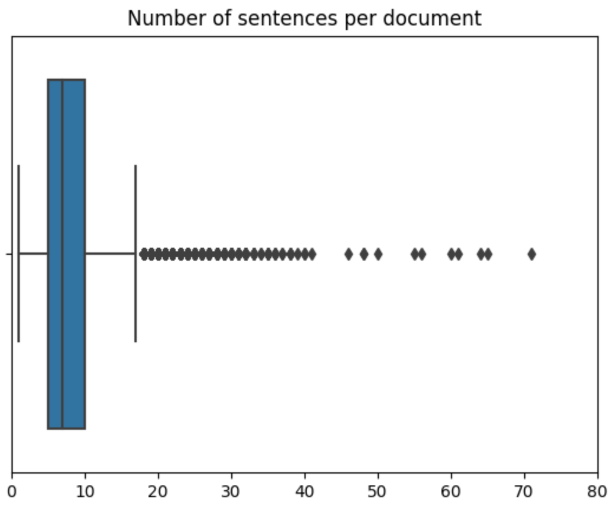


Figure 4.1: Box plot of sentence count per document

combination with additional statistical features, we will use a linear Support Vector Machine (SVM) to make final predictions, as SVMs tend to be effective in these types of classifications. A more detailed description of RNN, LSTM, CNN, and SVM can be found in chapters 4.4, 4.5, 4.6, and 4.7, respectively.

4.3 Features and Methods

Finding the best features for any classification task is a very difficult part. Choosing the wrong feature, even if it is one among many, can lead to poor performance. Here, we will describe the features used in this thesis. They can be broken into two categories, those learned by deep learning (LSTM and CNN), and those extracted directly.

The goal of the neural network models is to discover nuances of how people are writing their documents. The intuition behind this is that some writing habits carry over from a first language to a second language. This includes a tendency for one group of authors to write words or sentences in a specific way [14]. One of the structures that the neural network models will learn are lexical features of the raw text. For memory

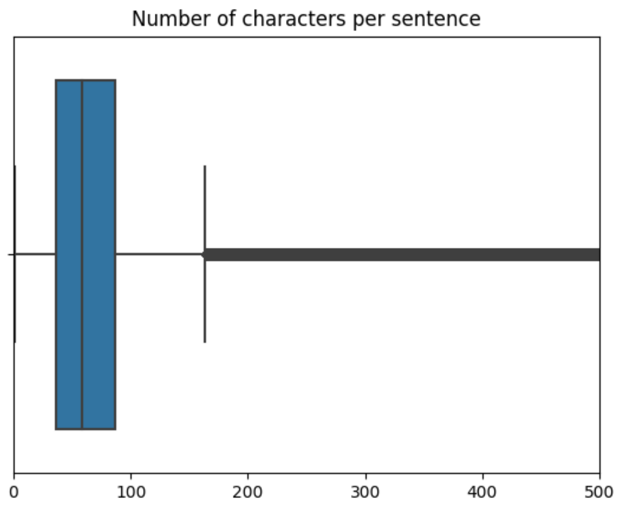


Figure 4.2: Box plot of character count per sentence across all documents

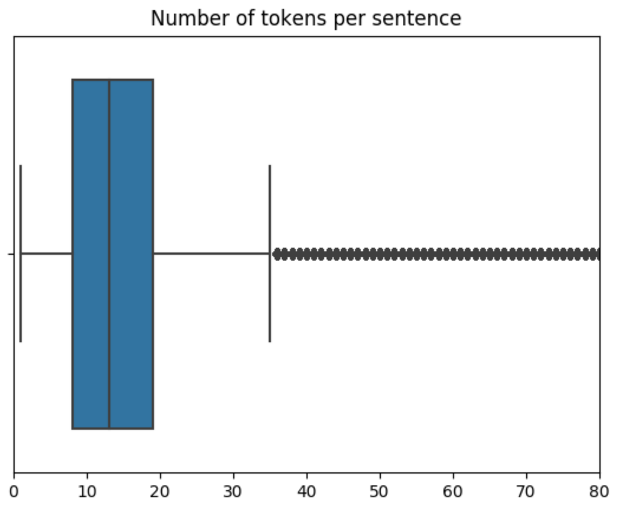


Figure 4.3: Box plot of token count per sentence across all documents

usage considerations, each document is broken down into a maximum of 25 sentences, from the first occurring. As seen in figure 4.1, over 99% of all documents in the dataset have fewer than 25 sentences. From each sentence, the first 300 characters are used as input to the neural network. Sentences which contain fewer than 300 characters account for more than 99% of the dataset, as seen in figure 4.2. Each character has a unique integer assigned to it, which is used as the input index to the neural network. A Convolutional layer is applied on the input to create an embedding using three filters of sizes 5, 3, and 3. The corresponding filters have a dimensional output of 196, 196, and 256, respectively. Each CNN layer uses a rectified linear unit activation, and passes through a dropout layer with a 10% rate.

The neural network models can also learn grammatical features from the text's part-of-speech (POS) tags. Like the first model, each document is broken down into the first 25 sentences. Using the Natural Language Toolkit (NLTK), each sentence is divided into a maximum of 50 tokens, consisting of words and punctuation. Over 98% of all documents contains sentences with fewer than 50 tokens. Each of these tokens have a corresponding POS tag, which are used to construct the document in place of the original tokens. Since there are only 45 unique POS tags used in the corpus, we have used the same technique as mentioned in the previous paragraph to generate embeddings for the POS tag inputs using a Convolutional layer.

The last structure for the neural network to learn are semantical representations using the raw words of the text. Here, we use the same tokens used for generating POS tags, but use the actual word of the token. Because there are so many unique words used in the corpus, we have trained a word2vec model to generate embeddings on all of the words. A description of word2vec and it's usage can be found in chapter 4.8. Each word has a 300-dimensional embedding from the word2vec model. These embeddings are used as input to the neural network.

The LSTM model implemented here uses two bi-directional LSTM layers, one for sequences of POS tags, tokens, or characters in each sentence, and one for the sequences of sentences in each document. Each LSTM layer has a dropout rate of 15%. The results of these LSTM layers go through an additional dropout layer with a dropout rate of 30%. The output is then passed to a fully-connected neural network with softmax activation, producing a probability distribution. This model uses the RMSprop optimizer, which divides the gradient by a running average of its recent magnitude.

The CNN model takes the same input as the LSTM model. The input sequence goes trough 4 filters of sizes 128, 256, 192, and 320 units, each with a tanh activation. The filters are passed as pairs (128 and 256, and 192 and 320) through a fully-connected neural network with a relu activation. This is then passed to two single directional LSTM layers, each with a dropout rate of 10%. The output of LSTM is then passed to another fully-connected neural network with softmax activation which outputs the probability distribution. This model also uses the RMSprop optimizer.

In addition to structures learned from the neural networks, supplementary features have been extracted from each document.

- Sentence length

  Determined by the number of tokens in the document. Here, a token represents an instance of a word or punctuation. It is expected that the lower level units will have fewer tokens per document, as higher level units have likely more experience writing longer sentences in English.

- Determiner-to-token ratio

  The number of determiners (*the*, *a*, *an*, etc.) used in a document divided by the number of tokens in the document.

- Punctuation-to-token ratio

The number of punctuation tokens (commas, periods, apostrophes, etc) used in a document divided by the number of tokens in the document.

- Ratio of long sentences

  The ratio of sentences which consist of over 60 tokens. [15] states that very long sentences are common in Chinese authors as it is acceptable to put several supporting ideas into one sentence.

- Ratio of short sentences

  The ratio of sentences which consist of fewer than 5 tokens.

- Ratio of sentences using the passive voice

  A passive voice is a sentence where a noun that would be the object of an active voice sentence becomes the subject of a sentence. One example of an active voice is *"The dog chased Frank"* where the corresponding passive voice is *"Frank was chased by the dog"*. Here, we determine a sentence to be identified by a verb token followed by the word *by*. There are other ways to identify passive voice sentences, however they are not currently implemented here.

- Rare-occurring POS bi-grams

  Koppel et al. [5] states that pairs of POS tags that are uncommon might indicate grammatical errors, or at least a non-standard usage. Instances of these pairs might be exhibited in certain groups. Here, a rare-occurring POS bi-gram is determined by a frequency between 0.05% and 20% of all documents.

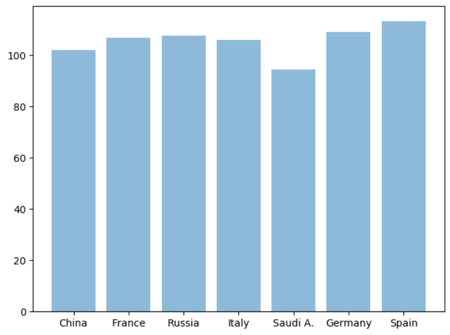
- Common spelling errors

  Spelling errors themselves do not necessarily help predict an author's native language, as even writers of one's own language can be prone to have spelling mistakes. However, if a particular word is misspelled among several authors of a group, it might indicate that some words are difficult for a certain group. Here, a common spelling error is considered if the frequency is between 0.05% and 20% of all documents.

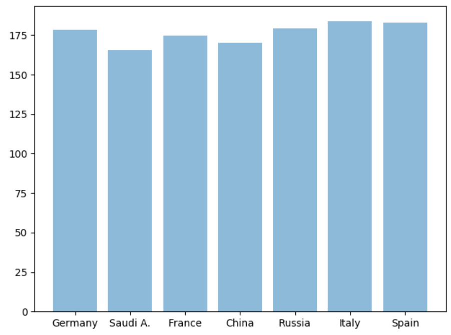
- Word uni-grams

  Word n-grams have been included in several past approaches, especially in teams that participated in [6]. Word uni-grams themselves are used as a baseline of performance, which will be described in chapter 5.1. Because considering every uni-gram can lead to a massive size of features, a Singular Value Decomposition was performed on the uni-grams matrix and the feature size was reduced to 500.

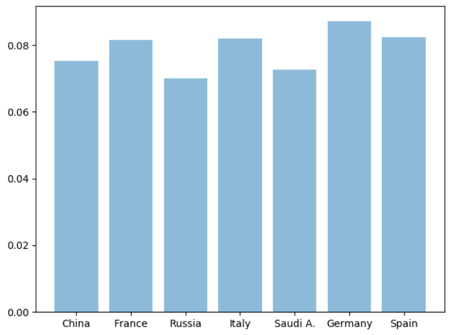
Figures 4.4 and 4.5 show the averages of some of these features across all documents in each label. In Figures 4.4(a) and 4.4(b), we can see that all labels have a higher amount in the higher level units, which is not unexpected, as higher level units have more experience writing longer sentences in English. Figures 4.4(c) and 4.4(d) shows that Saudi, Chinese, and Russian authors tend to use fewer determiners than others. [15] notes this about Chinese authors as Mandarin has no direct equivalent of determiners and the rules for using them are difficult for non-native writers and speakers. There is not a lot of variation among the different countries in punctuations as seen in figures 4.4(e) and 4.4(f), but Russian authors seem to make use of punctuations slightly more than others. According to [15], Chinese authors tend to group related ideas into a single sentence, and it shows here in the number of long sentences with figures 4.5(a) and 4.5(b). Authors from Saudi Arabia also seem to write very long sentences as well, especially in the higher level units. It would appear that Russian authors tend to write shorter sentences than others as seen in figures 4.5(c) and 4.5(d). Curiously, Chinese authors also write shorter sentences than others, although [15] made no mention of that as a common trait. Not surprisingly, all countries show



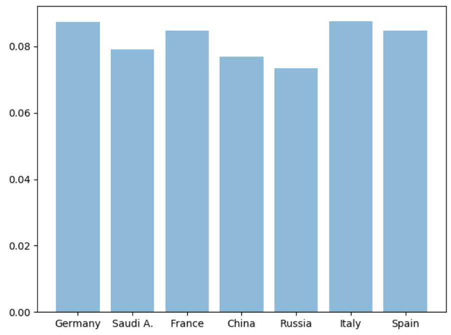
(a) Units 5-10



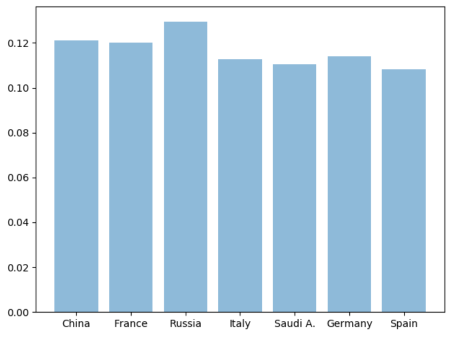
(b) Units 11-16



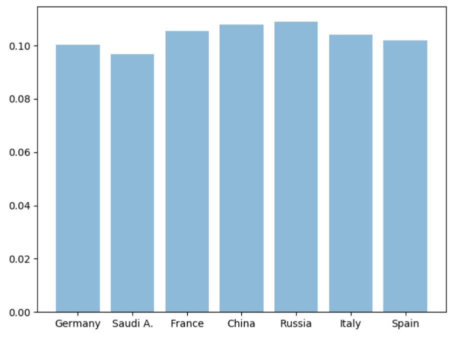
(c) Units 5-10



(d) Units 11-16

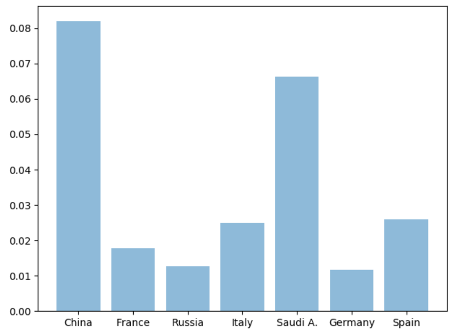


(e) Units 5-10

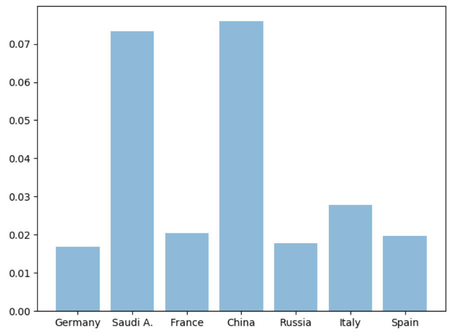


(f) Units 11-16

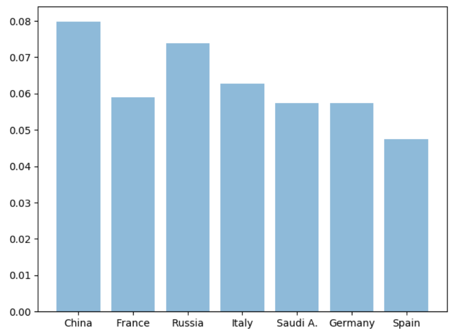
Figure 4.4: Average number of tokens (a, b), determiner ratio (c, d), and punctuation ratio (e, f) per document across each label



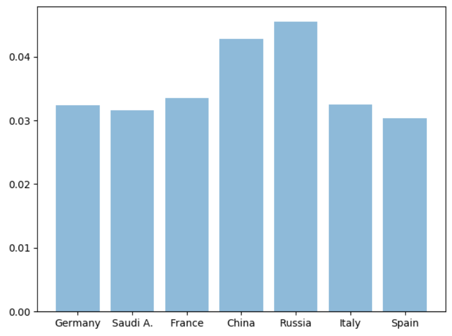
(a) Units 5-10



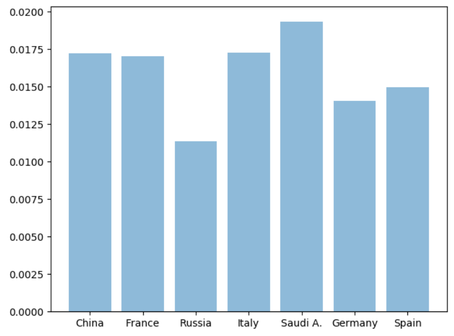
(b) Units 11-16



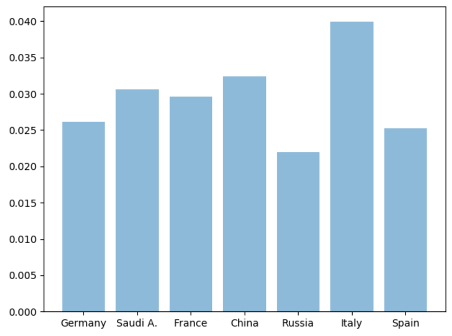
(c) Units 5-10



(d) Units 11-16



(e) Units 5-10



(f) Units 11-16

Figure 4.5: Average number of long (a, b), short (c, d), and passive sentence (e, f) ratios per document across each label

Correlation coefficients		
	Units 5-10	Units 11-16
Number of tokens	0.0005	0.0301
Determiner-token ratio	0.0383	0.0746
Punctuation-token ratio	-0.1033	-0.0864
Long sentence ratio	-0.0773	-0.0286
Short sentence ratio	-0.0706	-0.0456
Passive sentence ratio	0.0009	0.0118

Table 4.1: Correlation coefficients for each feature

fewer cases of these in the higher level units. From figures 4.5(e) and 4.5(f), we can see that French and Italian authors tend to write sentences in the passive voice more often than others, more so in the higher level units. Another feature considered was the average length of words. After analysis of this feature however, it was discovered that there is very little variation in word lengths across different labels. The average word length across all labels in both groups were around 4.2 characters.

In order to get an idea of how well these features help in deciding a label, we can see what the correlation coefficient of each feature. Table 4.1 shows the correlation coefficients of each feature for each group. According to the table, we can that the determiner-token ratios are likely to help the most to classify documents in both groups. However, negative coefficients do not necessarily mean that a feature is unhelpful, especially if combined with other features.

4.4 Recurrent Neural Network

Recurrent neural networks are a class of neural networks that learn structures of their input. The idea is to take a sequence of inputs, and learn a pattern by taking into account each previous input. With what has been learned from previous experiences, the neural network can make predictions on what will happen next. Figure 4.4[16] shows a sample of input, hidden layer, and output configurations. Each input is part of an input sequence x_i, from x_0 to x_n. Each hidden layer h_i excluding h_0,

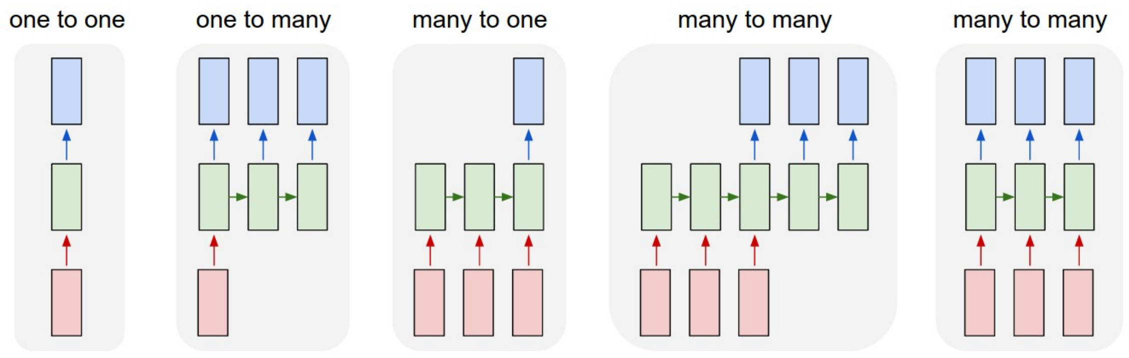


Figure 4.6: Recurrent Neural Networks. Red, green, and blue blocks represent inputs, hidden layers, and outputs, respectively.

receives input from h_{i-1} and x_i. Hidden layers can have either an output, or just pass information to the next layer. The sequence of inputs shape how each hidden layer pass information to the next hidden layer and output.

An example of how this model can be useful is to think of a book, song, or movie. When we read books, we think about what we read before to understand and predict what happens next at any given point. Likewise with music and films.

4.5 Long Short-Term Memory

RNNs have a fundamental weakness; they have trouble connecting inputs that have a large gap. For example, an RNN given the sequence "*the Earth revolves around the*" might easily predict that the next word will be "*Sun*". But let's say we have a sentence early on in a document that states "*I come from Spain*". Then some time later, the input "*my language is*" is received. Anyone who knows the context of the document would probably guess the next word is "*Spanish*". An RNN might be able to predict some language as the next word, but for it to predict the correct language, it will need to remember more context.

This is what Long Short-Term Memory (LSTM) attempts to solve. It extends the RNN model by adding a memory cell c_t to the architecture. This is defined by equations 4.1, 4.2, 4.3, 4.4, and 4.5, where x_t is the input at time t, i_t is the input gate, f_t is the forget gate, o_t is the output gate, and h_t is the input from h_{t-1}. W, U, and b are parameters to the LSTM.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{4.1}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{4.2}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{4.3}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \sigma(W_c x_t + U_c h_{t-1} + b_c) \tag{4.4}$$

$$h_t = o_t \odot \sigma(c_t) \tag{4.5}$$

LSTM has seen several applications in NLP research, such as language modeling [17] and sequence tagging [18].

4.6 Convolutional Neural Network

Convolutional neural networks (CNN) are a class of neural networks that act as detection filters for patterns and features within the original data. The first layer of a CNN will typically take a large chunk of the original data, and represent that chunk as a smaller chunk. This process can be repeated many times, representing the data in smaller and smaller chunks. CNNs have been used with great success in image recognition[19], edge detection[20], and video analysis[21]. Applications of CNNs have also been used with NLP tasks [22], even combining CNN and RNN [23] [24], with interesting results.

4.7 Support Vector Machines

Support Vector Machines (SVM) were first introduced by Bosner et al. in the 1992 paper *A Training Algorithm for Optimal Margin Classifiers* [25]. SVMs are often

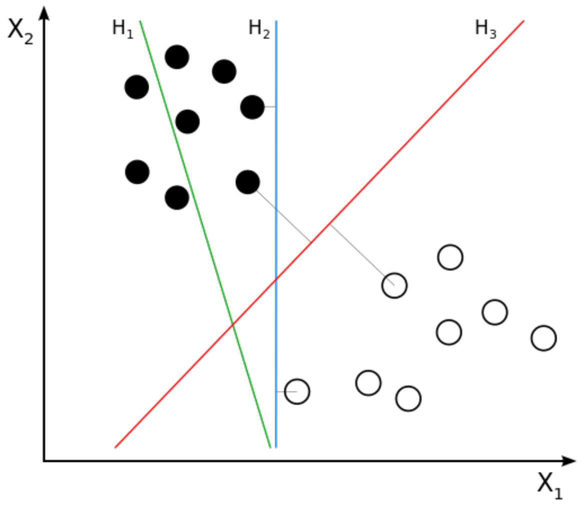


Figure 4.7: 3 Hyperplane separating labels

used in regression and classification tasks and have a simple objective. Given a set of vectors $X_1^d..X_m^d$ as inputs with labels $Y_1...Y_n$, find a hyperplane, or boundary, that separates vectors belonging to Y_i from Y_j with a maximum margin of separation between the nearest training samples. Figure 4.7[26] shows 3 hyperplanes set in a 2-dimensional plane. H_1 does not accurately separate the data. H_2 does separate the different labeled data, but with a small margin. H_3 separates these labeled points, and with the maximum margin.

4.8 Word2Vec

Word2Vec is a shallow neural network introduced by Mikolov et al. in 2013 [27]. It's function is to take a text input and output an n-dimensional Euclidean space representing word embeddings. a text frequency-inverse document frequency (TFIDF) can create word embeddings, but these embeddings only indicate a presence in the document. Word2Vec on the other hand, trains a Euclidean space based on the proximity of nearby words. This also has the benefit of keeping the dimensionality of the embeddings to a much smaller degree than if using TFIDFs on large corpora. Figure 4.8 shows the two model architectures of Word2Vec: Continuous Bag Of Words

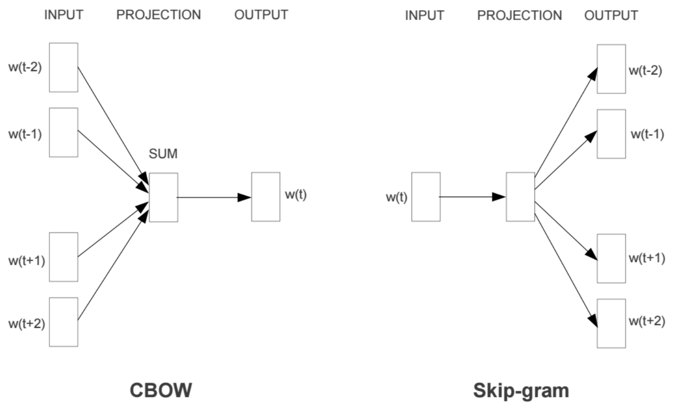


Figure 4.8: Word2Vec model architectures

(CBOW), and Skip-gram. The CBOW model takes surrounding words as input, and outputs a word that falls into the context of the input. Take for example the sentence "*The quick brown fox jumped over the lazy dog*". Inputting the words *quick*, *brown*, *jumped*, and *over* into a CBOW trained Word2Vec model would predict the word *fox*. With the Skip-gram architecture, the input is a word, and the output is a context surrounding that word. Using the same sentence as an example, inputting the word *fox* would output the words *quick*, *brown*, *jumped*, and *over*. A simple way of thinking of these two architectures is that CBOW predicts a word given a context, and Skip-gram predicts a context given a word.

Chapter 5

Results and Discussions

In this chapter, we will present and discuss the dataset used, the experiment methods and parameters, and the results.

5.1 Experiment setup

Before running an experiment, the sequences and supplementary features described in chapter 4.3 are extracted from all documents. Each sequence is saved in a separate 3-dimensional matrix which is used as inputs for the neural networks. The supplementary features are saved to be later used in union with the probability distributions of the trained neural network model.

The system is run across a 3-fold cross validation, each with a chosen neural network model and dataset group. The data is split into 66% training and 33% testing. The neural network is trained over a maximum of 50 epochs. Each epoch only continues with the model that has the lowest loss on the testing set. An early termination will occur if the model's evaluation of the testing set at the end of an epoch does not show improvement within the last 7 epochs. After the neural network model has been trained, it will make predictions on the entire dataset, producing a matrix of probability distributions of the labels for each document. This matrix is then concatenated with the matrix of supplementary features. The resulting matrix is then used to train a linear SVM with using the primal optimization algorithm and 'one vs. the-rest' strategy, over a maximum of 1000 iterations. At the end of the

Lower unites dataset (136,889 documents) 3-fold Cross Validation					
	# of features	SVM Micro F-1/Accuracy	SVM Macro F-1	Neural Network loss	Neural Network accuracy
Dummy classifier with stratified strategy	N/A	19.49%	14.37%	N/A	N/A
Dummy classifier with uniform strategy	N/A	14.33%	13.09%	N/A	N/A
Word uni-grams TF-IDF	115296	70.36%	63.11%	N/A	N/A
Additional features only	1477	58.97%	48.05%	N/A	N/A
LSTM					
POS	1484	61.45%	51.66%	1.3592	52.63%
Words	1484	70.44%	63.50%	1.1515	**70.25%**
Chars	1484	70.84%	63.22%	1.0921	66.46%
Combined (W,C)	1491	**72.14%**	**65.15%**	**1.0768**	68.58%
Convolutional					
POS	1484	57.96%	47.91%	1.6769	46.50%
Words	1484	66.39%	56.47%	1.1541	60.03%
Chars	1484	65.82%	57.98%	1.3335	59.78%
Combined (W,C)	1491	69.15%	61.01%	1.2415	59.34%

Table 5.1: Classification results for units 5-10 using probability distribution from neural networks

cross validation, the average micro and macro F1 scores are displayed, as well as the average accuracy and loss of the trained neural network on the testing test.

Baseline tests were also performed. In one scenario, dummy classifiers were used having stratified and uniform strategies. These dummy classifiers made random predictions, with the stratified strategy having probabilities proportional to the label distribution, while the uniform had equal chances for all labels. Another baseline used a simple word uni-gram TF-IDF as its feature set. All unique words across the corpus were considered. This was used with the Linear SVM classifier. Lastly, the additional features described in Chapter 4.3 without the use of any neural network were tested. This also used the Linear SVM classifier.

5.2 Results

Several variations of the LSTM and CNN models were used in the experiments. For each type of neural network, the model was trained to extract and learn structures using only the POS sequences, word sequences, character sequences, and a combination of words and characters. Tables 5.1 and 5.2 show the results of the experiments when using the fully-connected neural network layer that produces the probability distribution of each document. In the lower units, the LSTM model using word and character sequences, in combination with the supplemental features, scored the high-

Higher units dataset (17,371 documents) 3-fold Cross Validation					
	# of features	SVM Micro F-1/Accuracy	SVM Macro F-1	Neural Network loss	Neural Network accuracy
Dummy classifier with stratified strategy	N/A	18.82%	14.10%	N/A	N/A
Dummy classifier with uniform strategy	N/A	13.89%	12.94%	N/A	N/A
Word uni-grams TF-IDF	45330	**58.53**%	**52.11**%	N/A	N/A
Additional features only	1756	52.96%	45.65%	N/A	N/A
LSTM					
POS	1710	45.34%	39.39%	1.8971	35.94%
Words	1710	53.42%	46.79%	2.1105	**53.13%**
Chars	1710	46.74%	40.63%	**1.7719**	39.17%
Combined (W,C)	1717	52.70%	46.68%	2.1376	47.517%
Convolutional					
POS	1710	46.16%	40.56%	2.0116	30.58%
Words	1710	47.17%	40.84%	1.8962	35.12%
Chars	1710	50.30%	43.90%	1.9781	39.02%
Combined (W,C)	1717	48.21%	42.40%	1.8897	36.12%

Table 5.2: Classification results for units 11-16 using probability distribution from neural networks

Lower unites dataset (136,889 documents) 3-fold Cross Validation					
	# of features	SVM Micro F-1/Accuracy	SVM Macro F-1	Neural Network loss	Neural Network accuracy
Dummy classifier with stratified strategy	N/A	19.49%	14.37%	N/A	N/A
Dummy classifier with uniform strategy	N/A	14.33%	13.09%	N/A	N/A
Word uni-grams TF-IDF	115296	70.36%	63.11%	N/A	N/A
Additional features only	1477	58.97%	48.05%	N/A	N/A
LSTM					
POS	1733	62.35%	53.06%	1.3882	51.63%
Words	1733	70.42%	63.63%	1.1457	**70.48%**
Chars	1733	72.26%	64.80%	**0.9909**	67.39%
Combined (W,C)	1989	**73.94%**	**67.02%**	1.0483	69.10%
Convolutional					
POS	1569	57.39%	48.05%	1.6134	47.99%
Words	1569	67.30%	57.95%	1.1687	60.48%
Chars	1569	66.40%	58.28%	1.3567	60.14%
Combined (W,C)	1661	69.65%	61.15%	1.2556	60.17%

Table 5.3: Classification results for units 5-10 using neural network's final layer

Higher units dataset (17,371 documents) 3-fold Cross Validation					
	# of features	SVM Micro F-1/Accuracy	SVM Macro F-1	Neural Network loss	Neural Network accuracy
Dummy classifier with stratified strategy	N/A	18.82%	14.10%	N/A	N/A
Dummy classifier with uniform strategy	N/A	13.89%	12.94%	N/A	N/A
Word uni-grams TF-IDF	45330	**58.53**%	**52.11**%	N/A	N/A
Additional features only	1756	52.96%	45.65%	N/A	N/A
LSTM					
POS	1959	45.88%	39.21%	1.9327	37.26%
Words	1959	53.65%	47.63%	2.1519	**52.54%**
Chars	1959	47.68%	41.99%	**1.7708**	39.31%
Combined (W,C)	2215	55.35%	48.90%	1.9633	47.71%
Convolutional					
POS	1795	43.70%	37.78%	2.0699	33.99%
Words	1795	44.44%	38.65%	1.9053	36.04%
Chars	1795	50.28%	44.17%	1.7993	37.39%
Combined (W,C)	1887	47.61%	41.72%	1.9492	34.64%

Table 5.4: Classification results for units 11-16 using neural network's final layer

est accuracy at 72.14%, slightly higher than the naive TF-IDF approach at 70.36% accuracy. When using either word or character sequences with the LSTM, the final SVM predictions were only able to achieve marginally better accuracies than the TF-IDF. Across both LSTM and CNN models, using POS sequences gave the lowest accuracies. This might indicate that the lower unit essays across multiple languages have less variations in their grammar structure. It is possible that since most sentences across the dataset consist of fewer than 15 tokens as mentioned in chapter 4.3, the grammar structure is difficult to learn. The higher units have different results. None of the neural networks were able to achieve better results than the TF-IDF approach. In this group, the LSTM using word embeddings achieved the highest accuracy among the neural networks. In the higher level units, it's likely that since the authors are more proficient in English, there are not as many grammar or spelling errors, so there would be more defining features found in word usage.

Tables 5.3 and 5.4 show the results of the experiments without the probability distributions from the fully-connected neural network layer. Normally, the probability distribution would give a probability for each label that the document could be assigned to. This means that the output of the final layer for the LSTM and CNN models, 128 and 96 dimensions respectively, would be reduced to 7 dimensions. This might result in some lost information that an SVM classifier could better utilize. This experiment sees if using the output of the final layer of the neural network would help the SVM classifier achieve better results. There is a small increase in most experiment results when using this approach on the lower units group. The higher units group does not see much benefit however. Curiously, the SVM classifier made worse predictions with the CNN's results, even though there was an increase in the CNN's accuracies with the POS and Word sequences.

Chapter 6

Conclusion and Future Work

We studied the performance of different Long Short-Term Memory and Convolutional neural network configurations used to predict the native language of authors of English essays. We used neural network models to extract lexical, semantic, and grammatical features using character, word, and part-of-speech tag sequences, respectively. These features, combined with other statistical features, were extracted from essays obtained by the EF-Cambridge Open Language Database corpus and classified with a linear Support Vector Machine. The dataset was split into 2 groups representing essays from intermediate levels and more advanced levels. The results show that these features are more clear in the lower levels group than in the higher levels group. Using a Long Short-Term Memory model can provide marginally better results than a naive word uni-grams TF-IDF feature set, but with a much smaller feature set.

Neural networks are very tunable with hyperparameters and layer configurations. a few hyperparameters such as unit sizes and learning rates were experimented with during this thesis, but a more thorough search using random or grid search could see some benefits in the results. The layer configurations could also use more experimenting with. Only one LSTM layer at the sentence level was used, so having more layers might also see some changes. Cimino [9] was able to achieve some of the best results by using a meta classifier. One classifier predicts languages at the sentence level, and uses those predictions to help another classifier predict at the document level. This is an area that could be explored with neural networks.

REFERENCES

[1] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, 1950. [Online]. Available: http://www.jstor.org/stable/2251299

[2] D. Nadeau and S. Sekine. (2007) A Survey of Named Entity Recognition and Classification. [Online]. Available: http://nlp.cs.nyu.edu/sekine/papers/li07.pdf

[3] S. Malmasi and M. Dras, "Native Language Identification using Stacked Generalization," *arXiv preprint arXiv:1703.06541*, 2017.

[4] S. MASSUNG and C. ZHAI, "Non-native text analysis: A survey," vol. -1, pp. 1–24, 09 2015.

[5] M. Koppel, J. Schler, and K. Zigdon, "Determining an author's native language by mining a text for errors," in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, ser. KDD '05. New York, NY, USA: ACM, 2005, pp. 624–628. [Online]. Available: http://doi.acm.org/10.1145/1081870.1081947

[6] J. Tetreault, D. Blanchard, and A. Cahill, "A report on the first native language identification shared task," pp. 48–57, 01 2013. [Online]. Available: http://aclweb.org/anthology/W13-1706

[7] D. Blanchard, J. Tetreault, D. Higgins, A. Cahill, and M. Chodorow, "Toefl11: A corpus of non-native english," vol. 2013, pp. i–15, 12 2013. [Online]. Available: https://www.ets.org/Media/Research/pdf/RR-13-24.pdf

[8] S. Malmasi, K. Evanini, A. Cahill, J. Tetreault, 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*. Copenhagen, Denmark: Association for Computational Linguistics, September 2017. [Online]. Available: http://web.science.mq.edu.au/~smalmasi/papers/nli-2017.pdf

[9] A. Cimino and F. Dell'Orletta, "Stacked sentence-document classifier approach for improving native language identification," in *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications.*

Association for Computational Linguistics, 2017, pp. 430–437. [Online]. Available: http://aclweb.org/anthology/W17-5049

[10] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: ACM, 2008, pp. 160–167. [Online]. Available: http://doi.acm.org/10.1145/1390156.1390177

[11] A. Radford, R. Józefowicz, and I. Sutskever, "Learning to generate reviews and discovering sentiment," *CoRR*, vol. abs/1704.01444, 2017. [Online]. Available: http://arxiv.org/abs/1704.01444

[12] J. Bjerva, G. Grigonyte, R. stling, and B. Plank, *Neural Networks and Spelling Features for Native Language Identification*, 2017. [Online]. Available: http://www.aclweb.org/anthology/W17-5025

[13] "Ef - cambridge open language database." [Online]. Available: https://corpus.mml.cam.ac.uk/efcamdat2/public_html/

[14] R. A. BERMAN and E. OLSHTAIN, "Features of first language transfer in second language attrition," *Applied Linguistics*, vol. 4, no. 3, pp. 222–234, 1983. [Online]. Available: http://dx.doi.org/10.1093/applin/4.3.222

[15] F. Brittman, "The most common habits from more than 200 english papers written by graduate chinese engineering students," 01 2007. [Online]. Available: http://papertalks.org/p/resources/Academic/EnglishWritingSkills/MostCommonEnglishWritingHabitsOfChinese.pdf

[16] A. Karpathy, "The unreasonable effectiveness of recurrent neural networks," https://karpathy.github.io/2015/05/21/rnn-effectiveness/, May 2015, accessed on 2018-03-26.

[17] M. Sundermeyer, R. Schlter, and H. Ney, "Lstm neural networks for language modeling." [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.248.4448

[18] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," *CoRR*, vol. abs/1508.01991, 2015. [Online]. Available: http://arxiv.org/abs/1508.01991

[19] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng, and M. Chen, "Medical image classification with convolutional neural network," in *2014 13th International*

Conference on Control Automation Robotics Vision (ICARCV), Dec 2014, pp. 844–848.

[20] M. El-Sayed, Y. A. Estaitia, and M. A. Khafagy, "Edge detection using convolutional neural network," vol. 4, pp. 11–17, 11 2013.

[21] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *CVPR*, 2014.

[22] R. Johnson and T. Zhang, "Effective use of word order for text categorization with convolutional neural networks," *CoRR*, vol. abs/1412.1058, 2014. [Online]. Available: http://arxiv.org/abs/1412.1058

[23] V. D. Van, T. Thai, and M.-Q. Nghiem, "Combining convolution and recursive neural networks for sentiment analysis," in *Proceedings of the Eighth International Symposium on Information and Communication Technology*, ser. SoICT 2017. New York, NY, USA: ACM, 2017, pp. 151–158. [Online]. Available: http://doi.acm.org/10.1145/3155133.3155158

[24] J. Y. Lee and F. Dernoncourt, "Sequential short-text classification with recurrent and convolutional neural networks," *CoRR*, vol. abs/1603.03827, 2016. [Online]. Available: http://arxiv.org/abs/1603.03827

[25] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ser. COLT '92. New York, NY, USA: ACM, 1992, pp. 144–152. [Online]. Available: http://doi.acm.org/10.1145/130385.130401

[26] User:ZackWeinberg. (2012) Svm separating hyperplanes. [Online]. Available: https://commons.wikimedia.org/wiki/File:Svm_separating_hyperplanes_(SVG).svg

[27] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781