NTNU
Norwegian University of
Science and Technology

A Deep Learning Ensemble Approach to Gender Identification of Tweet Authors

Per-Christian Berg
Manu Gopinathan

Master of Science in Computer Science
Submission date: June 2017
Supervisor: Björn Gambäck, IDI

Norwegian University of Science and Technology
Department of Computer Science

NTNU – Trondheim
Norwegian University of
Science and Technology

A Deep Learning Ensemble Approach to Gender Identification of Tweet Authors

Per-Christian Berg
Manu Gopinathan

Master's Thesis in Computer Science
Supervised by Björn Gambäck
Spring 2017

Data and Artificial Intelligence Group
Department of Computer Science
Faculty of Information Technology and Electrical Engineering
Norwegian University of Science and Technology

Abstract

Author profiling is a field within Natural Language Processing, in addition to being a sub-field of the broader research area concerning authorship analysis, and aims to classify personal traits of authors, such as gender and age, based on their writing style. It is of growing importance with applications within fields such as forensics and marketing for identifying characteristics of perpetrators and customers, respectively. The emergence of social media platforms, such as Twitter, has resulted in a major increase in textual user-generated content publicly available for linguistic studies. Additionally, the informal language present in tweets provides linguistic material reflecting people's everyday usage of language.

Though representation learning using deep learning has shown much promise, most of the work within author profiling research in recent years has been based on the combination of expensive manual feature engineering, representations such as Bag of Words, and traditional machine learning methods exemplified by Support Vector Machines and Logistic Regression. In this thesis we show that better gender-identifying feature representations of English tweets can be learned using deep learning approaches.

We propose three classification systems, focusing on different granularities of text: a character-level Convolutional Bidirectional Long Short-Term Memory (LSTM), a word-level Bidirectional LSTM using Global Vectors (GloVe), and a more traditional document-level system utilizing a feedforward network and Bag of Words of n-grams as first-level representation. Furthermore, we propose using stacking to leverage the individual predictive powers of the sub-models in a combined effort. The experiments reveal that the word-level model outperforms the other sub-models, as well as the baseline models consisting of Logistic Regression, Naïve Bayes and Random Forest. The best performance is achieved by combining the character-level and word-level models, while the document-level model dampens the combined performance.

Sammendrag

Forfatterprofilering er et felt innenfor språkteknologi og har som mål å predikere personlige egenskaper ved forfattere, for eksempel kjønn og alder, basert på deres skrivestil. Mulige bruksområder for forfatterprofileringssystemer inkluderer kriminaletterforskning og markedsføring for å identifisere henholdsvis gjerningsmenn og kunder. Det siste tiåret har sosiale medier, som Twitter, opplevd en kraftig vekst i brukerbase. Dette har ført til en enorm økning i brukergenerert tekstlig innhold som er offentlig tilgjengelig for forskning innenfor lingvistikk. I tillegg gjør det uformelle språket i "tweets" det mulig å studere hverdagsbruk av språk.

Selv om dyp læring har vist seg å være fremragende for å lære representasjoner av tekst, viser det seg at mesteparten av arbeidet innenfor forfatterprofilering de siste årene har basert seg på en kombinasjon av manuell utplukking av tekstegenskaper, representasjonslæring med modeller, som Bag of Words, og tradisjonelle maskinlæringsmetoder, f.eks støttevektormaskiner (SVM) og logistisk regresjon. I denne masteroppgaven påviser vi at dyp læring kan brukes til å utarbeide representasjoner av tekst som har bedre evne til å fange opp kjønnsbaserte karakteristikker.

Vi foreslår tre klassifiseringssystemer som behandler tekst på ulike granularitetsnivåer: en Convolutional Bidirectional Long Short-Term Memory (LSTM) på tegn-nivå, en Bidirectional LSTM på ord-nivå som bruker Global Vectors (GloVe), og et mer tradisjonelt system på dokument-nivå som bruker et feedforward nettverk og Bag of Words med n-grams som initiell representasjon. Videre, foreslår vi å ta bruk stacking for utnytte de ulike modellenes individuelle prediktive evner i en kombinert innsats. Eksperimentene viser at modellen på ord-nivå oppnår bedre resultater enn de andre submodellene og baseline-modellene. Det beste resultatet blir oppnådd ved å kombinere modellene på tegn-nivå og ord-nivå, mens modellen på dokument-nivå bidrar negativt.

Preface

This thesis was written by Per-Christian Berg and Manu Gopinathan, during the spring of 2017, as a part of the Master of Science (MSc) degree in Computer Science at the Department of Computer Science (IDI) at the Norwegian University of Science and Technology (NTNU). We would like to thank Björn Gambäck for valuable guidance and meticulous feedback throughout the course of this thesis. In addition, we would like to thank Brede, Mette and Liv for cleaning our apartment while we were busy finishing the thesis.

Per-Christian Berg
Manu Gopinathan
Trondheim, 11th June 2017

Contents

List of Figures

List of Tables

Acronyms

Adam	Adaptive Moment Estimation
ANN	Artifical Neural Network
BoW	Bag of Words
BPTT	Backpropagation Through Time
BRNN	Bidirectional Recurrent Neural Network
CBOW	Continuous Bag-of-Words
CNN	Convolutional Neural Network
CPU	Central Processing Unit
GloVe	Global Vectors
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HTML	Hypertext Markup Language
LR	Logistic Regression
LSA	Latent Semantic Analysis
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multilayer Perceptron
NB	Naïve Bayes
NER	Named Entity Recognition
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
PCA	Principal Component Analysis
POS	part-of-speech
ReLU	Rectified Linear Unit
RF	Random Forest
RNN	Recurrent Neural Network

SVM	Support Vector Machine
t-SNE	t-Distributed Stochastic Neighbor Embedding
TF-IDF	Term Frequency-Inverse Document Frequency
URL	Universal Resource Locator
VADER	Valence Aware Dictionary and Sentiment Reasoner
XML	Extensible Markup Language

1. Introduction

The advent of social media has resulted in a massive boost in the volume of user-generated data available. This data is useful for different analytical purposes in a wide range of domains. Twitter is one such social networking service, specifically a microblogging service. It rapidly gained worldwide popularity after its creation in 2006 and provides a public online medium for sharing news, ideas, opinions and other forms of information. With approximately 300 million monthly active users[1], an enormous number of tweets are produced every day. An important detail about tweets is that they are often written colloquially, giving insight into how the authors express themselves in an everyday manner. Linguists have taken advantage of this great amount of informal textual data to study and discover various social aspects of language. For example, Gonçalves and Sánchez (2014) built a corpus of geotagged Spanish tweets and discovered the existence of two superdialects independent of nationality, one used in urban cities and another one more prominent in rural areas. However, this thesis will focus on another demographic aspect of language in tweets, namely the difference between male and female writing. In this endeavour, deep learning architectures will be utilized to construct text classification systems that predict author gender. This problem falls within the area of author profiling, which encompasses a myriad of traits. However, this thesis will be constrained to the aspect of gender.

1.1. Background and Motivation

Unlike the problem of author identification, where the task is to purely identify the true author of a text, the goal of author profiling is to acquire information about an unknown author based on the text. This information can be in the form of characteristics and traits. A few examples include gender, age and various personality traits. A key feature of author profiling is that it studies how linguistic traits are shared by different people. There are many application areas for these types of classifications, such as forensic linguistics and marketing.

Forensic linguistics is a branch of applied linguistics which encompasses several areas, where author identification is one. However, direct identification of an author based on a text can prove to be difficult if one does not possess any known texts by the unknown author. It can be trivially assumed that to be able to correctly identify the author of a text, one must have previously sampled writings by said person to form a basis for the

[1] https://about.twitter.com/company

identification. In this case, identifying characteristics to develop a profile of the author can be more useful as this is possible even when you do not have previously sampled text by the person. This can aid, for example, criminal investigations in eliminating a suspect or narrowing down the list of suspects. From a marketing perspective, it can be useful for companies to classify the different types of people who either like or dislike their products, based on articles and reviews on the web. Author profiling, combined with sentiment analysis, can prove useful for this.

The field of Natural Language Processing (NLP) is highly relevant for the task of author profiling as it is concerned with the interactions between computers and human language, including their ability to understand and process it. In recent years, deep learning methods have gained tremendous popularity because of their state-of-the-art results across the board, NLP inclusive. Despite this, deep learning can be considered a minority within the set of popular methods used for author profiling. This is underpinned by the observation that traditional NLP representations, such as Bag of Words, and other classification methods, e.g., Support Vector Machines (SVMs), represent the majority of approaches taken for author profiling in recent years. This will be apparent when state-of-the-art approaches, submitted to PAN, are presented in Section 4.2. PAN is an annual series of workshops and shared tasks on digital texts forensics, where author profiling is one of the categories. To explore its capabilities in the field of author profiling, deep learning will have an integral part in the gender classification architectures implemented as a part of this thesis.

1.2. Goals and Research Questions

Goal *Predict the gender of tweet authors based on linguistic differences.*

The goal of this thesis is to establish which characteristics of English tweets best distinguish the gender of the author. This is for the purpose of performing gender classification of novel tweet samples and deep learning will be used in this effort. The objective consists of several steps which are described in the research questions below.

Research Question 1 *What does the literature establish as the most useful gender identifying linguistic traits?*

To understand which aspects of texts are helpful for distinguishing genders, a review of several studies related to this will be made. This will provide a broad overview of previously identified findings in different textual genres and aid the process of choosing textual features to focus on in the architecture.

Research Question 2 *How can texts be represented in deep learning systems to capture meaningful information?*

Deep learning systems are designed to work with numerical values. Therefore, tweets need to be represented in a manner that reflects this constraint. A sub-goal of the thesis will be to explore different representations of texts, at different levels of granularities.

Research Question 3 *What types of deep learning models are viable for processing and classifying tweets?*

To describe deep learning as one method would be thoroughly inaccurate. It is rather a family of different Artifical Neural Network (ANN) models that can be combined to solve representation and classification problems. Understanding how the various ANN models work in NLP is an important part of being able to construct effective representation learners and classifiers.

Research Question 4 *Can multiple deep learning architectures be used in combination to recognize different characteristics of tweets?*

The development of different classification architectures may result in the separate models excelling at identifying different characteristics of the texts. This can be a consequence of a specific design decision or a combination of these, with regards to the choice of features, representation and classification model. Therefore, we will explore the use of combined models to leverage the predictive abilities of the individual models.

1.3. Research Method

To achieve the goal of this thesis, a combination of methodologies will be used. A study of literature concerning the field of author profiling and deep learning will provide a vital foundation for the development of text representations and model architectures. This is mostly relevant for Research Questions 1–3. Additionally, a thorough statistical analysis of the dataset used for model training, will provide insight into possibly relevant features. Regarding Research Questions 3 and 4, the construction of the models will be subject to a large degree of experimentation, and for the most part quantitatively evaluated. In addition. the models will be compared to each other and to baseline models, represented by Logistic Regression, Naïve Bayes and Random Forest. Qualitative analysis will be performed by examining correctly and incorrectly predicted tweets, to perhaps be able to observe any trends.

1.4. Contributions

We contribute to author profiling research with a review of studies on linguistic gender-based differences in tweets and a study of the current state-of-the-art approaches. Additionally, we present three separately developed classification systems based on deep learning. These systems are also combined to produce ensemble models, utilizing the concept of stacking. Except for the document-level model, the constructed models outperform the baseline models. The classification systems can be summarized as follows:

1. *A Convolutional Bidirectional LSTM model processing tweets at the character-level, representing these as one-hot vectors. This model achieves an F1-score of 0.592.*
2. *A Bidirectional LSTM model using pre-trained GloVe word embeddings to process tweets at the word-level. This model achieves an F1-score of 0.604.*
3. *A feedforward model which processes tweets at the document-level. This model focuses more on feature engineering, for which it is developed using more traditional NLP methods and Bag of Words is used as tweet representation. This model achieves an F1-score of 0.585.*
4. *Several ensemble models, using aggregation functions, to combine the three separately trained sub-models for collaborative predictions. The best ensemble model is the result of combining the character-level and word-level models, producing an F1-score of 0.613.*

The results produced by these models provide a clear indication of how models based on expensive feature engineering prove inferior to well-constructed models that implicitly learn representations. Our experiments show that well-defined representations of text allow for better learning of gender-specific features, by ANNs, as opposed to manually handpicking features. With ANNs, interpretation of results is challenging. We provide a qualitative analysis of the models, showing that the level of confidence the models have in their predictions is connected to their success rate.

The code to our implementation is publicly available at: `https://github.com/perchrib/masters_thesis`

1.5. Thesis Structure

Chapter 2 presents and explains the basic concepts of the machine learning methods that are used in our research, or in reviewed literature related to our work.

Chapter 3 introduces various methods and concepts used in Natural Language Processing to represent text, such as Bag of Words (BoW) and word embeddings.

Chapter 4 provides a broad overview of the development in gender-based linguistic studies, in addition to the state-of-the-art methods for author profiling based on PAN Shared Task submissions.

Chapter 5 presents the datasets of tweets used to train and evaluate the classification models. This is followed by an analysis of the training set, which provides the basis for several architectural decisions.

Chapter 6 describes the architectures of the gender classification systems developed.

Chapter 7 describes the experiments conducted to build and evaluate the systems, and the corresponding results.

As a conclusion to the thesis, **Chapter 8** will address the research questions with an evaluation and discussion of the profiling systems, along with potential future work.

2. Machine Learning Methods

This chapter and the next cover relevant theory, within the scope of this project, in the fields of machine learning and Natural Language Processing (NLP), respectively. This serves as an introduction to terminology and methods that are used in our research and in the reviewed literature related to our work. In most of today's work on text classification, some form of machine learning algorithm is used. While Section 4.2 will discuss the state-of-the-art techniques, this chapter will describe the basic concepts of the most commonly used supervised learning methods. The topic of deep learning has been given a separate section as a result of a more elaborate review.

2.1. Logistic Regression

Logistic Regression (LR) is practically a single layer neural network (described in Section 2.5.2) with only one neuron, in addition to the input layer. Despite the name, the method is used for linear binary classification rather than regression. The model takes a vector $\boldsymbol{x}$ with dimension size n as input. The number of dimensions corresponds to the number of features in $\boldsymbol{x}$. Given the input features $x_i \in \boldsymbol{x}$ where $i \in 1, ..., n$, Logistic Regression defines a mapping to $y = f(\boldsymbol{x_n}; \boldsymbol{w_n})$, which the model learns by updating the weights $w_i \in \boldsymbol{w}$ where $i \in 0, ..., n$. This is done iteratively to find the optimal values for the weights in $\boldsymbol{w}$. The search for optimal values is done by minimizing a maximum-likelihood loss function, defined as the negative log-likelihood, to find the model's error using gradient descent. Logistic Regression uses the sigmoid function $\sigma(z)$ (also called the logistic function), shown in Equation (2.1), as activation function. The sigmoid function holds the property that for any given value z in $\mathbb{R}$, the output is in the interval $[0, 1]$. The output value represents the model's classification confidence in binary classification, where a value closer to 0.0 indicates the first class, while a value closer to 1.0 indicates the other class.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.1}$$

Figure 2.1 displays how the model works, in accordance with Equation (2.2). The model takes a feature vector $\boldsymbol{x}$ and does element-wise multiplication with the weights $\boldsymbol{w}$, before the weights are updated with respect to the error of the output. w_0 is a special value called the bias and is multiplied with a fixed constant x_0, which is set equal to 1.0 in the feature vector $\boldsymbol{x}$. The interpretation of the weights and bias is further discussed in Section 2.5.2 about ANNs. In the case of binary classification, the error is calculated with a loss function, which measures the difference between the correct label $y = \{0, 1\}$

and the predicted output label $\hat{y} = [0, 1]$. For multiclass classification, a generalized version of Logistic Regression exists, called Multinomial Logistic Regression.

$$y = f(\boldsymbol{x_n}; \boldsymbol{w_n}) = \sigma(\mathbf{w^T x}) = \sigma\Big(\sum_i^N w_i x_i\Big) \tag{2.2}$$

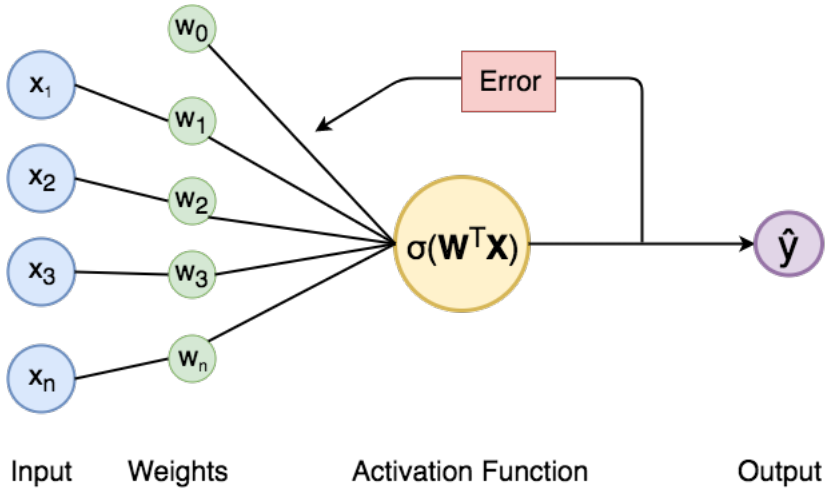


Figure 2.1.: Logistic Regression architecture.

2.2. Support Vector Machines

The Support Vector Machine (SVM) is a widely used algorithm for classification problems, as will be shown in Section 4.2 about state-of-the-art. It has undergone multiple methodological iterations, but the currently used standard is the one presented by Cortes and Vapnik in 1995. The SVM is a binary linear classifier which, given a spatial representation of the training data, will construct a model that maps new data onto the same space for classification. The algorithm tries to find the optimal hyperplane to divide the data of two classes. A hyperplane is a subspace of one dimension less than the ambient space. Figure 2.2 illustrates this in a two dimensional setting. The points closest to the hyperplane are called the support vectors. The space between the hyperplane and either support vector is known as the margin. When classifying new data, they will be mapped to this space and end up on either side of the hyperplane. There are multiple possible hyperplanes that can divide the training set, but not all of them will generalize well to new data. An optimal hyperplane will be as far away as possible from all data points, while still correctly dividing the data. This is because of the increasing probability of misclassification if the hyperplane is close to data points of either class. A slack variable is tuned to balance bias and variance. With small slack values, the margin can be larger and allow for errors when fitting the training data. This could result in a better generalized model and solve slightly non-linear problems. Many problems are, in fact, not

linearly separable, as exemplified by the XOR function illustrated in Figure 2.3. To deal with this, the SVM can be kernelized, which means that the original data is projected to a space of higher dimension, where it becomes linearly separable. The SVM is thus trained on the data in the higher dimensional space.

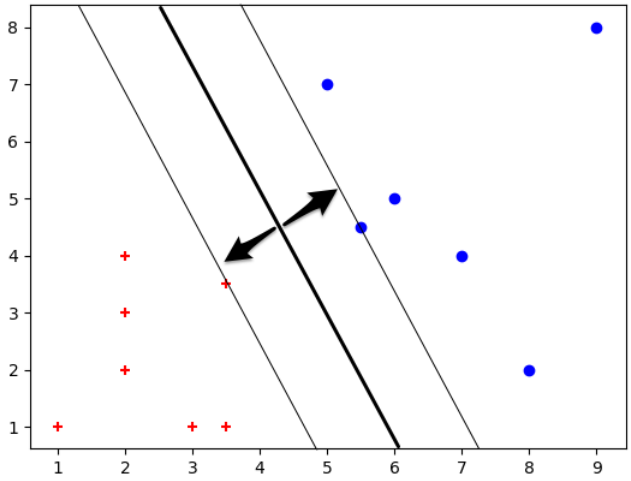


Figure 2.2.: SVM example. Data points of a two dimensional classification problem, separated by a hyperplane.

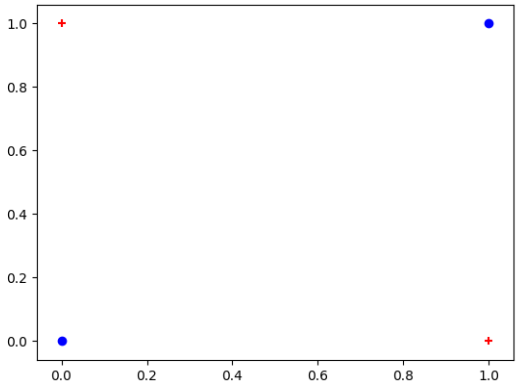


Figure 2.3.: The XOR function exemplifies a non-linear function.

2.3. Naïve Bayes Classifier

Naïve Bayes is a family of probabilistic classifiers based on Bayes Theorem, which states the probability of an event based on prior knowledge. There are different varieties of Naïve Bayes classifiers, but they are all based on the assumption of conditional independence between features. In practice, this means that the classification contribution of a feature is the same regardless of prior knowledge of other features. This could be considered a drawback of the algorithm, as one can usually assume that text features rarely are completely independent of each other. Nevertheless, the Naïve Bayes classifier has proven itself for practical application.

Assume that a document D is represented as vector of features $F = (f_1, ..., f_n)$. The probability that the document belongs to a particular class C_j is given by Equation (2.3).

$$P(C_j \mid D) = P(C_j \mid F) = \frac{P(C_j)P(F \mid C_j)}{P(F)} \tag{2.3}$$

Since the values of F are given at the time of classification, $P(F)$ is constant and can be disregarded. Normally, the chain rule would have to be applied to compute the probability of a feature f_i, given the other features $(f_{i+1}, ...f_n)$ and class C. However, because of conditional independence between features, given a class C, the probability of a feature is simplified, as shown in Equation (2.4).

$$P(f_i \mid f_{i+1}, ..., f_n, C_j) = P(f_i \mid C_j) \tag{2.4}$$

Thus Equation (2.3) can be transformed to Equation (2.5)

$$P(C_j \mid f_1, ...f_n) = P(C_j) \prod_{i=1}^{n} P(f_i \mid C_j) \tag{2.5}$$

Naive Bayes uses this probability model, along with the maximum a posteriori decision rule, which constitutes a likelihood function that chooses the option that maximises the probability. The actual Naive Bayes classifier is thus described by Equation (2.6). The classification y is decided by maximising the product of conditional probabilities.

$$y = \underset{k}{argmax} P(C_j) \prod_{i=1}^{n} P(f_i \mid C_j) \tag{2.6}$$

2.4. Random Forests

Random Forests (Breiman, 2001) is an ensemble learning method, which means that multiple algorithms or model instances, each of which is considered a weak learner, are combined to construct a strong learner and obtain better classification accuracies. As the main idea with Random Forests (RF) is to utilize multiple decision trees, it is helpful with a short review of how these work. A decision tree is a model which predicts the

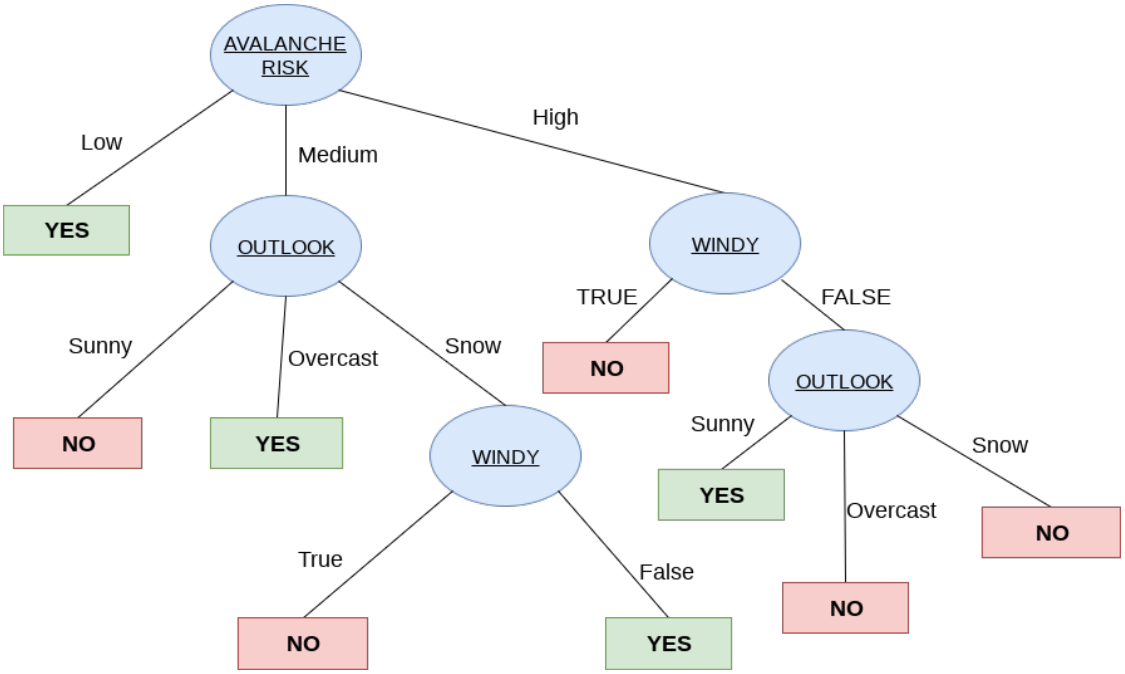


Figure 2.4.: Decision tree for whether or not to go on a ski trip based on weather conditions.

target class based on a number of input attributes. Each interior node corresponds to an attribute, while the leaf nodes can take on the values of possible classification categories. The edges of the tree cover the possible values for each attribute, thus leading to a prediction. This forms a conditional rule-based construct. An imperative part of creating a decision tree is the choice of attributes and the order in which they appear in the tree. Preferably, the number of attributes used to construct the tree should be minimal. The primary idea is to iteratively pick an attribute with a preference for the one which is able to partition the input samples the best way possible with respect to the classification categories. To choose the best attribute, metrics such as Gini Impurity or Information Gain (entropy) can be used. The details of these methods will not be covered here, but simply put they provide a measure for what can be considered the best possible split, conditioned by attributes that have already been picked along the current branch of the tree. Figure 2.4 illustrates how a decision tree may look like. This particular tree is for deciding whether or not to go on a ski trip, based on weather conditions.

Traditional decision trees are deterministic and are particularly prone to overfitting as they can learn irregular patterns when based on a large feature set. The Random Forest method uses the mode of multiple decision trees to reduce the resulting variance to combat overfitting. Random Forests use an ensemble algorithm called bootstrap aggregation (Breiman, 1996), or bagging, to construct the different decision trees. With this method, each decision tree is constructed from a randomly sampled subset of the

training data and each tree gets a vote which contributes to the prediction produced by the Random Forest. In the context of ensemble methods, the decision trees represent the weak learners, and the Random Forest constitutes the strong learner.

2.5. Deep Learning

This section is divided into several subsections as a result of its key position in this project and the level of depth it will be described in compared to other machine learning methods. Following is a brief historical review and a description of the deep learning concept, before the rest of the section describes different types of Artifical Neural Network (ANN) models that are relevant to this project.

2.5.1. Historical Review and Definition

As pointed out by Deng et al. (2014), there are numerous overlapping definitions of deep learning. Based on these definitions, it can be broadly described as a sub-field of machine learning, which utilizes multiple layers of non-linear information processing techniques to learn complex relationships between data and create high-level abstract representations of these. These days, however, as stated by artificial intelligence researcher Michael Jordan of UC Berkeley (Gomes, 2014), the term is very commonly used as a rebranding of ANNs in general.

From a historical perspective, the meaning of deep learning has seemingly changed along with the development of ANNs through the decades. Goodfellow et al. (2016) identified three waves of deep learning development. From the 1940s to 1960s, deep learning was related to cybernetics and the initial development of linear models in the early days of ANNs, e.g., the single-layered perceptron described in Section 2.5.2. The discoveries of Minsky and Papert (1969), concerning the perceptron's inability to solve non-linear problems, led to a temporary stagnation of the development in the field. The research during the 1980s–1990s related deep learning to increased focus on connectionism. Primarily, this concept suggested that a large number of computational units could solve more complex problems when connected together to form a network. Other important concepts that arose during this time period was the backpropagation algorithm, briefly described in Section 2.5.2, and distributed representations of input data, i.e., input should be represented by many different features which in turn are present in a large number of possible inputs. In the modern sense, these characteristics describe an ANN in a nutshell. Simply put, it is a layered network of neurons, which is trained to approximate a classification function by being fed labeled input data for a large number of iterations. The different neurons recognize different features of the input and activate accordingly to contribute to the classification. Largely due to the high computational cost of ANNs, the level of difficulty for training them, and the advent of other simple and efficient classifiers, such as the Support Vector Machine, researchers of the field were yet again discouraged.

The third wave of deep learning is the one we are currently experiencing. As stated by Goodfellow et al. (2016), this renewed appreciation of ANNs began as Geoffrey Hinton published his work on an alternative form of neural network, called Deep Belief Networks (Hinton et al., 2006), showing that they could be efficiently trained using a certain approach. However, this topic is outside the scope of this review. Another important catalyzer of the deep learning development in the most recent decade has been due to larger and faster computational hardware, enabling more complex network structures, in addition to the utilization of GPUs to drastically increase the speed of ANN training when compared to using CPUs. After its resurgence, deep learning has proven itself as a viable approach in various fields, e.g., speech recognition, image recognition and Natural Language Processing, which is a primary focus point of this project.

This subsection concludes with a short discussion about the brain analogy often used to explain ANNs, based on aspects put forth by Goodfellow et al. (2016). While the earliest work within the field may have tried to model the information processing capabilities of the human brain, which consists of a network of neurons, it may be more correct to say that the human brain serves as an *inspiration* for deep learning research. The current lack of rigorous understanding of how the brain functions does not justify stating that deep learning models are supposed to simulate the inner workings of the brain.

2.5.2. Feedforward Neural Networks

The feedforward network was the first type of neural network developed. To develop a sense of basic understanding and intuition for ANNs, this subsection will be slightly more technically descriptive than the remaining sections of other network types. Depending on the architecture, the feedforward network can be of varying complexity. In its most basic form, it is known as a perceptron and was formally introduced by Rosenblatt (1961). The perceptron was the first ANN model that could learn a classification function given input samples from each classification category. The perceptron consists of one neuron, which can take several input values and output a single binary output value, corresponding to the classification, based on Equation (2.7).

$$y = \begin{cases} 1, & \sum w_i x_i + b > 0 \\ 0, & \sum w_i x_i + b \leq 0 \end{cases} \tag{2.7}$$

The input is multiplied with the weights and a bias is added. If the result is above the value of 0, the perceptron activates, or "fires", and outputs a 1. Otherwise, it outputs a 0. One can say that, the activation function of a perceptron corresponds to the use of a Heaviside step function. In practical terms, the weights can be viewed as the importance of each input value, while the bias is a measure of how easy it is for the perceptron to "fire", because larger values increase the chance of activating the perceptron. The weights and bias are learned parameters, which are intizialized to some value at the beginning of training. When passing through training samples, these parameters are updated to

improve the corresponding output. Optimally, a small change in the parameters should result in a small change in the output because then one can gradually close in on the optimal values. Only outputting binary values makes this difficult because a small change in parameters could result in large changes in the behaviour of the network as a whole. For example, the network could improve at recognizing class 1, but become a lot worse at recognizing class 2. Thus, it is more common to use sigmoidal neurons, in the final output, to get smooth output values that can be gradually adjusted by making small changes to the weights and bias. By sigmoidal, we refer the use of a sigmoidal activation function, i.e., a function with a characteristic S-shaped curve as illustrated in Figure 2.5. The term sigmoid function is often used to refer to a special case of the logistic function, as shown in Equation (2.1), but there are other functions, such as tanh and softmax, that have S-shaped curves and can be characterized sigmoid as well. To avoid this confusion, we use the term "sigmoidal" when referring to functions with S-shaped curves in general, and "sigmoid" when addressing the logistic function. The output of a sigmoidal neuron

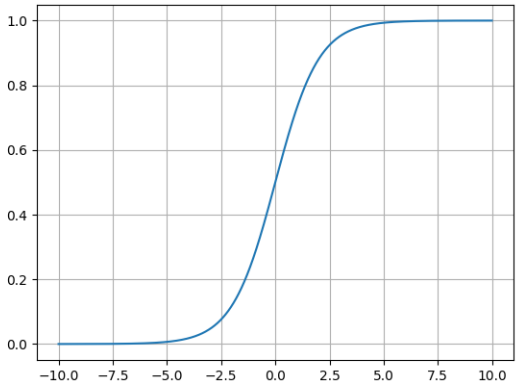


Figure 2.5.: The logistic function is a sigmoidal function.

is shown in Equation (2.8), where σ corresponds to the use of a sigmoidal function.

$$y = \sigma(\sum w_i x_i + b) \tag{2.8}$$

As previously mentioned, the limitations of the perceptron were brought to light by Minsky and Papert (1969), who showed that a single perceptron was incapable of learning non-linearly separable functions, such as XOR. This changed with the development of the backpropagation algorithm (Werbos, 1974), which resulted in the possibility of training Multilayer Perceptrons (MLPs). Such networks contain multiple layers of several perceptrons, or sigmoidal neurons, where each layer of neurons is fully connected to the next layer of neurons. The general construction is as shown in Figure 2.6, with one or more so-called hidden layers between the input layer and the output layer. This allowed for solving the more complex non-linear problems. To update learned weights

and bias, the single-layer perceptron uses the delta rule, a gradient descent based learning rule. Simply put, it computes the update value based on the amount of error, also known as loss, between the desired output and the perceptron's actual output. The backpropagation algorithm is a generalization of this rule, enabling weight and bias updates of multiple layers of neurons. After a training sample has been passed through each layer of the network, the amount of error for each output neuron is computed according to a loss function. The error values are then propagated backwards to each neuron in the network. More specifically, the backpropagation algorithm uses gradient descent, or a variant of gradient descent, which functions as an optimizer to compute the partial derivatives of the loss with respect to each weight and bias parameter. Nowadays, optimizers are based on Stochastic Gradient Descent (SGD). We defer the explanation of SGD to Goodfellow et al. (2016, p. 286). Furthermore, this allows each neuron to gradually correct their weight and bias parameters according to the amount of error imposed. The appropriate loss function depends on the type of activation used in the output layer. For categorical classification, it is usual to use the sigmoid function or the softmax function, when the problem is no longer binary, i.e., there more than two classification categories. The softmax is another special case of the logistic function. The number of output neurons equals to the number of classification categories, and the output values form a probability distribution over the categories.

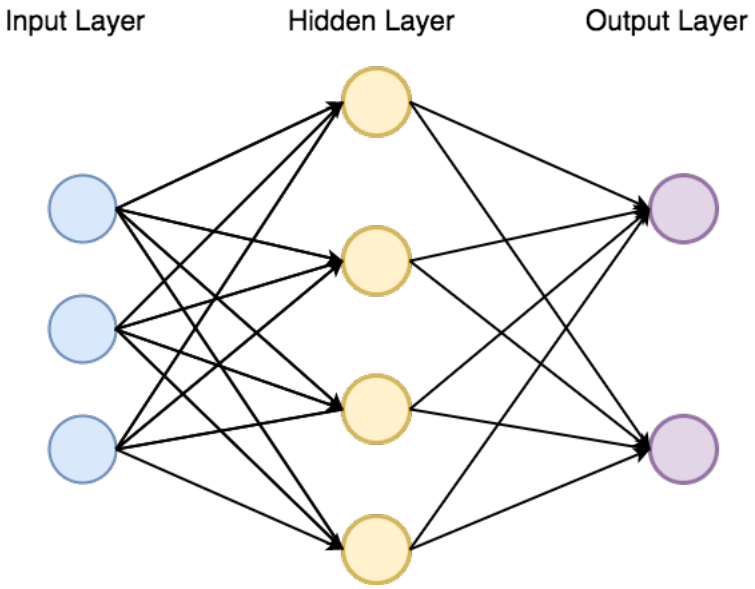


Figure 2.6.: A Multilayer Perceptron with one hidden layer. The two output neurons correspond to the number of classification classes.

The MLP is what one commonly refers to when using the term feedforward network, even though any neural network which is not recurrent (described in Section 2.5.3), can be denoted as a feedforward network. The common characteristic is that the data flows

in one direction, from the input neurons towards the output neurons.

2.5.3. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) hold the properties of regular feedforward networks, in addition to allowing cyclical connections. While feedforward networks assume independence between the input values, RNNs are designed to process sequential information as they possess a memory of previous calculations in the networks' internal state. The output is thus produced by considering the input values as dependent of each other (Graves, 2012). This is especially useful in NLP where the order of words in a text sequence has a direct impact on the semantics of the text as a whole. For example, when creating a text generator by learning to predict the next word in a text sequence, a memory of previous words is of considerable value.

Figure 2.7 shows an example of an RNN with three input units and two hidden units. The cyclical connections in the hidden layer represent the recurrent property of the network in form of memory, as the calculations at each time step are sent to the next time step. If the network is used to process a sequence of words, each time step would correspond to performing the same operations on each word in the sequence, with information about previous words kept in memory.

Figure 2.8 illustrates more closely how to think of the cyclical connections of an RNN as the network is unfolded over time. Each recurrent neuron represents a layer of units in itself. Equation (2.9) illustrates how the hidden state $\boldsymbol{h}^{(t)}$ at each time step is dependent on the previous state $\boldsymbol{h}^{(t-1)}$ and the current input $\boldsymbol{x}$. The hidden state corresponds to the network's memory. $\boldsymbol{\theta}$ represents the parameters that are being learned to produce the best output.

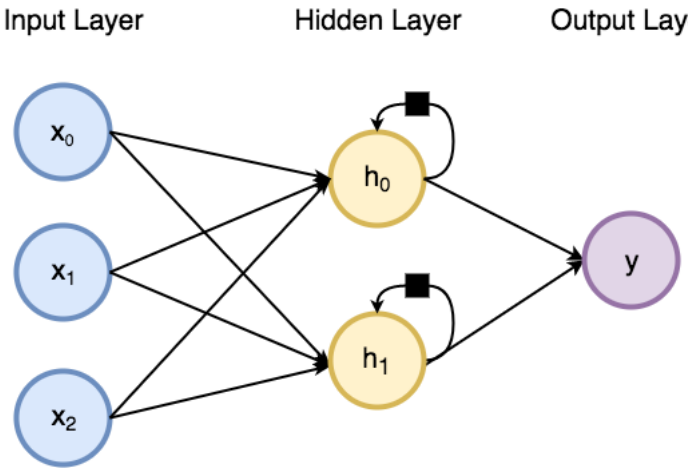


Figure 2.7.: A simple RNN network with two recurrent hidden neurons.

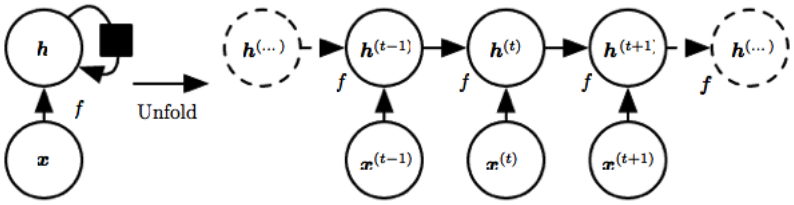


Figure 2.8.: Recurrent network unfolded over time. Image used with permission by Goodfellow et al. (2016).

$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}) \tag{2.9}$$

Unlike in feedforward networks, the loss in RNNs depends on the outputs from the hidden neurons to themselves in the next time step, in addition to the regular forward-moving outputs. Therefore, a modified version of the backpropagation algorithm, called Backpropagation Through Time (BPTT) (Werbos, 1990), is used to accommodate this detail.

As a means for accessing future information to add more contextual value, and to avoid the short-coming of the RNN with regard to assigning too much weight to the inputs last in the sequence, Bidirectional Recurrent Neural Networks (BRNNs) were introduced by Schuster and Paliwal (1997). The general concept consists of feeding the training samples to a pair of separate hidden layers. One layer processes the sequences in a regular fashion, while the other layer is fed each sequence in reverse order. These are then connected to the same output layer, yielding information about past and future sequence values at each time step. This process is illustrated in Figure 2.9.

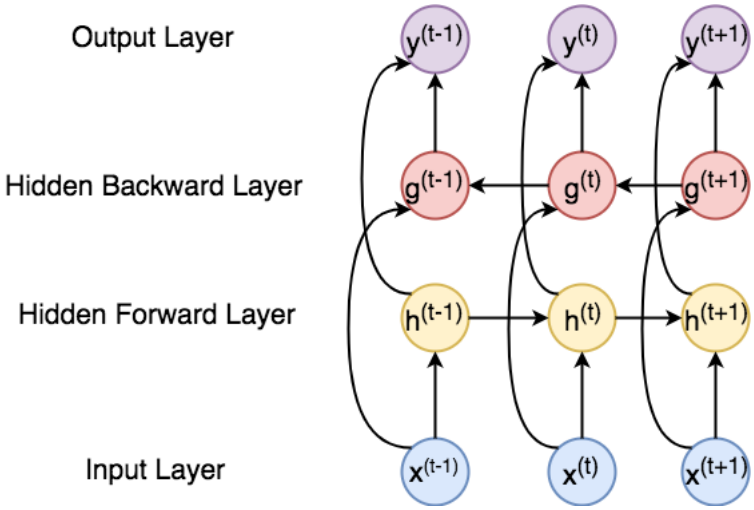


Figure 2.9.: An unfolded BRNN over three time steps.

2.5.4. Long Short-Term Memory Networks

A weakness of the RNNs, described in Section 2.5.3, is that they have difficulties with handling dependencies when the number of layers and time steps increase, i.e., as the length of text sequences increases (Bengio et al., 1994). In such cases, signals from textual terms decay over time. The so-called vanishing gradient problem was formally identified in 1991 as a negative effect on the gradient updates when many small-valued numbers are multiplied together, exponentially decreasing the values as they are passed to the front of the network (Hochreiter, 1991). This results in slow training of certain parts of the network, typically those closest to the input layers. To handle this, the Long Short-Term Memory (LSTM) was introduced by Hochreiter and Schmidhuber (1997). The LSTM is an improvement of the regular RNN as it is more capable of learning long-term dependencies. Practically speaking, this makes the LSTM useful for cases where there is a great distance between words in a text that are contextually related. Related to the vanishing gradient is the exploding gradient, which occurs as numbers of large magnitudes are multiplied together to exponentially increase the gradients, as the name implies. The LSTM architecture is composed of special units called memory

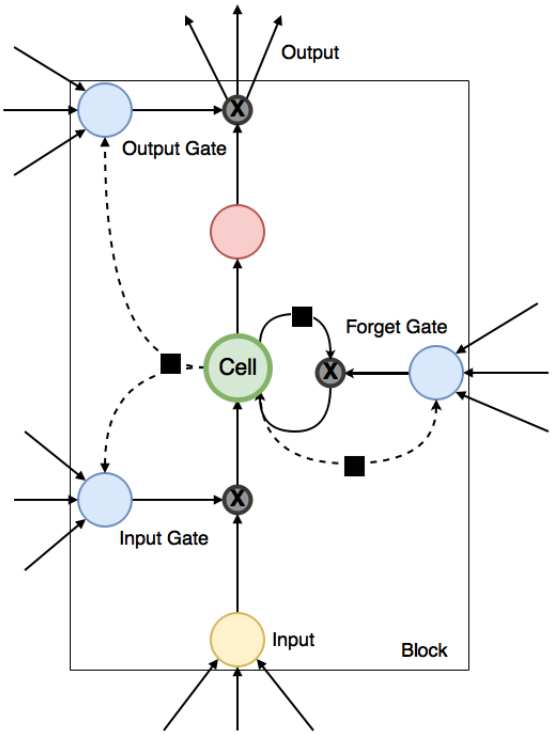


Figure 2.10.: Diagram of LSTM memory block with one cell, illustrating how the gates are connected to the input, output and the memory cell. The dashed lines represent “peepholes”. The final output from the block is the output from the memory cell multiplied with the output gate.

blocks instead of the regular RNN neurons. These blocks contain one or more memory cells, in addition to three multiplicative gates: an input gate, a forget gate and an output gate as shown in Figure 2.10. The memory block uses these units to store temporal states of the network. The multiplicative gates can be either open or closed and are used to control the flow of information through the memory cells. They allow the cells to store and access information over long periods of time. An important feature of the LSTM is that there are no activation functions used on the recurrent aspect of the cells, thereby countering the vanishing gradient phenomena because the values are not iteratively squashed. If the input gate remains closed, the activation of the cell will not be overwritten by any new input and will thus be available for a longer time period. The forget gate effectively controls which input should be stored for later access or forgotten. With regards to a more detailed discussion of RNNs and LSTMs we recommend Graves (2012).

There exist multiple variants of the LSTM, which have been introduced over the years. With each performance increasing model, the LSTM standard has been updated as well. For example, the forget gate was not initially present in the original model. Early on, peephole connections from the the gates to the memory cells were introduced as well, allowing the gates to inspect the cell states (Gers and Schmidhuber, 2000). Quite recently, Gated Recurrent Units (GRUs) were introduced by Cho et al. (2014), essentially merging the input gate and forget gate to a single update gate. The purpose was to produce similar results in a more efficient manner.

2.5.5. Convolutional Neural Networks

The Convolutional Neural Network (CNN) (Le Cun et al., 1989) is a type of feedforward neural network which differs from the traditional variant structurally and computationally. Traditional feedforward networks, with their fully connected layers, are prone to the curse of dimensionality and scale poorly for higher dimensional problems, e.g., high resolution images. Thus, as the number of connections and parameters increase in deep neural networks, they tend to have issues with overfitting and decreased ability to generalize (Siriwardhana, 2016). In addition, the fully connected nature of the layers results in many wasteful connections, which is expensive. The CNN overcomes many of these issues. It is biologically inspired by how the visual cortex of animals are organized and how they respond to stimuli in overlapping areas of the visual field, known as receptive fields. Therefore, CNNs are viable for computer vision and image processing by design. While the RNN makes more intuitive sense for NLP, the literature displays competitive results with CNNs. What follows is a brief explanation of how CNNs work on images, before explaining how it can be used for NLP.

Convolutional Neural Networks for images use the pixel intensities of the images as input. The neurons of the hidden layer are connected to regions of the image, as shown in Figure 2.11 where each hidden neuron is connected to a 5x5 pixel region of the image. These regions correspond to local receptive fields, as mentioned earlier, and are

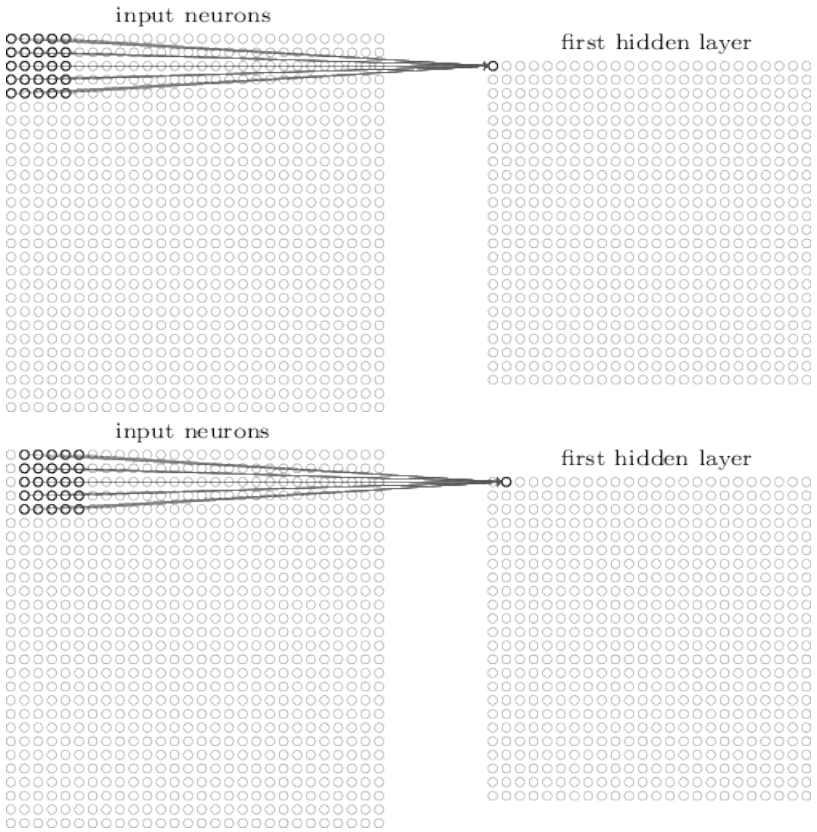


Figure 2.11.: Illustration of how a CNN convolves a region of of input values to a single value. Image from Nielsen (2015). Used under the Creative Commons - Attribution-NonCommercial 3.0 Unported Licence.

commonly called filters. The second image in Figure 2.11 shows how the second neuron is connected to a slightly different region of the image by sliding the receptive field one pixel to the right. The mapping from the input layer to the hidden layer is often called a feature map. In contrast to the traditional feedforward network, the entire hidden layer of neurons has shared parameters, which results in each hidden neuron learning the same feature, e.g., an edge in different parts of the image. Thus, CNNs are invariant to the location of features. This is quite useful for, e.g., classification tasks where we do not care where in the image an object is located, but only its actual presence. By using more filters, other simple features can be learned and later hidden layers can build on simple features from previous layers to learn more complex shapes and objects. To be able to summarize the most important features, it is also common to use a form of pooling layer after the convolution layer. Max-pooling is commonly used and entails selecting the highest values from the convolution. The number of selected values depends on the specified pooling size.

When the input is text instead of image pixels, the representation and structure of the convolutional turns out to be a bit different. Instead of having a receptive field sliding over pixel intensities, we now have words or characters instead, i.e., some form of textual tokens. Typically, the text is represented as a matrix where each row is a token, represented as a vector, and the number of columns is defined by the dimensionality of the token vectors. This vector can be a word embedding produced by, e.g., word2vec, which will be presented in Section 3.5.2, or a one-hot vector. In one-hot representations the size of the vectors equals the size of the vocabulary and only one element has the value of one, while the others are zero. Each index of the vector representation corresponds to a word or character, depending on the granularity, in the vocabulary. Thus the representation satisfies the condition of uniquely representing each token in the vocabulary. This ensures spatial representations of text, which CNNs can process.

Figure 2.12 illustrates how a CNN can interpret the sentence "I like this movie very much!". Each row in the left-most matrix corresponds to the vector representation for each word. The next set of matrices represent filters, with different number of rows. The size defines how many tokens are in the receptive field, i.e., how many words to consider when identifying patterns. In the figure, filters of size 2, 3 and 4 are used. Thereafter, feature maps are developed using the filters, before the 1-max pooling layer uses the highest value of each feature map to summarize the most important features. Pooling serves multiple purposes. It is a way of summarizing a set of values and thus reducing dimensionality and at the same time it offers a generalizing effect. A vector produced by concatenating the pooled values can represent the sentence as a feature vector. The last step in a neural network heavily depends on the application and what sort of output is desired. As previously mentioned, softmax activation is commonly used for classification problems, because a probability distribution over the classification classes can be developed. This is exemplified in the figure as the last step, where softmax is used as activation, with the feature vector as input.

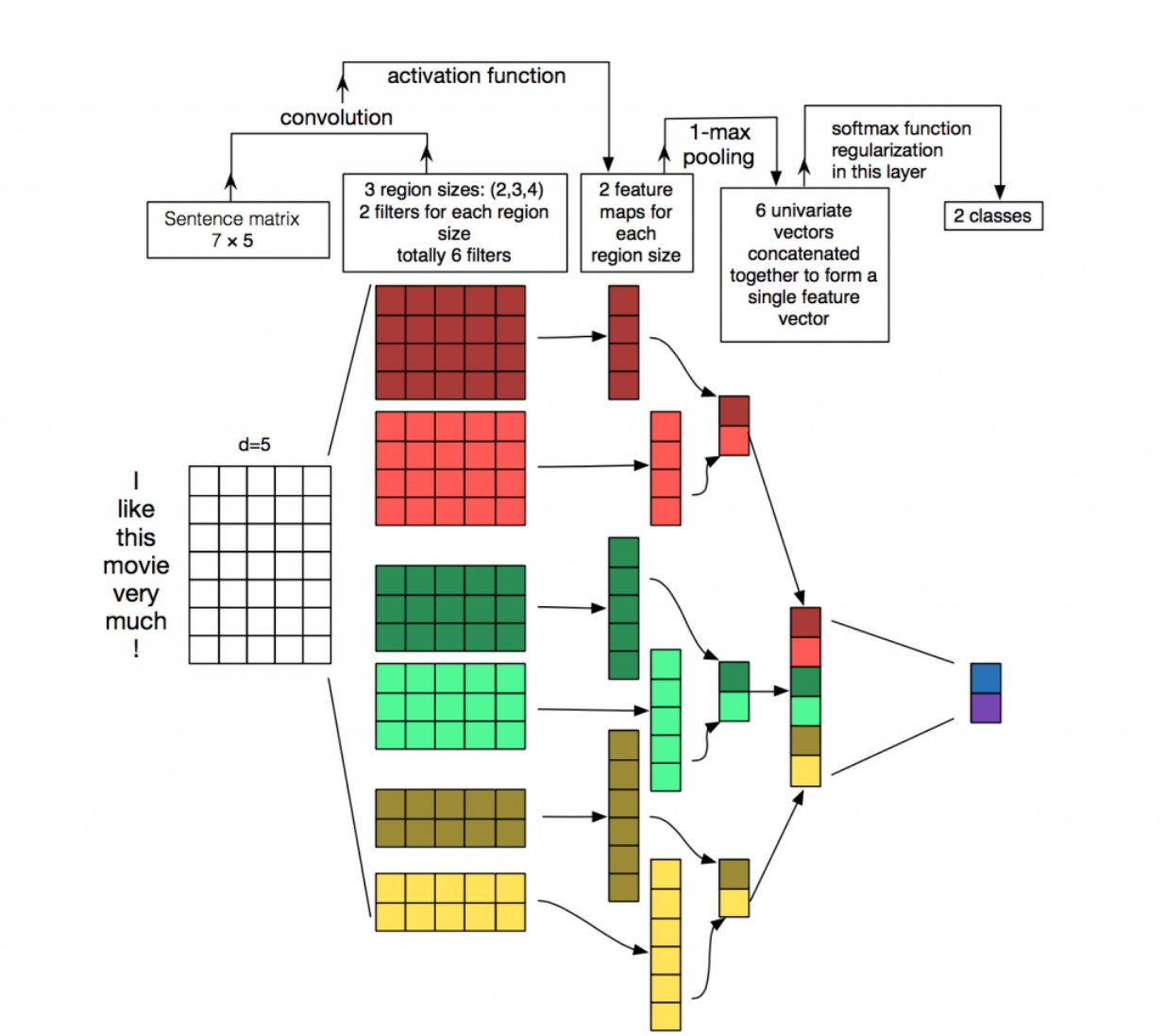


Figure 2.12.: Convolutional neural network for NLP. Image used with permission by Zhang and Wallace (2015).

2.5.6. Autoencoder

An autoencoder is an Artifical Neural Network that is trained to copy its input to its output. This makes the autoenoder a unsupervised learner since it uses the correct output is the input itself. Autoencoders have been in the field of deep learning for decades and was first introduced by LeCun in 1987. The network contains two parts: an encoder and a decoder. The encoder processes the input vector $\boldsymbol{x}$, using a function f, resulting in $f(\boldsymbol{x}) = \boldsymbol{h}$, where $\boldsymbol{h}$ is represented in a lower dimension than the input vector $\boldsymbol{x}$. The decoder attempts to reconstruct the original representation from the encoded representation, using a function g, such that $g(\boldsymbol{h}) = \boldsymbol{r}$. If $\boldsymbol{x} = \boldsymbol{r}$, it means that the autoencoder is able to perfectly reconstruct the input from the produced encoding, though this is not considered especially useful (Goodfellow et al., 2016, p.494). Autoencoders are designed to produce incomplete copies, thus forcing them to focus on extracting the most important features and learn useful properties of the data. This quality makes them suitable for dimensionality reduction and feature learning, but recently they have been useful in generative models as well (Goodfellow et al., 2016, p.494). Usually, we are only interested in the encoder for further use, and the decoder is typically discarded.

Figure 2.13 shows an example of how an autoencoder can be structured. It illustrates an autoencoder with two hidden layers, in addition to the hidden layer representing the encoding. It can also be constructed with only the encoding layer, with no additional hidden layers. The output represents the decoding.

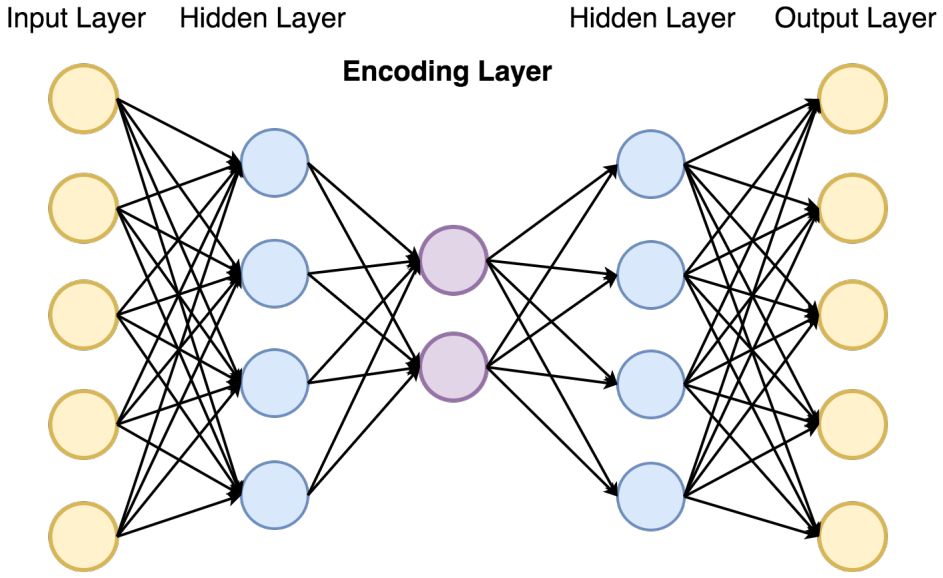


Figure 2.13.: Illustration of how an autoencoder can be structured. The deep autoencoder model contains two hidden layers, in addition the encoding layer. The output layer represents the decoding.

3. Text Representation

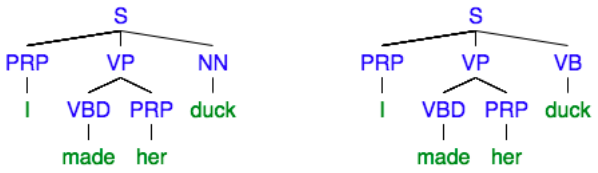
To process text for various purposes, the text needs to be represented in a way that is computationally feasible and interpretable. This chapter covers methods used in Natural Language Processing for this very purpose. The first part of the chapter covers language processing concepts, such as n-grams and part-of-speech tagging, in addition to statistical methods, such as a Term Frequency-Inverse Document Frequency (TF-IDF) and Bag of Words. Thereafter, a review of the neural language model development of the last decade is provided. Often known as vector space models, these are now commonly known as word embeddings in relation to deep learning. The last section introduces the concept of stylometric features with regards to author profiling.

3.1. Part-of-Speech Tagging

Part-of-speech (POS) tagging is the process of categorizing words of a document into a particular part-of-speech, based on the context of adjacent words. The term part-of-speech is also known as word class or lexical category; these word classes can be, e.g., 'nouns', 'verbs', 'adjectives' and 'adverbs'. POS-tagging has several applications in linguistics, e.g., when translating a sentence from one language to another and the grammatical structure of the languages differs. POS-tagging may also be used to resolve word ambiguity where a word may fit into several word classes, unless the context can be derived from the surrounding words. Besides ambiguities in natural language, the performance of a POS tagger also depends on how the tagger is trained on a treebank and what kind of corpus is used. The domain of Twitter poses additional challenges to POS-tagging since users often express themselves in a imprecise way by using slang words and abbreviations. Figure 3.1 exemplifies an interesting case where the correct POS tag is also dependent on the surrounding sentences to provide the context, because multiple tags may be appropriate, even given the surrounding words of the sentence. In the sentence "I made her duck", the word "duck" presents lexical ambiguity as it can refer to both the animal and the action of literally ducking. In this case, the NLTK POS-tagger categorizes the word as "NN", which is the abbreviation for a singular noun, but it could also have been correctly tagged as a verb.

3.2. N-grams

In Natural Language Processing, an n-gram is a consecutive sequence of words (word n-gram) or characters (character n-gram) with a length n, called the window size. When



(a) (b)

Figure 3.1.: Two different parse tree structures for the same sentence, exemplifying lexical ambiguity. The POS-tags are in blue, while the words from the sentence are in green.

$n = 1$, it is called a unigram, which only contains single terms. Similarly, it is called a bigram when $n = 2$ and a trigram with $n = 3$. The principle is the same for $n > 1$, with the n-gram containing n co-occurring elements of a text. N-grams have multiple usages, including spell checking, word breaking and text summarization. In text classification, it has proven to be a viable part of feature extraction, as we will see in Section 4.2.2. Figure 3.2 shows the final feature vectors where each element in each vector represents how many times a word occurs in the specific sentence. The vector indices correspond to a key in the vocabulary shown in Table 3.1 where keys are mapped to every unique word in the total amount of sentences. It is worth mentioning that the Bag of Words model does not consider the order of how words are structured in a sentence, but rather treat all words or tokens as a set of elements.

3.3. Term Frequency-Inverse Document Frequency

Term Frequency-Inverse Document Frequency (TF-IDF) is a method, or perhaps more accurately a metric, which is used for measuring the importance of a term in a document and assigns a weight to the term with respect to the document. This has served Information Retrieval applications such as ranking documents given a query of keywords, but the method has also found its way to text classification.

A term can be any token, such as a word or an n-gram. The higher the value of the weight, the more important is the term with respect to the document. Equation (3.1) shows how TF-IDF is calculated for a term t in a document d. TF-IDF combines the term frequency with the inverse document frequency, which is a measure of how much information the term actually provides with respect to how common the term is across all documents.

$$TF\text{-}IDF(t,d) = TF(t,d) \times IDF(t,d) \tag{3.1}$$

$$IDF(t,d) = \log \frac{N_d}{1 + DF(d,t)} \tag{3.2}$$

The term frequency $TF(t,d)$ is, as the name implies, the number of times the term occurs in the document, while the inverse document frequency $IDF(t,d)$ is calculated as shown in Equation (3.2). N_d is the total number of documents and $DF(d,t)$ is the number of documents d containing term t. Essentially, TF-IDF assigns higher values to terms that occur many times within a small number of documents, discriminating these documents from the rest given these identifying terms. A term is assigned a lower value when it occurs few times in a document or if it occurs a large number of times across all documents. In these cases, the term has no discriminating value for extracting important information about any document. In fact, the inverse document frequency was introduced to counter the effect of assigning very common terms high importance (Spärck Jones, 1972). Stopwords, such as "a", "the" and "was" serve as common examples.

3.4. Bag of Words

Bag of Words (BoW) is a model for representing text or documents as numerical feature vectors. The main part of creating a feature vector using the BoW model is to:

- Create a vocabulary of unique tokens.
- Create a feature vector for every document or text and count the number of times each word occurs in the given text.

To illustrate how the bag of words model works, consider these three sentences:

1. Swimming is very refreshing
2. Ice tea tastes very very good
3. After swimming, ice tea is refreshing

A vocabulary from these sentences is illustrated in Table 3.1, which stores all different words with unique keys.

$$\begin{aligned} 1 &= \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \\ 2 &= \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 2 \end{bmatrix} \\ 3 &= \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \end{aligned}$$

Figure 3.2.: Sentence 1, 2, 3 as feature vectors.

Key	Word
0	after
1	good
2	ice
3	is
4	refreshing
5	swimming
6	tastes
7	tea
8	very

Table 3.1.: Bag of words vocabulary.

3.5. Word Embeddings

In 1957, the British linguist John Rupert Firth famously coined the saying "you shall know a word by the company it keeps", referring to how the semantics of a word depends on its context. This is relevant for the topic of word embeddings as it describes the concept in a nutshell. Word embeddings revolve around mapping linguistic terms from a space with an infeasible high dimension, as with one-hot vectors, to a continuous vector space with much lower dimension. Thereafter, the embeddings can be used for other tasks, such as classification problems. This research area falls under the general area of distributional semantics, which is based on the notion of the Distributional Hypothesis (Harris, 1954), stating that that linguistic terms with similar distributions in text have similar meanings. This section will give a short historical introduction to word embeddings, before describing a few of the leading embedding methods.

3.5.1. A Historical Review of Word Embeddings

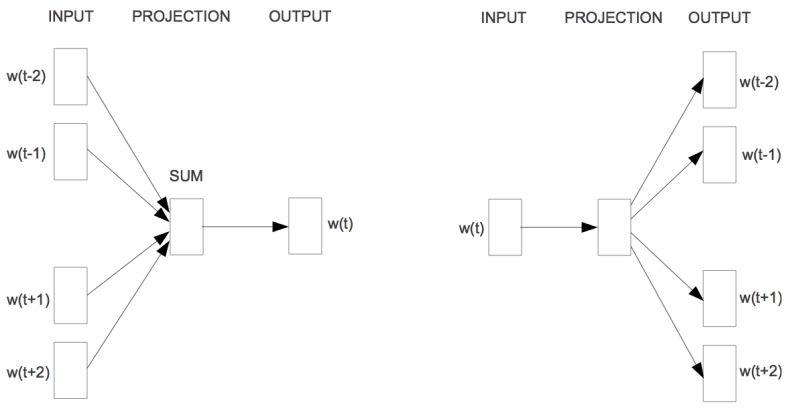
Studies within distributional semantics have been ongoing since the end of the 1980s with Latent Semantic Analysis (LSA) (Deerwester et al., 1990). While this approach mainly focused on document level representation, the new wave of distributed linguistic representations in the 2000s presented studies of word-level representations. This started off with Bengio et al. (2003), who coined the term "word embedding" as they used a feed-forward network (see Section 2.5.2) to learn distributed low-dimensional representations of words. With the addition of faster computers, the quality of word embedding research improved as well. An issue with the previously mentioned work was the computational complexity of the softmax function (see Section 2.5.2), which requires one neuron, in the output layer, per word in the vocabulary. This poses an issue when the vocabulary size is large. Collobert and Weston (2008) improved upon the work of Bengio et al. by avoiding the use of the expensive softmax function and rather using an alternative loss function. This work produced promising results with properties characterizing word embeddings today, e.g., semantically and syntactically similar words occurring close to

each other in vector space. However, word embeddings did not gain serious traction before Mikolov et al. (2013a) introduced word2vec.

3.5.2. Word2vec

Word2vec (Mikolov et al., 2013a) is an unsupervised word embedding technique which builds on previous work mentioned in Section 3.5.1, using feedforward networks (see Section 2.5.2) to train word embeddings. The following description will serve as a way to build intuition about the process, rather than going into the overly technical details. Mikolov et al. (2013a) further lowered the computational cost of word embedding training, compared to Collobert and Weston (2008), by removing the non-linear hidden layer altogether, replacing it with a linear projection layer, though some literature often refer to this layer as a hidden layer. Thus, this method, which has proven to be quite effective, is surprisingly shallow, i.e., not a deep neural network with many non-linear layers. It is also window-based, which will be illustrated momentarily. The original paper proposed two different, but related, architectures for training word embeddings. Both are illustrated in Figure 3.3.

- Continuous Bag-of-Words (CBOW): Based on the context words, i.e., the surrounding words within a specified window-length, the current word is predicted. The window-length specifies the number of words to consider before and after the current word. The model is named after the Bag of Words model (see Section 3.4) because it does not care about the order of the words; only its presence within the window-length. It is continuous as the vector consists of continuous real values.
- Continuous Skip-gram: Similar to CBOW, but given the current word, the surrounding context words, within a specified window-length, are predicted.



(a) CBOW (b) Skip-gram

Figure 3.3.: Architecture of CBOW and the Skip-gram model. Figure used with permission by Mikolov et al. (2013a).

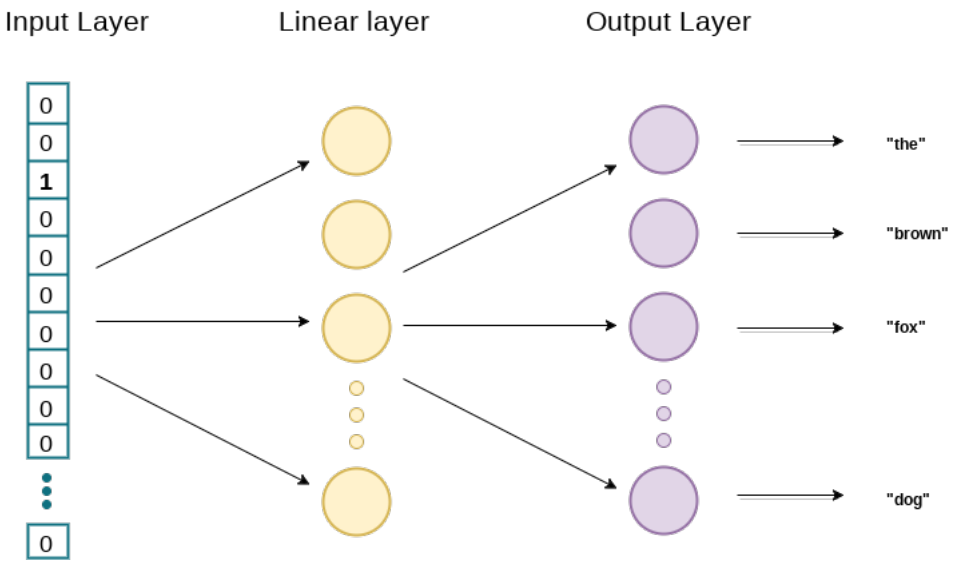


Figure 3.4.: The word2vec Skip-gram model. The number of neurons in the input layer is equal to the number of words in the vocabulary. It is is represented as a one-hot vector in this figure. The number of neurons in the linear layer corresponds to the desired number of features in the words embeddings. The output layer consists of probabilities for each word in the vocabulary, in the context of the current word.

For the sake of brevity, and because the models are similar, only the Skip-gram variant is considered in the rest of the description of word2vec. Essentially, the model consists of an input layer, one linear layer and an output layer, all of which are fully connected. The input layer takes a word as input, in the form of a one-hot encoding. Thus, the input is the same size as the number of words in the vocabulary. This is illustrated in Figure 3.4. The output layer uses the softmax activation function and consists of a neuron for each word in the vocabulary as well, representing the probabilities of these words occurring in the context of the current word. Even though the network is trained to predict context words, the output layer is disregarded at the end of training. The word embeddings are then extracted by retrieving the learned weights of the linear layer. This layer consists of an $n \times m$ weight matrix, where n corresponds to the number of words in the vocabulary and m is the number of neurons in the linear layer. The number of neurons in the linear layer corresponds to the dimensions of the word embeddings, which practically means the number of features in the embeddings. Typically, the dimension size is tuned for the desired complexity and results. Since one-hot vectors are used as input, each word has its own corresponding column in the weight matrix of the linear layer, representing the word embedding. This is a result of all but one value being zero in the input and the resulting multiplications with zero in the linear layer.

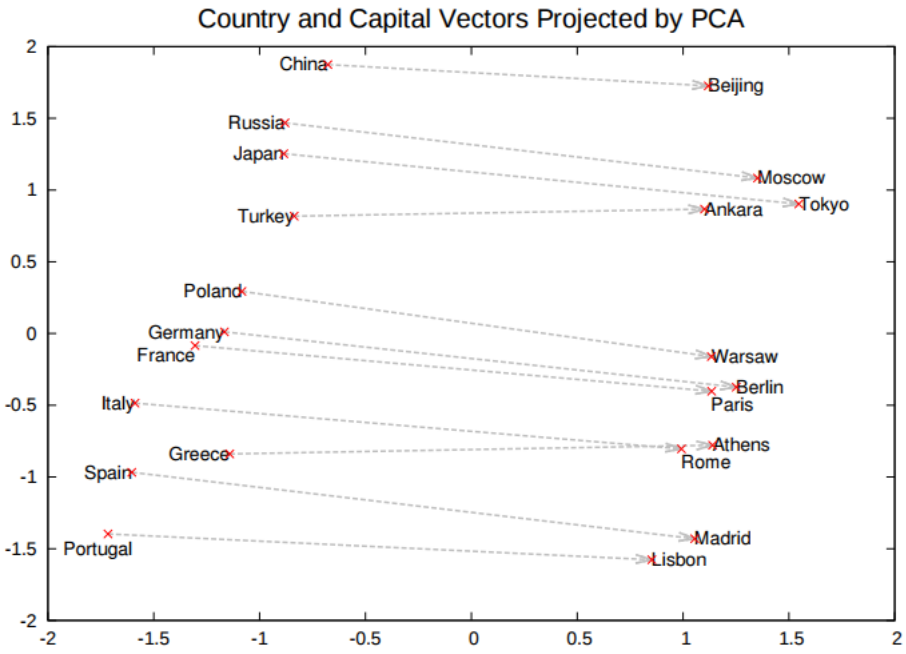


Figure 3.5.: Word2vec embeddings of different countries and capitals in vector space. This illustrates how semantically and syntactically similar words are assigned similar embeddings. Figure used with permission by Mikolov et al. (2013b).

As described, word2vec explicitly trains the network to recognize the context of words, which results in the impregnation of syntactic and semantic information in the embeddings. Since similar words are assigned similar probabilities of contextual words, their word embedding vectors will also be similar. This is reflected in vector space, where similar words will be located close to each other, as illustrated in Figure 3.5. An issue with the "vanilla version" of word2vec described in this section is that it is still computationally expensive, with the fully connected layers, if the vocabulary is large. This results in a large number of weight parameters. To handle this, Mikolov et al. (2013b) followed up with improvements not long after the original paper, to increase the quality of the embeddings and the speed of training by the use of subsampling and negative sampling. However, this will not be covered here.

3.5.3. Global Vectors (GloVe)

Before concluding this section, another embedding method will be briefly described. Inspired by the development in the vector representation of words, such as the work done by Mikolov et al. (2013a), GloVe (Pennington et al., 2014) was developed at Stanford. The motivation was making the properties of the embedding model more explicit by com-

bining traditional statistical methods used in methods such as Latent Semantic Analysis (Deerwester et al., 1990), and the more recent approaches exemplified by word2vec. Specifically, GloVe utilizes the statistical method matrix factorization and the idea of context windows used in word2vec. Arguing that word2vec does not utilize statistical information about the document corpus, GloVe makes use of this in the form of global co-occurrence counts of words. The co-occurence value of a word w_1 with respect to another word w_2 is defined as how often w_1 appears in the context of w_2, within a specified context window size. The method then proceeds by training on the non-zero elements of the co-occurrence matrix, in contrast to word2vec where the input consists of one-hot vectors which are sparse by nature. GloVe has been described as a prominent method alongside word2vec. Compared to neural network models, such as word2vec, GloVe exemplifies a method which is more transparent with respect to interpretation, while still capturing meaningful semantic substructures in the embeddings.

3.6. Stylometric Features

Within the field of authorship studies, stylometry is the statistical analysis of variations in writing style between authors.[1] An example of such style is the frequent use of emoticons or a particular part-of-speech. Many stylistic aspects are deliberate by the author, while there may also be subconscious elements in play. By identifying specific stylometric features of different authors or groups of people, it can be easier to distinguish between them based on their produced text. This section serves as a basic introductory passage, briefly touching upon various stylometric features that have been used in the literature, as summarized in various surveys on authorship analysis techniques (Reddy et al., 2016; El Bouanani and Kassou, 2014). Some aspects will be further described and discussed in Section 4.2.2, where the state-of-the-art in feature engineering for author profiling purposes is described.

3.6.1. Lexical Features

Lexical features are derived from considering a text as a series of sentence producing tokens, where a token can be a word, punctuation mark or number. The advantage of lexical features is that they only require a tokenizer and can be applied to any corpus in any language (El Bouanani and Kassou, 2014). Simply put, the features of this category revolve around the author's preferences with regards to words, which are depicted by the text's vocabulary. Many typical lexical features are based on simple concepts, e.g., total number of words, frequency of words, and average length of words. Different features aid in identifying different linguistic traits of the author. Traits as vocabulary richness can be measured with features like the number of words only appearing once or the ratio between the size of the vocabulary and the total number of tokens in the text. However, the value of measures such as these depends on the length of the text. This is due to the difficulty in getting a good measure of vocabulary richness if the text is short. Other

[1] https://en.oxforddictionaries.com/definition/stylometry

traits, such as an author's idiosyncrasy can be measured by various features like spelling errors (Koppel and Schler, 2003) and the use of slang. The latter has, not surprisingly, proven to be useful in profiling (Prasath, 2010).

3.6.2. Syntactic Features

As the choice of words are context-dependent with regard to topic, the length of words tend to vary from one text to another, as do the length of texts in general. Thus, the lexical features can vary, even in texts by the same author, which can lead to some unreliability. On the other hand, authors tend to have particular syntactic styles which reappear in their texts and this can prove to be a more reliable indicator of authorship than lexical features (Stamatatos, 2009). However, acquiring syntactic features can be more challenging as they are language-specific and require accurate tools. Function words have been shown to be useful in representation of syntactic features as they are used to structure sentences. A simple way of acquiring syntactic information is to use a part-of-speech tagger and compute frequencies of different types of function words, but for more useful features it is necessary to study how words are combined to produce sentences.

3.6.3. Structural Features

The structural composition of texts can provide useful information about the author as people can have individual preferences with regard to, e.g., paragraph composition. Examples of features that have been explored in the literature are average length of sentences, total number of paragraphs, number of sentences per paragraph, and average length of sentences in terms of characters (Reddy et al., 2016). The intuition behind using features like these lies in being able to infer if the author's style consists of complex or simple sentences and text structure. It has been shown that such features can be used to discriminate between gender and age groups. The set of possible structural features is large and, with the substantial growth of text in social media, further exploration has been done in the use of features such as the number of URLs and HTML tags, and the number of emoticons. In regards to tweets, domain specific features have been used, such as the number of hashtags (Giménez et al., 2015) and mentions.

3.6.4. Content Specific Features

The idea behind using content specific features for classifying, e.g., gender is that individuals of the same gender tend to have similar interests. Thus, features related to the topic of the text could aid in classification of gender (Reddy et al., 2016). Clearly, this is a feature that generalizes individuals based on gender, but it may carry factual value. This also applies to age as people tend to write about different topics in different stages of life, which is naturally related to age. People in their teens or twenties tend to write about their life related to school and studies, while people of older age groups, who may be married and have children, tend to write more about things related to this. One way

of acquiring such features is to define content specific keywords and study the frequency of these in text.

3.6.5. Semantic Features

Studies show that various authorship classifications can be improved by using semantic information combined with other features (Stamatatos, 2009). A common method is mapping semantically similar words to concepts with basis in the assumption that words with similar meanings will appear in similar texts. Methods such as LSA (Deerwester et al., 1990) and word2vec (Mikolov et al., 2013a) are used to construct vector representations of text, where semantically related words are close in vector space. Word2vec will be explained further in the next section. Removal of stop words, i.e., words that contribute little discriminative value, has been a common method to reduce noise. However, with regard to analysis of tweets, it appears that removing stop words could have negative effects on the classification accuracy (Saif et al., 2014).

4. Related Work

This chapter will present a review of gender-based linguistic research. In addition, a study of the current state-of-the-art approaches in author profiling will be presented, with emphasis on gender prediction.

4.1. Studies on Language and Gender

To get a broad overview of the development in gender-based author profiling, this section will present research from as far back as the 70s and move on till present day. This way one will get an impression of how the state of society has possibly influenced language and how some linguistic aspects and observations may have changed over time. This section will mainly focus on linguistic elements identifying gender and sociolinguistic aspects connected to them, while leaving out processing methods and classification models for later sections.

4.1.1. Early Studies

Credited to Wayne Dickerson, the word "genderlect" was coined by Kramer (1974). The term is a based on the words "gender" and "dialect" and describes the possibility that men and women may have a particular gender-based style of speech. The relationship between language and gender has been extensively studied over the years. Most of the early work focused mostly on spoken language, but based on the observations they could very well apply to and be useful for analyzing written language as well. *Language and Woman's Place* (Robin Lakoff, 1973) is often regarded as a highly influential publication, which put attention on how men and women express themselves differently. This article mainly focused on how inequality between men and women is reflected in their use of language. Lakoff stated that women experience linguistic discrimination through the way they are taught to use language and that it is caused by a bias in our culture against women as rational individuals. Girls are taught to talk like "ladies" from an early age and are criticized as unfeminine if they express themselves differently. This is exemplified through the choice of certain word types by men and women. Men tend to use a limited set of color-words, while women's vocabulary of colors is more nuanced, e.g., purple, lavender, mauve. Another example is the use of swear-words. In a situation where a man might say "shit", Lakoff claimed that it was more acceptable for a woman to say something like "oh, dear". This is viewed as an indication of men being allowed to express themselves more strongly, while women are restricted to weaker expressions of feelings, reinforcing men's position of strength.

On the aspect of syntax, Lakoff asserted that women make more frequent use of hedges, words used to lessen the impact of an utterance ("kind of", "it could be that"), and the closely related phenomena of tag questions, which are questions that consist of a statement with a certain word sequence, the tag, at the end, turning them into questions. For example, "You're John" is a normal statement, while "You're John, aren't you?" is a tag question. Lakoff related this to how women are imposed this form of insecurity during childhood. Additionally, it is noted that women in general are more polite and make more use of intensifiers, such as "so", e.g., "That is so sad". As the publication is more than 40 years old, the relevance of the particular examples could be questionable, because they relate to the situation in the US at that time. Additionally, little statistical evidence is provided for some of the presented points. Lakoff does in fact assert that her data is mostly collected through introspection and her claims are mostly based on her own intuition. The takeaway is rather the peculiar dissimilarities that could be present in modern usage of language by males and females, and that they could be occurring from an early age.

Haas (1979) provided a literature review of the studies made in the 70s, where many of Lakoff's points reappear, e.g., that men are more prone to using indecent language and women tend to use more intensifying adverbs. Additionally, the difference in preferred topics is addressed. The study upheld the stereotype that men talk more about topics such as business, politics, sports, cars and legal matters, while women have a preference towards social life, family, books, food and lifestyle. Bodine (1975) made an important remark about specifying language features of each gender. She pointed out that there is a distinction between gender-exclusive and gender-preferential features, i.e., it is hard to find features that can exclusively identify gender, because even though some linguistic styles are exhibited to a greater degree by women, it does not mean that a man saying, e.g., "Oh my god, that wall is terribly aquamarine" is an impossible occurrence. It is more correct to say that some linguistic styles occur more commonly with one of the genders. Such overlap of individual linguistic peculiarities, makes identifying gender-specific characteristics more troublesome.

It is worth pointing out that, as with most things, linguistic features are largely dependent on the period of time and the culture. The way people speak and what is regarded as normal can change over time as certain styles can become more acceptable with either gender. Based on the sentence "The wall is mauve", Robin Lakoff (1973) stated that "(...) if the man should say [the sentence], one might well conclude he was either imitating a woman sarcastically, or a homosexual, or an interior decorator." Had the material been contemporary, it could have been deemed a rather strong reaction to what seems like a plausible sentence for either gender at the present time.

While Lakoff (1973) mainly focused on the spoken language, the differences in written language between males and females started to become a topic of interest a few

years later. Mulac et al. (1990) examined impromptu essays by primary- and secondary school students, specifically fourth, eighth and twelfth graders. This was to investigate dissimilarities and, if present, see if they adhered to gender role stereotypes. At the time, these were regarded as men being perceived as more confident, dominant and aggressive. Women were more affectionate, sympathetic and literate. In the study, 19 features were used and their capabilities of differentiating between genders were explored. Based on these features, Mulac et al. were able to accurately classify the gender of approximately 85% of the students overall. However, only some features were consistent across all age groups. Boys were in general more judgmental and girls made greater use of filler words ("Well...") and phrases related to emotions. For primary school students, female writing style showed greater sense of literacy, with more elaborate sentence compositions by use of, e.g., adverbs, while male writers wrote more informally with greater amount of contractions ("didn't" instead of "did not"). Females seemed to soften their sentences more with hedges supporting the notion of greater politeness and less dominating behavior, as also proposed by Robin Lakoff (1973). Additionally, Robin Lakoff and Mulac et al. seem to agree on the early occurrence of gender-linked linguistic differences. The set of differentiating features consisted of other variables for secondary school students than for primary school students, indicating a correlation with age when classifying gender. This adds an extra layer of complexity to the task, as it means that a linguistic feature of gender does not necessarily persist as a marker across age groups. In accordance with Bodine (1975), Mulac et al. (1990) also remarked on the fact that discriminating features were sex-preferential, rather than sex-exclusive.

4.1.2. Modern Studies

As presented, early studies on language and gender were predominantly on speech and informal writing. Argamona et al. (2003) investigated the phenomena in formal texts. They analyzed a corpus containing both fiction and non-fiction, identifying lexical and syntactical differences with regard to gender. It was found that females seemed to make more use of negations, e.g., 'not', and pronouns, while males had a larger amount of determiners, phrases that provide contexts to nouns, such as 'a', 'the' and 'several'. This was viewed as an indication of women being more "involved" in their writing, while male-authored texts had linguistic features characterized as "informational". Especially the more frequent use of first and second person pronouns by female authors could indicate a desire to form a relationship with the reader. The greater use of determiners and quantifiers suggested that males provide more specification in their texts. In a related publication by Koppel et al. (2002), it was revealed that the inclusion of both fiction and non-fiction as genres had an impact on the classification accuracy. Training on both fiction and non-fiction together resulted in a lower classification accuracy of a test set containing fiction, than when training only on a dataset containing fiction. The opposite seemed to be true as well. Additionally, training purely on fiction and then testing on non-fiction resulted in an accuracy of 50% which is the same as random guessing. This exemplifies another layer of complexity for classifying gender. As style of writing can change with respect to genre, the linguistic markers do not necessarily persist across

genres.

A new domain opened up for studying language and gender at the beginning of the 21st century. With the increased popularity of blogs, an abundance of public data became available. Schler et al. (2006) retrieved all blogs available at blogger.com one day to put together a corpus containing 1.4 million blog entries from 37,500 blogs, adding up 295 million words. In comparison, Argamona et al. (2003) used a corpus containing 600 documents, comprising 25 million words, and Mulac et al. (1990) had only 96 essays in their possession. With the introduction of the social media genre, new challenges and characteristics appeared as well. Blog posts are usually shorter in nature, compared to formal publications. Additionally, the language can be quite informal with the use of emoticons and blog words/Internet slang, such "lol" and "u" ('you'). These types of words were used as features, along with hyperlinks, and it turned out that women made more frequent use of blog words, while there was a higher occurrence of hyperlinks in posts by men. Supporting the findings of previous literature not based on social media (Argamona et al., 2003), females made more use of pronouns and negations, while males made greater use of articles, a sub-group of determiners, and prepositions. These findings also cohere with the previously mentioned aspects of men being more informational in their writing, as women appear more involved and personalize their texts. Putting aside these style-based features (see Section 3.6), the content was found to also support these observations and some stereotypes about gender. Men seemed to write more about politics and technology, whereas women favored topics related to family, relationships and lifestyle.

More recent studies have made similar observations to the previously mentioned research. Garimella and Mihalcea (2016) also analyzed blogs, to identify semantic and psycholinguistic word classes that characterize each gender. The word classes used in their study were composed of multiple words related to the broader category. They found that some of the dominant word classes for women were sweetness, touch, sourness and texture, interpreted as women's greater sense of world perception. In addition, there were the word classes related to family and lifestyle. Top word classes for men were those related to religion, sports, science and work. This study also went more in-depth to identify differences in semantic usage by performing a gender-based word sense disambiguation, i.e., they looked into if polysemous words, words with multiple meanings/senses, were used by men and women differently in terms of the chosen sense. However, the results indicated that the chosen word senses were mostly similar.

4.2. State-of-the-Art

PAN is an annual series of shared tasks on digital texts forensics. The name stems from the early days, when it was an abbreviation for "International Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection". In recent years, this has changed to "International Workshop on Uncovering Plagiarism, Authorship, and So-

cial Software Misuse", though the workshop is commonly simply known as PAN. Since 2007, they have hosted numerous tasks within plagiarism detection, author identification, author profiling, and vandalism detection. Relevant to our project is the author profiling task, which has been hosted four times since 2013. The topics have mostly been gender and age. The workshop motivates research in this field and is likely to have accelerated the development and performance of such systems.

This section describes the state-of-the-art within author profiling, with focus on gender classification of tweets. Most work done in this field appear to follow a general systematic approach, which can be structured into four procedures:

- data collection
- pre-processing
- feature extraction
- model building

Data collection is the process of retrieving and annotating texts to construct a dataset for training and testing a profiling system. The datasets used in this thesis will be described in Chapter 5. As we did not perform any annotation of data, and the datasets used were pre-annotated, data collection will not be further described in this section. Textual pre-processing consists of preparing and cleaning the text by removing or replacing words and characters that presumably contain no valuable information for inferring gender and may cause obstruction. Feature extraction is the task of extracting useful information from the text and creating a numerical representation, which can be fed to a classifier. In recent studies, this step varies a lot between researchers, some use standard natural language processing techniques such as part-of-speech (POS) tagging or Bag of Words (BoW), others create their own heuristics or embeddings that fit the dataset. The last step is developing a classification model. Machine learning is commonly used for the task. To the extent of our knowledge, there are no known approaches for bypassing the need of a labeled dataset. Therefore, supervised learning seems to be the only used method.

4.2.1. Pre-Processing

Pre-processing can be viewed as various steps taken to prepare the text for optimal extraction of features. This is done by structuring and filtering the text by removing parts of the text that can be viewed as noise. Pre-processing is an important first step and can be viewed as the foundation to the system as it has a direct impact on the value of the features used to represent the text and the performance of the model. Burger et al. (2011) kept the pre-processing simple, with tokenization to separate words with non-alphanumeric characters as separators. This could be because of their use of only simple n-gram features from the tweet text itself. Recent work from the PAN workshop shows that participants make more elaborate approaches to pre-processing the text that

are quite similar to each other. Due to how the dataset is made available by PAN, which is in the form of XML files containing tweets surrounded by HTML tags, participants usually clean and remove these markup tags in order to obtain the plain text (Ashraf et al., 2016; Bilan and Zhekova, 2016; Liebeck et al., 2016). From an intuitive perspective, working with plain text is favourable as the markup tags are not actually parts of the tweet content. Lemmatization and stemming were used by Bougiatiotis and Krithara (2016) and Deyab et al. (2016), but did not lead to an improvement in the classification results. A reason for this could be that tweets are short in nature and limited discriminative information is available. Thus, if the use of, e.g., specific verb tenses could be indicative of gender and have a positive impact on classification, lemmatization and stemming will generalise these words and hence eliminate this positive effect. On the other hand, these methods could contribute positively depending on the term space. If the vocabulary of word embeddings is limited, which was not the case for the previously mentioned authors, it could be more valuable to have an embedded word in its root form rather than not having it at all.

Other pre-processing methods include removing punctuation marks and numbers before lowercasing the whole text (Bougiatiotis and Krithara, 2016). Removing stop-words is typical (Deyab et al., 2016), with the supporting argument that they are too common to be indicative of gender. The majority of today's solutions replace unique URLs and Twitter specific terms, such as mentions and hashtags, with common identifying tags. Markov et al. (2016) additionally replaced numerical digits and distinguished between URLs referring to pictures and textual hyperlinks. Table 4.1 exemplifies this technique. The underlying argument is that specific URLs and numbers do not help classify gender, and that it is more advantageous to only capture the occurrence of these. Additionally, if word embeddings are used, the vocabulary can be further limited to a computationally feasible size. Liebeck et al. (2016) chose to altogether remove the previously mentioned terms, i.e., URLs, hashtags and mentions. They reasoned with keeping the classifier genre-neutral, i.e., not overfitting to tweets, for the purpose of being able to classify texts from other social media as well, though one could say that these terms are likely to appear in other social media. On the other hand, the authors claimed that hashtags and mentions may be useful for differentiating genders. Therefore, it could be interesting to investigate the classification value of terms, such as hashtags and mentions, which has not been extensively explored in today's solutions.

Twitter Specific Syntax	**Replace With**
#hashtag	#
@mention	@
www.youtube.com	URL
pic.twitter.com/vYpLShlHs7	PIC

Table 4.1.: Replacement of twitter specific syntax

4.2.2. Feature Extraction and Representation

Before a document can be fed into a classification model, it needs to be represented somehow. As a computational convenience, this frequently means creating a vector representation of the document, which the classification model can work on. Optimally, this vector would contain the key features for successfully classifying all texts. Section 3.6 describes various types of stylometric features that have been used in the literature. Following is a review of features used by some of the best performing teams of PAN Author Profiling 2016, representing state-of-the-art. The features used by each paper are not thoroughly described, for the sake of brevity, as there is some overlap.

Most work on author profiling seem to represent documents by using a combination of stylometric features and one can see some level of conformance with early studies. Among other features, op Vollenbroek et al. (2016) used frequencies of function words, part-of-speech, average word and sentence length, capital letters, and proportion of unique words across all documents of a user to measure vocabulary richness. Reflecting the age of social media and the increased use of Internet language, the proportions of emoticons and out of dictionary words, i.e., slang and misspelled words, were also used. Frequencies of n-grams seem to be common. Different forms of n-grams can characterize different aspects of a document, e.g, word n-grams for content and part-of-speech n-grams for writing style (op Vollenbroek et al., 2016; Liebeck et al., 2016). With regard to the importance of each feature, it seems that it is usually the combination of features that results in the discriminatory power and thus it is difficult to assess the individual contribution. Based on work done by Lopez-Monroy et al. (2013) from PAN 13, second order attributes were explored by op Vollenbroek et al. (2016) and Bougiatiotis and Krithara (2016). The concept is slightly convoluted computationally, but the main idea is creating profiles based on terms/tokens that are more common for a gender and representing documents as a vector where each value contains its relationship to each profile. Thus, the occurrence of some terms counts more towards a gender. As mentioned, Burger et al. (2011) only used n-gram features from the tweet text itself, but they also tried utilizing any available information from the Twitter profiles for comparison, such as full name, screen name and user descriptions. The study concluded that (not very surprisingly) the full name and screen name were most informative, but interestingly enough the tweets were more gender descriptive than self-written user descriptions.

Similar to word embeddings (see Section 3.5.2), there have been efforts to create embeddings of tweets based on the raw text content, although these have not been used for author profiling. Dhingra et al. (2016) constructed a character-level neural network, taking raw text as input, for classifying hashtags of tweets. Similar to one-hot word vectors, the characters were processed as one-hot character vectors. They used Bidirectional Gated Recurrent Units (GRU; similar to an LSTM) in their model. By being bi-directional, the input was processed in reverse as well. The idea, as described in Chapter 3, is that each textual element is connected to both previous and future text in the sequence, and bidirectional networks are better capable of capturing this

dependency. The motivation behind the design of this model was capturing the out of vocabulary words that commonly occur in tweets, which word-level models usually end up skipping since they can only process words that are present in the embedding dictionary. Character-level models are, in this case, only restricted by the known characters, which results in far less ignored input. However, semantic and syntatic structures of text have to be learned from scratch, as opposed to when word-level models are used.

One can say that representations with hand-picked stylometric features are more "engineered" than those constructed by neural networks from the text's near-natural form. More precisely, these representations are based on preliminary statistics indicating which linguistic features are the best classification markers, such as pronouns and determiners in some cases. An implication of this is that different profiling attributes, e.g., gender, age and personality traits, may not be associated with the same features. Thus, profiling becomes a separate task for each classification attribute, where feature extraction should ideally be performed separately. With representations of documents created by feeding a neural network characters or words, one relies on the model's capabilities of identifying features of the terms, such as part-of-speech. This makes it harder to control and interpret the representation, as it is more difficult to understand which features or word usage dominate the classification, but it may also result in a representation that captures linguistic information that one could miss with traditional statistical means. This would be a result of the model extracting underlying intrinsic properties of the terms, from the way they are used in the text. In summary, a key difference between traditional feature engineering and using a deep learning model to extract features from text is that by traditional means you have to explicitly investigate the value of particular features, while with deep learning one delegates this responsibility to the model. Another possible advantage of the deep learning way is that by continuously feeding a model new texts, when available, it could adapt to new writing styles. Additionally, it may be difficult for traditional methods, such as n-grams, to match recurrent networks' abilities of capturing long-term dependencies (see Section 2.5.4), for which the n-gram size would have to grow substantially.

4.2.3. Classification Models

Author profiling is being approached as a machine learning problem by the majority, as with many other text classification problems these days. To get an idea of the types of models being used for author profiling, Table 4.2 contains the top 10 contestants of PAN Author Profiling 2016. Evidently, a popular choice is the Support Vector Machine (SVM), which can be considered the current state-of-the-art model for author profiling. However, there are a few instances of other approaches. Ashraf et al. (2016) tested different tree-based methods, including Random Forests, which was also used by Pimas et al. (2016). These methods were seemingly only explored for the sake of exploration, as the papers did not comment on the rationale behind the use of Random Forests and other tree algorithms. Even though it is a trivial observation, it is worth pointing out that the overall results are not only determined by the classification model, but rather as the

sum of choices made with regards to model, features and pre-processing. Nevertheless, the results were not as promising compared to other models, which is why they are not on the list in Table 4.2. Liebeck et al. (2016) actually tried multiple methods, Random Forest being one of them, but found Logistic Regression to be superior. Agrawal and Gonçalves (2016) used the method called stacking, where a combination of various classifiers were used instead of a single one. This allowed them to explore the strength of multiple classifiers, such as SVM, Naïve Bayes and Logistic Regression, and simultaneously reduce the error imposed by each classifier. As with Random Forests, stacking falls under the category of ensemble learning methods, because the classification is an aggregate of the sub-classifiers.

Compared to the amount of SVM models being used, there are few mentions of deep learning based models for author profiling. In Table 4.2 there is only one occurrence of neural networks (Dichiu and Rancea, 2016). This team used both SVMs and simple neural networks, i.e., networks containing a small number of hidden nodes and layers, which were deemed competitive against the SVM. The lack of PAN participants utilizing deep learning is surprising, as it has shown promising results for other NLP tasks. Zhang et al. (2015) showed promising results with their character-level convolutional neural networks, when compared to traditional methods, such as Bag of Words with TF-IDF (see Chapter 3) and bag of n-grams with TF-IDF, and deep learning models using pre-trained word embeddings. In a way, feature extraction and classification become parts of the same model because the tweet representation is made implicitly in the neural network. The models were tested by performing topic and sentiment classification on various datasets. A trend in the results seemed to be that traditional methods were best on small datasets of hundreds of thousand of samples, but when the size grew to several millions, the character level networks performed better. Additionally, it could be that deep networks were better at processing raw informal user texts, i.e., those containing misspellings, emoticons and expressing unfiltered thoughts, but the authors deferred this conclusion to further experiments. If this is the case, it could be useful for the task of profiling tweets, as they definitely go under the category of raw informal texts.

Parts of the discussion in the previous subsection also applies to classification models. SVMs and neural networks are more difficult to interpret than traditional machine learning techniques, such as decision trees and Random Forests, which by nature provide an intuitive understanding of what the classification is based on. SVMs can seem to be easier to use than neural networks. Libraries like Scikit-learn provide easy access to off-the-shelf SVMs, where few initial parameters must be specified before execution. Neural networks can seem more suitable for more complex problems with larger datasets, but the challenge lies in constructing the optimal model with the right topology and parameters, where a large amount of time has to be put into trial and error. Neural networks are also prone to overfitting and to getting stuck in local optimas, while SVMs always find a global solution (Burges, 1998). The difference in usage complexity may partially be a reason for the high popularity of SVMs.

Team	Model Type
Busger et al.	Support Vector Machine
Modaresi et al.	Logistic Regression
Bilan et al.	Support Vector Machine
Markov et al.	Support Vector Machine
Bougiatiotis & Krithara	Support Vector Machine
Dichiu & Rancea	Support Vector Machine and Neural Networks
Devalkeneer	Restricted Boltzmann Machine
Bayot & Gonçalves	Support Vector Machine
Gencheva et al.	Support Vector Machine
Agrawal & Gonçalves	Stacking: Bayesian Logistic Regression, Naïve Bayes, SVM

Table 4.2.: Overview of classification models used by top 10 contestants at PAN 2016 (Rangel et al., 2016).

5. Data

With supervised learning, the aim is to infer a classification function based on labeled training data. In this context, it is assumed that the training data is representative of the possible data for the given classification. An appropriate representation of this data is critical for the model's ability to estimate an accurate function. Therefore, it is beneficial to obtain an overview of the presence and absence of specific features to avoid wasting time searching for non-existing characteristics.

The following chapter will present the datasets that will be used to train and evaluate the models of this thesis. Additionally, a statistical analysis of the training set will be provided, with regard to certain stylometric aspects, based on the material presented in Section 3.6, about stylometric features, and Chapter 4, describing related work. This provides an overview of the tendencies in the dataset, with respect to the characteristics of each gender, and a foundation for the feature engineering process. The findings will be displayed and explained through visualizations. Furthermore, they will be used to justify any omission or inclusion of features in the classification systems, with emphasis on the document-level system, where feature engineering is an integral step.

5.1. Data Collection

The datasets used in this thesis are collected from two different sources. The training data, of which a fraction is withheld for validation purposes, is composed of a collection of datasets provided by PAN[1] for shared tasks in recent years, while the test set is collected from Kaggle[2], though it is provided by CrowdFlower[3]. Table 5.1 displays the number of tweets in the training set from the different years, summing to a total of 655,268 tweets. Figure 5.1 shows the distribution of the number of tweets and authors by gender. We see that the number of authors is approximately equally distributed across genders, while there a larger number of tweets by male authors than female authors. To be more precise, the male fraction represents 54% of the training set, while 46% of the tweets are written by females. This detail introduces a minor bias to the training process. It also entails that the number of tweets per author is non-uniform as well, which was verified to be true.

[1] http://pan.webis.de/

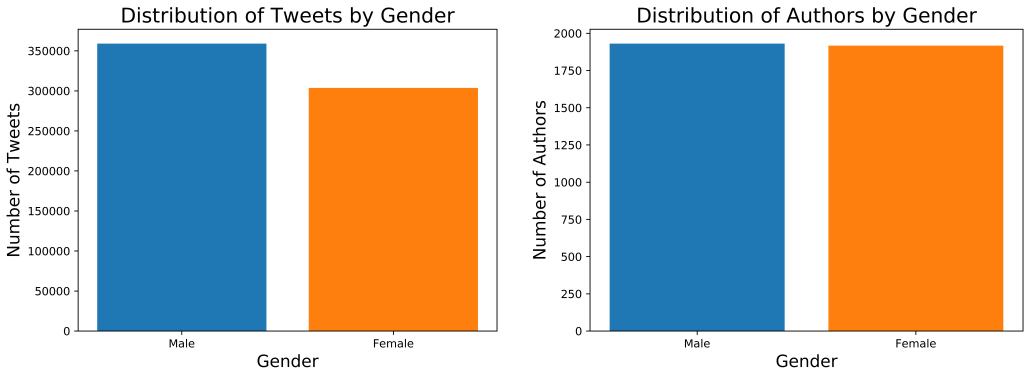
[2] Kaggle is crowdsourcing platform for predictive modelling and analytics competitions. Dataset: https://www.kaggle.com/crowdflower/twitter-user-gender-classification

[3] CrowdFlower is a data mining and crowdsourcing company.

The test set contains 12,727 tweets, of which 52% are authored by males. As the tweets in this set are supposed to represent unseen observations, they are excluded from the data analysis.

Year	Samples
PAN 2014	132,696
PAN 2015	13,862
PAN 2016	212,413
PAN 2017	296,297
Total	**655,268**

Table 5.1.: Overview of the number of tweets, from each PAN dataset, used for model training



(a) (b)

Figure 5.1.: Distribution of total tweets and authors in the dataset divided by gender. Figure 5.1(a) displays a total of 359,098 tweets by male and 303,678 tweets by female. Figure 5.1(b) displays a total of 1,931 male authors and 1,917 female authors.

5.2. Characteristics

This following section will present the actual data analysis of the training set and will cover the following points:

- Frequencies of Internet terms, such as hashtags and URLs
- Frequencies of specific emoticons
- Lengths of tweets, with respect to characters and words

- Sentiments using Valence Aware Dictionary and Sentiment Reasoner (VADER) (Hutto and Gilbert, 2014)
- Frequencies of part-of-speech (POS) tags

When studying frequencies of terms to discover trends, with respect to multiple classes, a skewed dataset poses difficulties for interpreting the results, as we need to keep in mind that some classes are represented to a larger extent. In our case, the majority class represents tweets by male authors. To compensate for the imposed imbalance, the frequency counts among female authors are scaled by a factor $f = 1.1711$, in the presented plots. f is calculated using Equation (5.1). Some of the figures use a logarithmic scale to be able to see the differences between low magnitude frequencies. The figures should make it clear when this is the case. The use of a logarithmic scale makes it more difficult to visually see the ratio between the male and female frequencies, but this will be clarified when it occurs.

$$f = \frac{total\ number\ of\ male\ tweets}{total\ number\ of\ female\ tweets} \tag{5.1}$$

5.2.1. Internet/Twitter Terms

In the domain of informal social media texts, it is usual to encounter URLs, images and emoticons. Images, in this context, are URL references to images. In addition, we have hashtags and mentions, which are phenomenons popularized by Twitter to describe the theme of the tweet and mentions of other users, respectively. As mentioned, only a frequency study was conducted. Therefore, each instance of Internet terms are generalized to the corresponding category, disregarding the specific content of, e.g., hashtags. The frequencies are displayed in Figure 5.2. The histogram shows that the overall frequencies of URLs and mentions are quite uniform, with respect to gender, while females have more frequent use of hashtags and emoticons. Also, males reference images in their tweets more often; twice as much, to be more specific, but the frequency is low compared to the size of the dataset.

5.2.2. Emoticons

A more detailed study was made on the use of emoticons. Figure 5.3, on page 49, shows the distribution of different emoticons that were detected in the dataset (note the logarithmic scale). Overall, the frequencies are surprisingly low, with respect to the size of the dataset. Nevertheless, the histogram shows that the most significant difference in usage frequency is represented by the heart emoticon (“<3”). This is used almost three times more in tweets by females compared to the tweets written by males. It is also one of the more overall frequently used emoticons. Given the magnitudes, the set of emoticons for which there is a large gap between male and female frequencies include “:)”, “:-)”, “;)” and “;-)”. For these the differences constitute more than 25%. The rather low overall frequencies indicate that the use of emoticons as features is of little

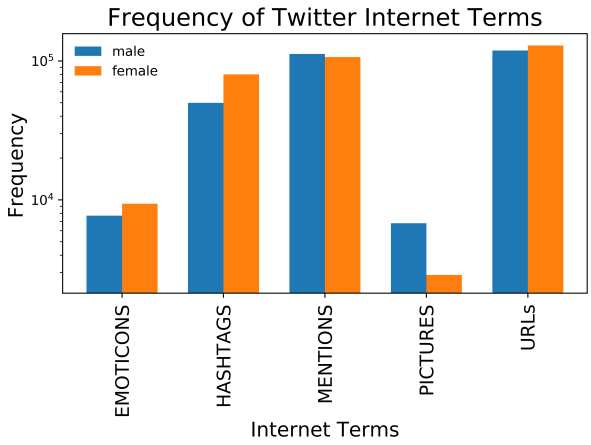


Figure 5.2.: Frequency of internet terms among gender.

use. However, since there are a few emoticons which do differ, it may be worthwhile to explore this.

5.2.3. Tweet Length

As previously shown in Section 4.2.2, various studies mention using structural features, such as the length of texts, to distinguish between genders. Figures 5.4 and 5.5, on page 50, visualize the distribution of tweets with respect to the number of characters and words, for each gender. They show that in the domain of tweets, the length provides little value for determining gender. In both figures, we see that the distributions are nearly identical for each gender. We find that there are 12 tokens in a tweet, on average, by both genders. Similarly, there are on average 66 characters in tweets, by both genders.

5.2.4. POS-tags

From the previous PAN shared tasks, we observed that participants used counts of part-of-speech tags as features (op Vollenbroek et al., 2016; Liebeck et al., 2016). The NLTK Taggers package[4] was used to perform POS-tagging on our training set. Figure 5.6, on page 51, displays the distributions of the most basic POS tags. The results of more advanced tagging procedures, with a larger number of POS categories, can be viewed in Appendix D. A description of each tag abbreviation is included in Appendix C. Regardless of the tagger complexity, the POS-tag distribution between genders is quite uniform.

[4]http://www.nltk.org/api/nltk.tag.html

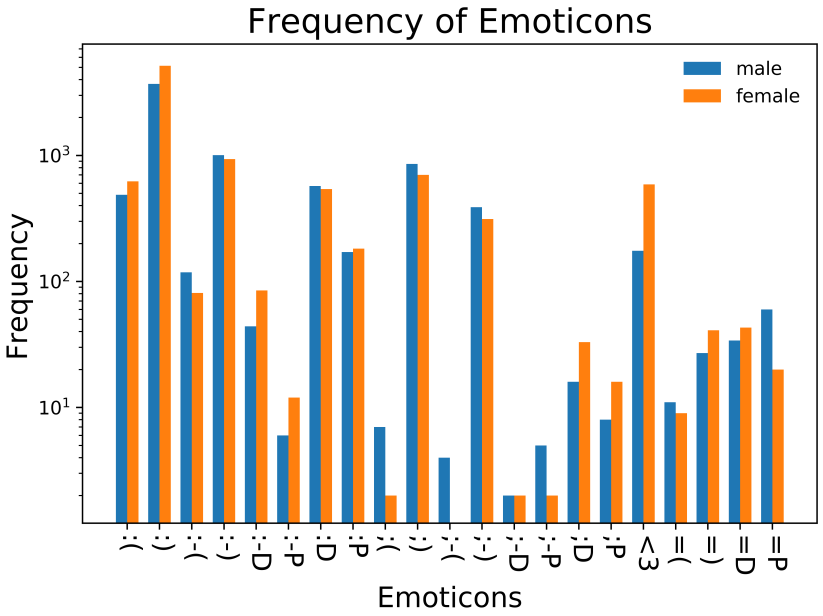


Figure 5.3.: Frequency of emoticons used among gender.

5.2.5. Sentiment Analysis

Sentiment analysis is a field within NLP aiming to identify opinions in text. In this context, a text can either be neutral or polarized, i.e., positive or negative. Using sentiment analysis on a gender-annotated dataset can reveal if tweets by either gender are more polarized, either positively or negatively. Sentiment analysis is a complex field on its own and we did not attempt to develop a separate system for this. For this, the NLTK library.[5] was used to access VADER (Hutto and Gilbert, 2014). Figure 5.7, on page 51, displays the sentiment results and shows that the predicted sentiments provide virtually no indicators of gender. The grand majority of tweets have been classified as neutral, for which the frequency among males is slightly higher than for females. We also see that the frequency of positive tweets among females is somewhat higher than for males, while the distribution of tweets predicted as negative is identical for both genders. Given these results, the only motivation for using sentiments to predict gender is if they provide predictive value in combination with some other feature.

To see if there are any differences in the usage of sentiment-indicative words, as defined by VADER, word clouds are constructed based on the most frequent positive and negative words, with respect to each gender. These are displayed in Figure 5.8 on page 52.

[5] http://www.nltk.org/_modules/nltk/sentiment/vader.html

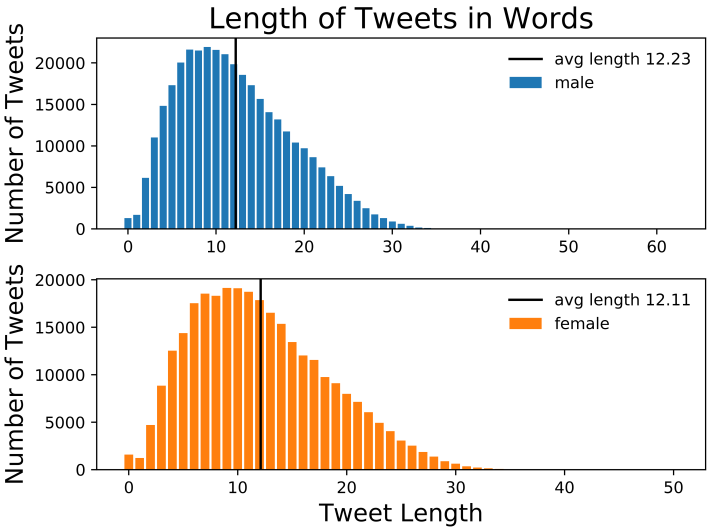


Figure 5.4.: Distribution of total words in every tweet categorized by gender.

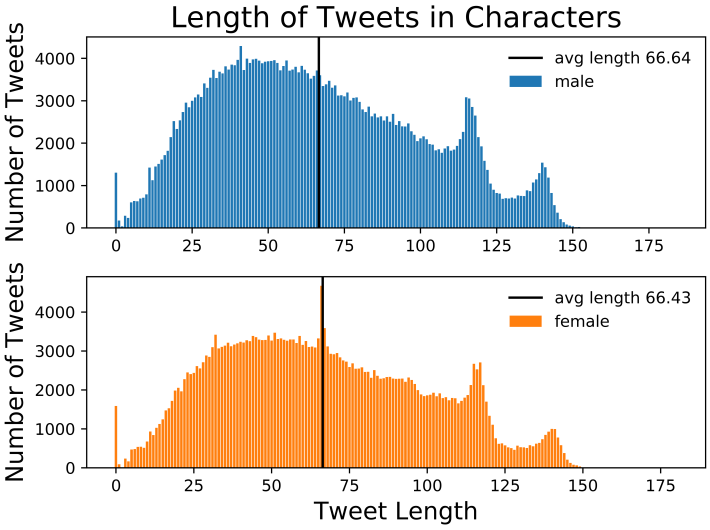


Figure 5.5.: Distribution of total characters in every tweet categorized by gender.

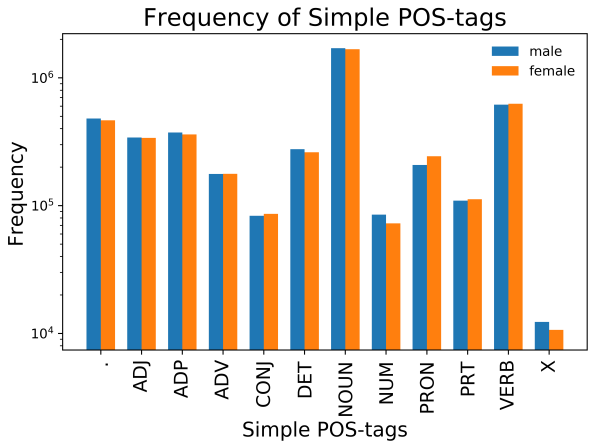


Figure 5.6.: Distribution of part-of-speech tags categorized by gender.

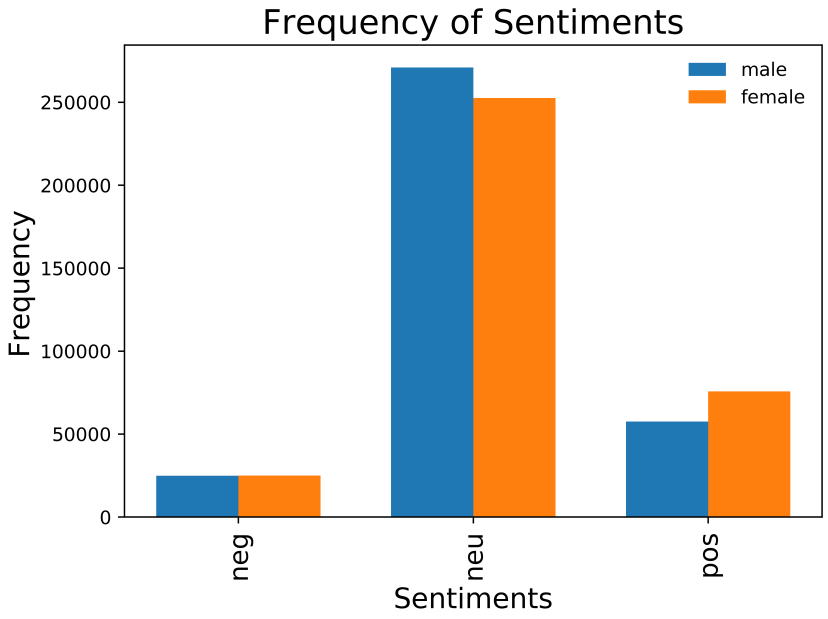


Figure 5.7.: Frequency of tweets with respect to sentiment (classified by VADER), and gender.

(a) male positive

(b) male negative

(c) female positive

(d) female negative

Figure 5.8.: Word cloud of most frequent sentiment words classified as positive or negative. Figures 5.8(a) and 5.8(b) show the most frequent sentiment-indicative words which occur in tweets by males, while Figures 5.8(c) and 5.8(d) show the same for females. The green color is an indication of positive terms, while the red color is an indication of negative terms.

6. Architecture

In this chapter, the architecture of our gender profiling systems will be described. Section 4.2, concerning state-of-the-art, outlined a general set of procedures for author profiling, based on the work by others in the field. Our system conforms to these and a coarse outline of the architecture is shown in Figure 6.1, illustrating the elements of pre-processing, feature extraction and classification. During the course of this project, three separate deep learning-based systems were developed, and this chapter describes the aforementioned aspects for each of them. The different systems process text at different levels of granularity, these being character-level, word-level and document-level. The pre-processing step is for the most part identical, while feature extraction and the classification models largely differ. Therefore, the architecture of the respective systems will be described in isolation with regards to the latter aspects. It should be noted that the final architecture of each system is based on experiments, which are presented in the next chapter. Therefore, this chapter will describe what the implementations support, while specifying what is used in the final architecture when necessary. The terms model and system will mainly be used to refer to the ANN model and the system as a whole, respectively, but the terms are interchangable and the context should make the meaning clear.

6.1. Text Pre-Processing

As explained in Chapter 5, the dataset used is a composition of data provided by PAN from the last four years of the annual shared task (2014–2017). When the data was downloaded, it was very "raw" with multiple elements of noise, which begged for a preliminary high-level filtering, before the actual filtering based on the linguistic values of terms. In addition to HTML markup tags being present, duplicate tweets were encountered, opening up for the possibility that the datasets overlap. A large number of foreign symbols were removed, leaving alphanumeric characters and punctuation. Also, even though PAN provides a set of explicitly labeled datasets in multiple languages, implying that the English dataset only should contain tweets in English, tweets in other languages were encountered in this dataset. This posed a problem because this project only focused on English tweets. A closer examination, using the library "langdetect" [1], revealed tweets in German and Dutch, among other languages. However, Spanish tweets seemed to represent the majority of this noise, while other languages were present to a much smaller degree. The language detection library was not 100% reliable, often

[1] A Python port of a Google's language detection library.
URL: https://pypi.python.org/pypi/langdetect

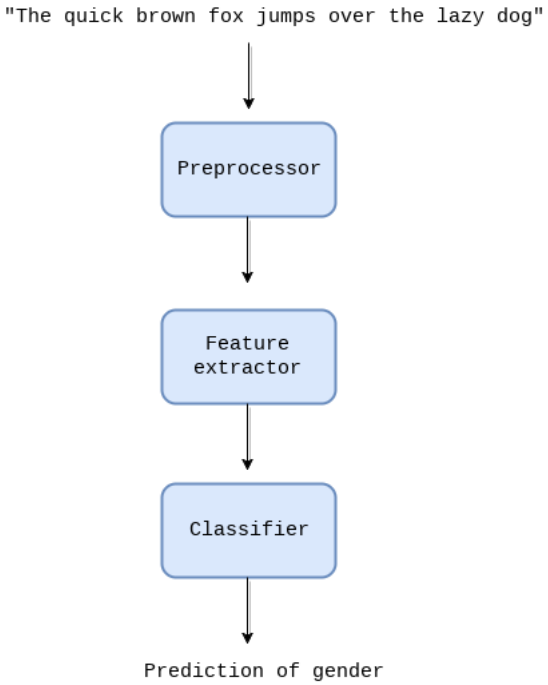


Figure 6.1.: Coarse outline of the author profiling architecture.

classifying English tweets as non-English. This issue seemed more frequent with some languages, e.g., German, than with Spanish. Because of this unreliability, a coarse filtering was deemed most appropriate. More specifically, only tweets classified as Spanish with high confidence, by the language detector, were removed from the dataset. This resulted in the removal of just above 8000 tweets, constituting approximately 1.2% of the total dataset. This rather small quantity implies that if the fraction of other languages present is even smaller, the noise imposed may be insignificant. For term tokenization, the NLTK Tweet Tokenizer was used. The preliminary filtering process just described is illustrated in Figure 6.2. Further details about the most important frameworks used in this work is described in Appendix B Based on the analysis of the data, presented in Chapter 5, and approaches found useful by others, presented in Section 4.2 about state-of-the-art, a set of methods were chosen as possible pre-processing steps for the main filtering process:

- **Lowercasing:** By not differentiating between upper-case and lower-case letters, it is assumed that content-related features will be easier to capture.
- **Replacement of Twitter/Internet terms with placeholder tags:** This branch of terms are composed of hashtags, mentions, URLs and image URLs. By replacing these terms with a corresponding placeholder tag in the text, information regarding the presence of a term of the particular type is captured, while disposing of information related to the exact hashtag, mention, URL or image.

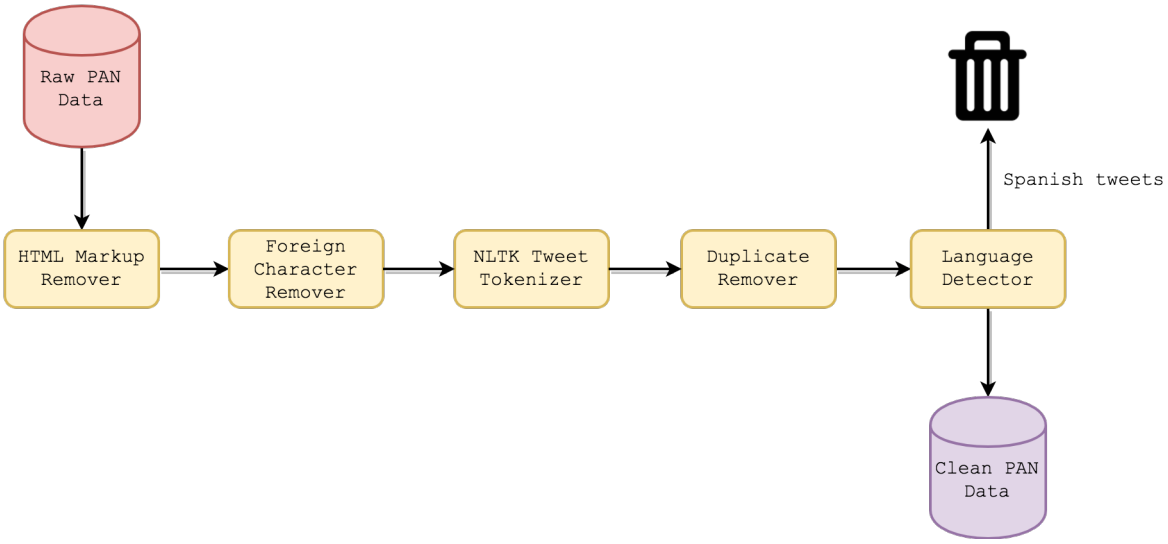


Figure 6.2.: The first level of text filtering of data.

- **Stopword removal:** Stopwords are the most common words of a language, not providing any value for differentiating classes.
- **Lemmatization:** Lemmatization is the process of reducing inflected forms of words to their root form, or lemma, and thereby treating these words as the same term. Examples of inflected forms of words include different verb tenses and plural forms. Unlike stemming, which simply chops of pre-defined suffixes in the hope of performing the correct root form reduction, lemmatization takes the context into account, identifying the meaning and the part-of-speech of the word. Hence, words are more likely to be reduced to the correct root form. However, this increased precision leads to a greater computational cost.
- **Punctuation removal:** Removing punctuations has been found useful by state-of-the-art work.
- **Emoticons removal:** The data analysis revealed that there were only a couple of emoticons which could be useful for differentiating between genders. Thus, it could be better to just remove them, reducing noise.
- **Removal of texts shorter than two characters:** Tweets are shorts in nature, making it difficult to harvest characteristic linguistic information from them. By removing tweets that are shorter than a certain threshold, the difficult and infeasibility of the problem is restricted and made more doable.

Clearly, other filtering and text modification methods could have been explored as well, but it was decided to constrain the set of methods. Since tweets are short by nature, there is a fine line between removing noise elements and loosing descriptive information

as a result of removing too much. Taking this into consideration, the contributions of the methods are evaluated. This is further described in Chapter 7.

6.2. Word-Level System

As the name implies, the word-level model processes a text by treating it as a sequence of words. In other words, the notion of characters is disregarded and the lowest level of tokens that exist, from the perspective of the model, are words. In this setting, punctuation marks are also considered words. With this in mind, perhaps "token-level" would have been a more appropriate name, but "word-level" was decided based on the use of word embeddings in this system. The following section will describe the architecture of the word-level system.

6.2.1. Text Representation

To represent these words as numerical vectors, pre-trained GloVe embeddings are used. The notion of word embeddings and GloVe were described in Section 3.5. Since our model is applied to tweets, it is desirable to use pre-trained word vectors pertaining to this domain. Therefore, we have acquired a set of 200-dimensional embeddings, containing 1.2 million words, which have been trained on 2 billion tweets. These pre-trained embeddings have been made available by Stanford[2] for the public domain. Of this large vocabulary, only a subset corresponding to the 50,000 most frequent words of the training set is used. The next chapter will include experiments conducted for choosing an appropriate vocabulary size.

6.2.2. Feature Extraction and Classification Model

The flowchart in Figure 6.3, on page 58, illustrates how to the word-level system works, after the pre-processing step. By analyzing the training set, a word index is created. This is essentially a lookup dictionary with words as keys and a unique index as value. This index is based on the frequency of the word in the dataset and acts as an identifier for the word, such that the texts can be transformed to sequences of word indices before they are fed to the Artifical Neural Network. The word index dictionary is used to create a word embedding matrix, containing the available pre-trained GloVe embeddings. In this matrix, each row contains a word embedding and each column corresponds to each feature in the 200-dimensional GloVe embedding. The row indices correspond to the word indices in the word index, i.e., row i in the embedding matrix contains the word embedding of the term with word index i. This embedding matrix is used in the ANN model for accessing word embeddings.

[2] https://nlp.stanford.edu/projects/glove/

In this architecture, the input to the ANN model needs to be of fixed size. This poses a problem since all tweets do not have the same length. Based on the word count distribution of the data, displayed in Figure 5.4, a size of 15 was chosen. This is slightly more than the average count of 12, but in return more information is covered in the longer tweets. Tweets shorter than 15 words are zero-padded to the correct size, i.e., zero values are appended, while longer tweets are truncated, leading to information loss. The size of the longest tweet could have been used as the limit, to cover all words in all tweets, but this would introduce sparsity by zero-padding and was deemed a disadvantage.

The word-level neural network architecture consists of an embedding layer followed by a Bidirectional LSTM (Section 2.5.3 explained the notion of Bidirectional Recurrent Neural Networks). The embedding matrix is used in the embedding layer as a lookup table for word embeddings by index, such that the word indices can be converted to embeddings. Figure 6.4, on page 59, illustrates the architecture of the ANN model as it is unfolded over time. Here, each input x corresponds to each word in the text. While the forward LSTM processes the text according to the original sequence, the backward LSTM is fed the text in reverse sequence. Thus, information about past and future words are made available when processing each word. Since the model is used for classification, only the output after processing the entire sequence is needed. Both LSTMs consist of 250 memory units. The outputs from both of these are merged by concatenation of vectors, composing a layer with 500 neurons. This vector of values can be considered the feature vector, or the embedding, of the tweet. To prevent overfitting, early stopping of training is induced by monitoring the performance on the validation set. Both regular and recurrent dropout are used to regularize the model. The likelihood of dropping a unit is set to 0.2 in the recurrent layers and 0.5 in the merged layer. The values were chosen through experimentation and the words of the original creators were used as guidance (Srivastava et al., 2014). However, we were not able to experiment as much as desired because of time constraints.

The output layer uses softmax activation, resulting in a distribution describing the probability of each gender. To calculate prediction error during training, categorical cross-entropy is used as loss function. As activation function in the LSTM layers, the Rectified Linear Unit is used because of its ability to learn faster than other smoother non-linear activation functions, such as tanh and sigmoid, which has granted it a high level of popularity in recent years (LeCun et al., 2015). Adaptive Moment Estimation (Adam) (Kingma and Ba, 2014) is used as optimizer. It is a gradient descent based algorithm which keeps adaptive learning rates for each parameter and appears favorable compared to other stochastic methods because of its faster and more accurate learning abilities.

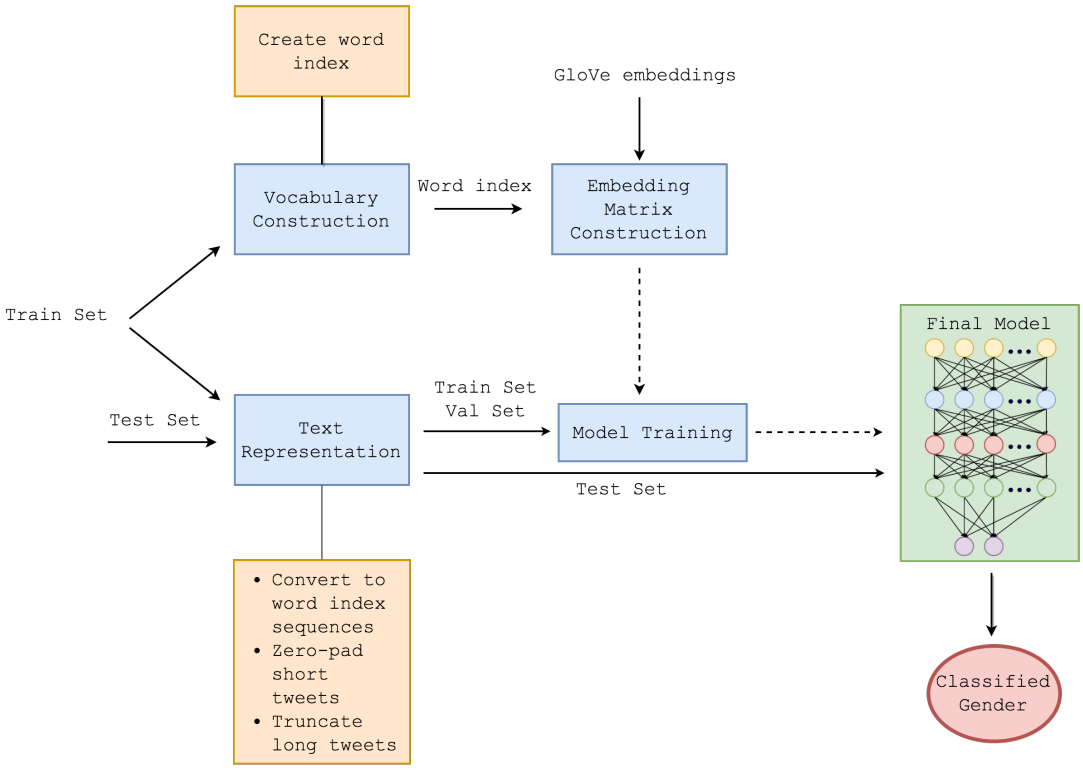


Figure 6.3.: Flowchart of the word-level system outline. The pre-processing step is omitted here.

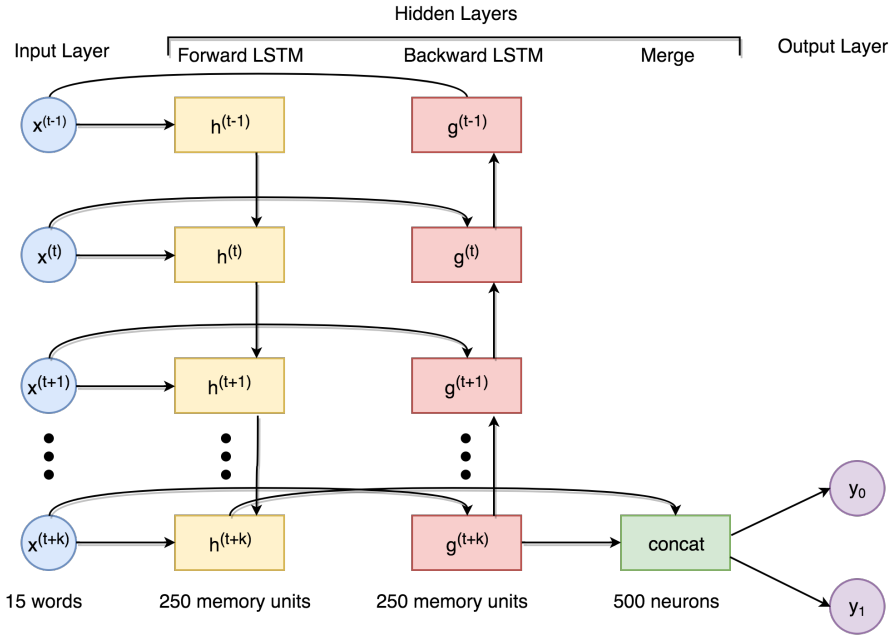


Figure 6.4.: The architecture of the word-level Artifical Neural Network model, with the LSTMs unfolded over time. The embedding layer has been omitted in the figure.

6.3. Character-Level System

An obvious drawback of the word-level model is that a vocabulary of pre-trained embeddings are needed, in addition to the fact that it is unable to process words that do not exist in the vocabulary. The character-level model is designed to bypass this limitation by processing text as a sequence of characters, rather than words. A result of this is that as long as each encountered character is known and part of the predefined vocabulary, it can be acknowledged and processed. Several studies show that character-level models prove effective for text classification (Zhang and LeCun, 2015; Zhang et al., 2015) and that the produced representations are better able at capturing morphological structures (Ling et al., 2015). This section describes the architecture of the character-level system that was developed.

6.3.1. Text Representation

Figure 6.5 shows how the architecture of the character-level system is similar to that of the word-level system, but the different level of tokenization granularity does introduce differences. The character vocabulary is constructed by processing all characters in the training set, after pre-processing, and creating a character index. Similar to the word index in the word-level model, it is a dictionary containing a unique id for each character. Thus, the texts can be transformed to sequences of character indices before feeding them to the ANN. Because the character space is very small compared to the word space, it is feasible to use sparse one-hot vectors to represent the characters.

6.3.2. Feature Extraction and Classification Model

Based on an analysis of the tweet length distribution over all tweets in the training set, provided by Figure 5.5, in Chapter 5, a length of 100 characters is chosen as the maximum sequence length. Tweets longer than these are truncated, while shorter tweets are padded to the correct size. As texts, represented as character indices, are given as input to the character-level ANN model, the characters are converted to one-hot vectors before being further processed. The architecture is displayed in Figure 6.6, on page 62. It consists of a Bidirectional LSTM, as in the word-level model, in addition to a preceding convolution layer with 1024 filters, i.e., feature maps, considering five words at a time, and a max pooling layer which summarizes two values at a time from the convolutions. The LSTMs contain 256 memory units in each direction and are merged into a layer with 512 neurons. The penultimate layer in the model is an ordinary fully connected layer with 200 neurons, comprising the embedding of the tweet. As before, softmax, ReLU, early stopping and cross-entropy loss are used, along with the Adam optimizer. Dropout is used as regularization, with a drop likelihood of 0.2 for the regular and recurrent dropout in the Bidirectional LSTM. The likelihood of dropping a neuron from the the convolution layer and merged layer is set to 0.5.

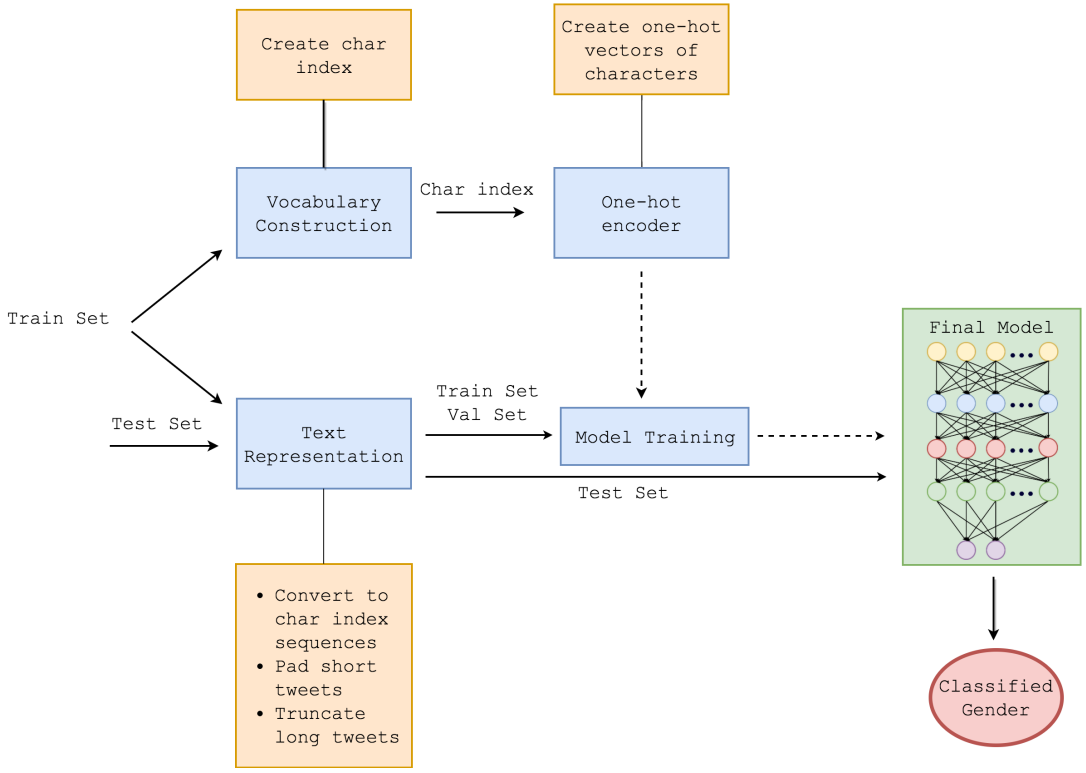


Figure 6.5.: Character-level system outline. The pre-processing step is omitted here.

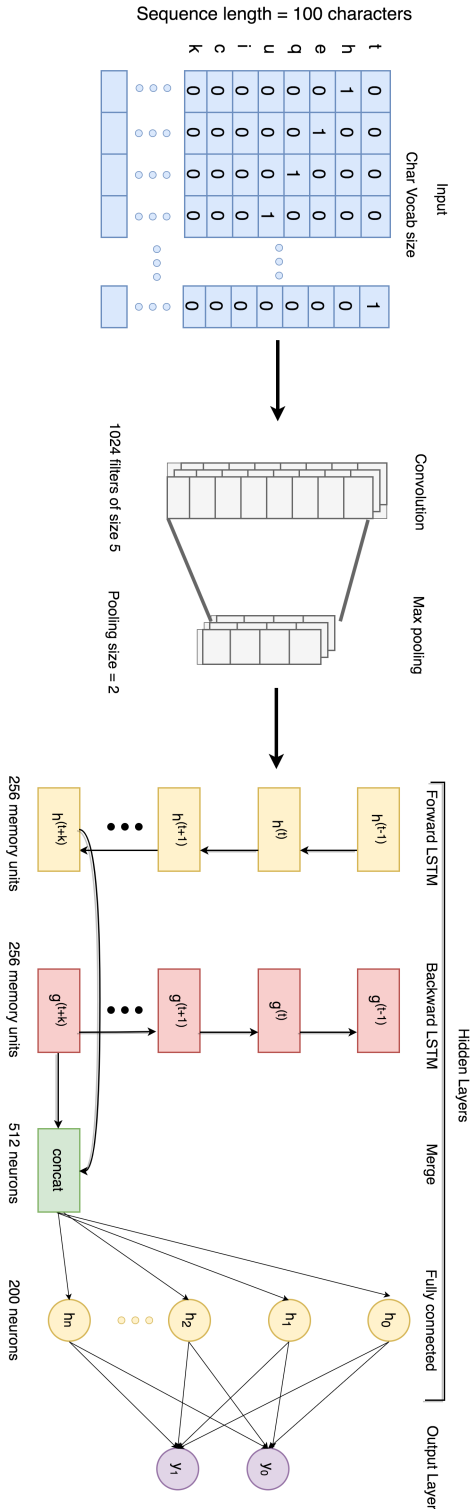


Figure 6.6.: The architecture of the character-level model. The one-hot encoder layer is omitted in the figure.

6.4. Document-Level System

The document-level system is developed significantly differently than the other systems. The feature engineering process is more explicit and the system is built supporting multiple representations, such as BoW and TF-IDF, and feature sets, which for the most part revolve around n-grams, but the text sentiment is also briefly explored. Thus, the development of this system has to a larger extent focused on feature extraction for finding the best features and the representation of these features. This section will describe the capabilities of the document-level system, which is illustrated by the flowchart in Figure 6.8, on page 66, describing feature extraction and representation methods, including the two methods used for building the n-gram vocabulary.

6.4.1. Feature Extraction

n-grams

Section 4.2, presenting state-of-the-art, showed how n-grams, being content-specific, have proved to be powerful for characterizing texts and that they are frequently used. Therefore, n-grams are used as the foundation for how texts are represented in the document-level system. In this implementation, unigrams, bigrams and trigrams are supported and tested, but as the next chapter will show, only unigrams are worth using for this problem domain. Since Internet terms, such as URLs, are a part of the most frequent terms, they will be a part of the set of unigrams.

Vocabulary Construction

The document-level system has two different methods implemented for constructing the vocabulary, using different strategies to assess the usefulness of the terms in the objective of predicting gender. The trivial approach is extracting the n most frequent terms in the training set and using these to represent the vocabulary. With an adequate size of n, the vocabulary should contain enough frequent terms that characterize the texts, and will be able to characterize new texts.

Given the problem of classifying gender, the issue with constructing the vocabulary using the most frequent terms is that words that do not distinguish genders may be present. This is a result of the terms being frequent with respect to the entire set of tweets, consisting of both male and female authors, and only leads to unnecessary high dimensionality and sparsity in the feature vectors, by terms supplying no value. While the pre-processing methods are designed to take care of a large part of this, a method for quantitatively measuring the gender-based dissimilarity of terms was engineered. For each term in the training set, a dissimilarity score is calculated using Equation (6.1), which contains the following variables:

- $numTerm_{male}$: The number of times the term occurs in the entire male subset of the training set.

- $numTerm_{female}$: The number of times the term occurs in the entire female subset of the training set.
- *total number of* $terms_{male}$: The total number of terms in the male subset of the training set.
- *total number of* $terms_{female}$: The total number of terms in the female subset of the training set.

$$dissimilarity = \left| \frac{numTerm_{\ male}}{total\ number\ of\ terms_{male}} - \frac{numTerm_{female}}{total\ number\ of\ terms_{female}} \right| \tag{6.1}$$

In other words, each term's occurrence counts in the male and female subsets of the training set are calculated. The absolute value of the difference between these two values constitute the dissimilarity score. The terms with the highest scores are then chosen to be in the vocabulary. The practical implications of this procedure is that instead of picking the most frequent words, the vocabulary is constructed using the most different words with respect to how frequently they are used by each gender.

Sentiment

In addition to using n-grams, the use of text sentiment as a feature for predicting gender is explored. Though the data analysis in Chapter 5 showed that the difference in the sentiment distribution by gender is nearly neglible, we decided to investigate if the model would be able to find a correlation between text sentiment and other features. Thus, the document-level architecture supports the use of sentiment, predicted by VADER (Hutto and Gilbert, 2014)[3], as an additional feature in the feature vector consisting of n-grams.

6.4.2. Feature Representation

The system provides two possible core representations of the texts using the features described, being:

- Bag of Words (BoW)
- Term Frequency-Inverse Document Frequency (TF-IDF)

As described in Chapter 3, BoW keeps the word count of each vocabulary term, while TF-IDF normalizes this value by down-weighting terms that occur frequently in the training set as a whole. In addition, the possibility of reducing the dimensionality of the feature vectors using autoencoders is implemented. The number of dimensions in the feature vectors depends on the size of the vocabulary, i.e., a large vocabulary results in high dimensionality. Thus, using autoencoders to reduce the dimensionality can be advantageous for creating a lower-dimensional representation, reducing computation time and memory usage, since the representations will only be a fraction of the original size, while still retaining the information in the original representation.

[3] http://www.nltk.org/_modules/nltk/sentiment/vader.html

6.4.3. Classification Model

The classifier used in the document-level system is a feedforward neural network with three hidden layers. The size of the hidden layers, in terms of neurons, are respectively 2048, 1024 and 512, as shown in Figure 6.7. The input layer is determined by the size of the feature vector which in turn is determined by the size of the vocabulary. As a vocabulary size of 10,000 results in the best performance during experiments, the size of the input layer corresponds to this value. The output layer consists of one neuron for each gender.

As with the character-level and word-level models, ReLU is used as activation function in all layers except the output layer, where softmax is used. Additionally, early stopping is used, along with categorical cross-entropy to compute loss, and Adam (Kingma and Ba, 2014) as optimizer.

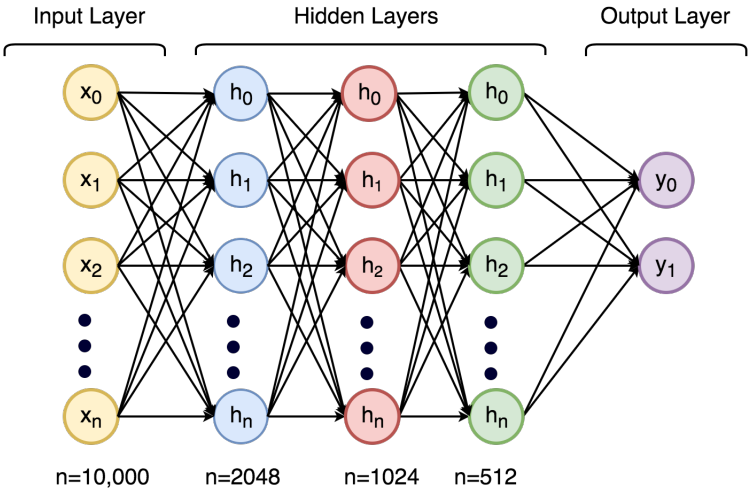


Figure 6.7.: Structure of how the Artifical Neural Network at the document-level is build. The input layer representing the x variable for each neuron that works as a placeholder for the feature vector. The hidden layers are marked by the variable h, and the predicted values with the variable y, representing the nodes of each gender.

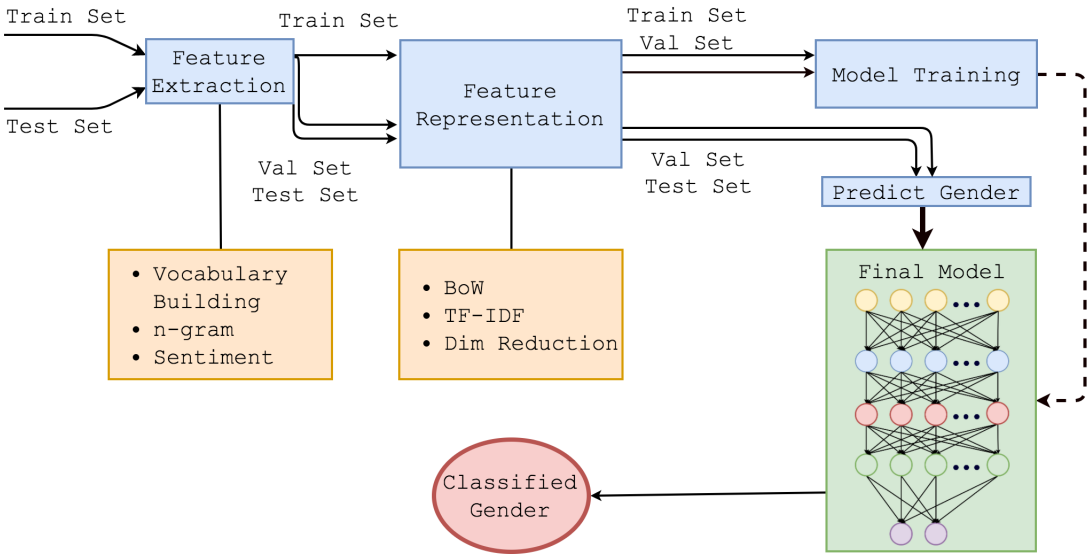


Figure 6.8.: Overview of the document-level system representing the stages the sample texts need to go through before the model can be trained and ready for the classification task.

6.5. Stacking Models

To combine the efforts of the three separately developed models, the ensemble method called stacking is used. Stacking is a method where several learners, which are based on different methods, are trained using the same data. The predictions of each classifier is then used for the final prediction. It also usual to further train a model taking the predictions of the sub-models as input. However, in this implementation we have restricted ourselves to only aggregating the predictions made by the sub-models, using different aggregation functions and without further training a combined model. This section is brief and describes how the sub-models are used collaboratively.

Aggregation Functions

Because softmax is used, each sub-model outputs a likelihood distribution summing to 1.0, which describes to which degree it believes the author is male or female. These confidence values can be aggregated in different ways. The following aggregations are made possible in this implementation and the results concerning the experiments conducted with each function are presented in the next chapter.

- **Majority:** The prediction of each sub-model represents a vote. The class represented among the majority is used as the final prediction. The motivation behind this method is that of three separate learners, if one model performs badly for certain types of tweets, the other two may steer the prediction in the right direction. The disadvantage is that if two of the models perform poorly in certain areas, they will dominate the last model, even if it is correct.
- **Maximum:** With regards to the likelihood distribution of each model, the most confident model is allowed to solely dominate the prediction. This method tries to leverage on the assumption that the more confident the model is, the more correct it is. Thus, it is not inhibited by the same issue as majority. On the other hand, if the most confident model is, in fact, often wrong, then this will have a prominent negative effect on the classifications.
- **Average:** The likelihood distributions are averaged over the three models, with respect to gender, resulting in a likelihood distribution that reflects each sub-model's prediction. This method tries to balance the advantages and disadvantages of the two former methods. Here, each model will have a say in the final prediction, weighted by the model's confidence in its prediction. The more uncertain the model is, the less it will affect the final prediction. If all models are confident of their predictions, the final prediction is practically a majority decision. However, if two models are quite uncertain of their predictions, while one model is very certain, the method practically works like maximum.

7. Experiments and Results

As with most complex problems, the path to a well-functioning system consists of many iterations of trial and error. This chapter will present the experiments that were conducted to construct the best model for gender profiling texts. The experiments serve the purpose of attaining the most accurate profiling system, in addition to understanding the behavior of the models and drawing inferences about the quality of tweet embeddings. The chapter is divided into two sections. The first section will describe the experiments that have been conducted, while the second section will present the setup of the experiments and the corresponding results.

In this project, three approaches with different foundations were explored, these being character-level, document-level and word-level (as described in the previous chapter). While the character-level and word-level architectures are quite similar methodologically, the document-level architecture is fundamentally different. The former two use Artifical Neural Networks to learn tweet features implicitly, while the latter takes a statistical approach where features are explicitly chosen before they are fed to the classifier. An implication of this is that the development patterns vary for the different approaches. Additionally, different types of ANN models were explored for each of the systems. As a result of this, some parts of this chapter will be split accordingly, to present the experiments concerning each approach in a readable fashion.

7.1. Experimental Plan

An important detail about Artifical Neural Networks is that model building is a more complex endeavour, compared to methods such as SVMs and random forests. With the various network types, optimization methods and network topologies, i.e., the number of layers, neurons and how these are connected, the number of possible models is endless. Additionally, with increasing complexity in the model, the number of hyperparameters and the possible values for these increase, along with the possibility of overfitting. It is also the case that there are cyclical dependencies between optimal pre-processing, feature extraction and classifiers, i.e., to evaluate one of the steps, the other steps must be held constant. A brute-force approach, trying all combinations is infeasible; therefore assumptions have been made, and system-specific methodologies have been followed to be able to build models in a practical manner.

The first part of the experiments concerned the matter of optimizing each of the three sub-systems, with regards to pre-processing of text, extracting features and classification

model. For each system, we followed an approach where an initial set of topologies were explored. The best performing one of these was chosen as the one to develop and optimize further. In the character-level and word-level architectures, the feature extractor and classifier are embedded in the same ANN, and thus the experiments can essentially be summarized as trying various neural network architectures, and tuning hyperparameters, to find the one that was best able to capture gender-specific features. For the word-level model, different vocabulary sizes were tried as well. After building appropriate classifiers, an ablation study of pre-processing methods was done to examine if lowercasing and the removal or replacement of specific terms had any effect.

For the document-level model, the features needed to be defined explicitly and thus the approach taken was slightly different from those of the character-level and word-level systems. Because of this, a large part of the experiments was about defining the set of features, before it could be fed to the classifier, and the set of experiments was somewhat more elaborate. The model building process for the document-level system was divided into three steps in the following order:

1. Construction of a base ANN and feature extraction
2. Ablation study of pre-processing methods
3. Optimization of classification model

The first step was to use a base ANN to find the best set of features and constructing an efficient representation of these for the classifier to process. The strategy for choosing features was based on the linguistic analysis presented in Chapter 5 and all of the features explored were n-grams, except for one, which was the text sentiment. Different vocabularies were explored based on the most frequent words and an additional method, which measured the most distinct words with respect to usage by each gender, i.e., we attempted to use the terms that are frequently used by one gender and not as frequently by the other. As representation, the methods explored were Bag of Words and Term Frequency-Inverse Document Frequency, along with various vocabulary sizes. The use of TF-IDF was motivated by its wide usage in the literature and its foundational idea of assigning less importance to words that occur too frequently in the training set. This step was followed by determining the most viable methods for pre-processing the texts by performing an ablation study. Unlike what was done for the other systems, the ablation study was performed before the optimization of the classifier. This was motivated by the increased focus on feature engineering in this approach, compared to the others. Since the number of terms captured by this model is limited by the vocabulary size, the choice of pre-processing methods was relatively important. Especially since many of them are text-filtering methods that remove or keep certain types of terms. Additionally, we attempted to optimize the representation by using an autoencoder. The last step was to optimize the classifier, with regards to hyperparameters and regularization.

To optimize the neural network classifier in each sub-system, it was initially planned

to perform automated hyperparameter optimization for selected parameters, such as the number of hidden neurons/filters/memory units, dropout rate, and other regularization constants. Grid search was one alternative, but the computational cost, of the fine-grained variant, grows exponentially with the number of parameters, as a result of trying every combination of the possible values. This is only practical when there are few hyperparameters. A course grid-search is a better option, since it limits the number of searches, but since we are using ANNs, which consist of a very large number of tunable parameters, random search was considered an even better approach. This is reasoned by its ability to explore a larger set of values, without extra computational cost, and faster convergence to viable hyperparameters (Goodfellow et al., 2016, p. 420-422). With some of the models possibly training for several hours, we were unable to execute an automated parameter optimization procedure because of time constraints. However, the various model topologies and results of manually tuning parameters will be presented.

With the number of neural network model variations that were planned to be tried, a base pre-processing configuration was defined as a starting point when evaluating the initial set of models. This base step consisted of:

- lowercasing of texts
- removal of stopwords
- replacement of URLs, mentions, hashtags and image URLs (Internet/Twitter specific terms) with placeholder tags
- Removal of texts shorter than two characters

Of the pre-processing methods described in Section 6.1, lemmatization, emoticon removal and removal of punctuation were left out because there was more uncertainty related to the effects of these methods. Thus, their usage was deferred to a later point in the model building process. Lemmatization also proved to be quite expensive computationally, which is very impractical in the initial stage of exploring a large number of neural network models.

To explore the combined power of the optimized sub-models, ensemble techniques aggregating the predictions of the individual models were used, these being majority voting, averaging the predictions, and choosing the most confident prediction. The second part of the experiments presents the gender profiling results on the test set for each individual system and the ensemble models. As an attempt to understand the behavior of the individual sub-systems better, we also took a closer look at their confidence distribution, i.e., how confident the sub-systems were in their predictions, and if any trends were present in the tweets they were most confident of.

7.2. Model Building

As described in Chapter 5, we had two sets of data at our disposal, one for training and another to use as test set. The training set contained approximately 655,000 tweets, of which 10% was used as a validation set to assess the generalization abilities of the models. The rest was used for actual training. The main metric used to evaluate the ANN models during optimization was the validation loss. The problem with metrics such as precision and recall is that they only provide information about the fraction of correctly classified data samples, but not how certain the models are of the predictions made. As earlier described, by using softmax in the output layers of the classifiers, each prediction contains confidence values summing to 1.0, describing the model's estimation of how likely it is for the sample to belong to each prediction class. In an ideal model, the correct classes will have 100% confidence, whereas the incorrect classes will have zero confidence. The loss value describes how much error there is in the confidence values. Hypothetically speaking, in a binary problem, precision and recall can both have values of 100% as long as the correct class is predicted for each sample with 0.51 confidence. In this case, the loss value will still be somewhat high, though this is an extreme example. Hence, for achieving the best possible generalization, the validation loss is the most appropriate evaluation metric during optimization.

It should be clear to the reader that the experiments in this section concern the intermediate layers, i.e., the input layer and the output layer will be held constant, as described in the architecture (Chapter 6), if not otherwise stated. The descriptions of the various ANN topologies will therefore only consist of the intermediate hidden layers.

7.2.1. Character Level Model

Model Topology

Figure 7.1 shows the training process for a subset of the character-level neural network models, after some degree of manual optimization through trial and error. The tested model topologies are based on theoretical background presented earlier, concerning ANNs and NLP, in addition to related work. The graphs illustrate the training through the loss on the validation set per epoch. Below is a high-level description of the model topologies. The description excludes the input layer and output layer:

- **2x512LSTM**: Two LSTM layers with 512 memory units each.
- **BiLSTM**: One Bi-directional LSTM layer with 256 memory units in each direction.
- **Conv_BiLSTM**: One convolution layer with 1024 filters, one max-pooling layer, one bi-directional LSTM layer with 256 memory units in each direction, and one fully connected layer with 200 neurons.

- **2xConv_BiLSTM**: Two convolution layers with 256 filters, one max-pooling layer, one bi-directional LSTM layer with 256 memory units, and one fully connected layer with 128 neurons.
- **Conv_2xBiLSTM**: One convolution layer with 1024 filters, one max-pooling layer, and two bi-directional LSTM layers with 256 memory units.

The entire set of hyperparameters of the models have been omitted at this time, as the selected models are only meant to show the variety in types of ANNs explored and the qualities of these, in addition to an impression of the model building process. For regularization, dropout was used. From the graphs, it is clear that the Conv_-BiLSTM model is superior with less loss than the other models, even though 2xConv_-BiLSTM and Conv_2xBiLSTM are more advanced versions of the same model with respectively more convolutional layers and LSTM layers. Figure 7.2, on page 75, shows the training loss and validation loss for the Conv_BiLSTM model, illustrating how the level of overfitting is rather low. Therefore, other regularization techniques than dropout were not found necessary.

Figure 7.1.: Comparison of a subset of character-level models, showing the validation loss for each epoch.

Filter Size

The filter size defines how many characters the filter of the convolution layer considers at a time in its receptive field when looking for patterns. Table 7.1 illustrates how this value

Filter-Size	Val Loss
1	0.605
2	0.590
3	0.568
4	0.561
5	0.557
6	0.558
7	0.570

Table 7.1.: Validation loss for different kernel sizes on the character-level convolution layer.

affects the system, with a value of 5-6 seeming to be optimal. Providing a meaningful interpretation of this value is difficult, but it could perhaps be related to the length of the tweets or the average length of the words.

Regularization

Figure 7.2 shows the training loss and validation loss during training. Here we can see that there are few overfitting tendencies during the first half of training, but this then grows stronger and we can see that the curve of the training loss is still moving downwards as the validation loss converges. At the point of convergence, the difference between the loss values is not too high, but to see if the loss could be decreased further, L1 and L2 regularization were explored. However, as Figure 7.3 shows, none of these improved the results. It should be noted that the regularization constants were not experimented with extensively and default values were used. The figure also shows how the model is affected considerably by using neither recurrent dropout nor regular dropout.

Visualizing the training loss vs the validation loss also revealed something else. The validation loss is actually lower than the training loss at the beginning. This is very peculiar, because it may compromise the variety of the validation set, i.e., that the validation consists of many samples that are similar.

Figure 7.2.: Training and validation loss for the convolutional bi-directional LSTM.

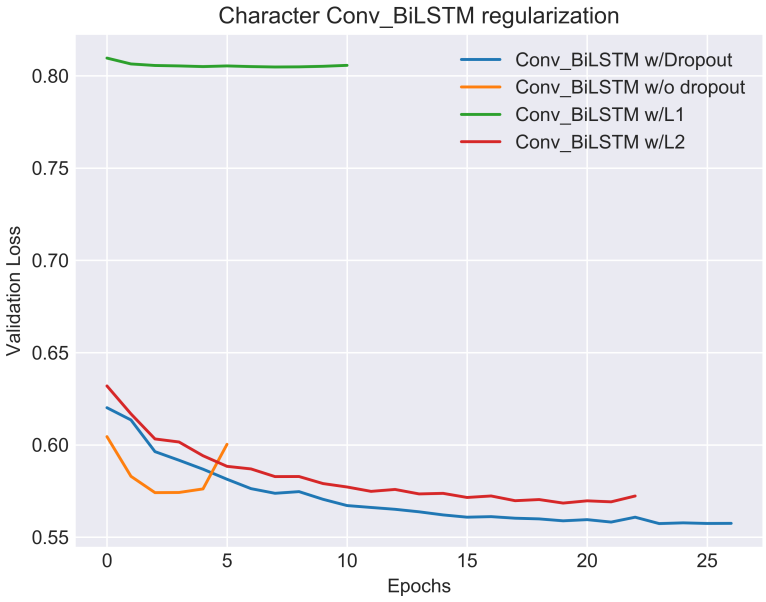


Figure 7.3.: Validation loss for the character-level model, with and without the use of different regularization techniques.

Ablation Study of Pre-Processing Methods with Character-Level Model

Table 7.2 shows the combinations of pre-processing methods that were explored with the character-level system. The results imply that most of the differences are rather marginal, with the loss for the most part staying the same regardless of the presence or absence of emoticons and stopwords. This could indicate that the neural network model is able to disregard these noise terms that are equally common for each gender. The data analysis in Chapter 5 did indeed indicate that there were no significant trends in emoticon usage with regards to gender. There were a couple of exceptions, but the low frequency of emoticons in the dataset is likely to weaken the benefits of these. Lemmatization did not change anything significantly either, while the removal of punctuation did cause a slight increase in loss, suggesting the model is able to recognize some patterns in use of punctuation by each gender. What stands out the most is the improvement in results when lowercasing is not used, promoting the model's ability to capture the usage of capital letters and use this as a feature for predicting gender.

Non-Base Methods			*Base Methods*			**Val loss**
Lem.	**R. Emo.**	**R. Punct.**	**R. Stop.**	**Lower**	**Int. tags**	
			x	**x**	**x**	0.557
x			**x**	**x**	**x**	0.558
	x		**x**	**x**	**x**	0.554
		x	**x**	**x**	**x**	0.573
				x	**x**	0.555
x	**x**	**x**	**x**	**x**	**x**	0.579
			x		**x**	0.497

Table 7.2.: Ablation study of pre-processing in character-level system. The table shows the validation loss when the various pre-processing methods are used in combination or omitted. The abbreviations represent lemmatization, removal of emoticons, removal of punctuation, removal of stopwords, lowercasing of text, and replacement of Internet terms with placeholder tags. The base methods compose the initial pre-processing configuration used to explore various model topologies.

7.2.2. Word Level Model

Model Topology

The word-level model was developed with a similar approach as with the character-level model. Different topologies were tested with regards to their ability to map gender-indicative features. Figure 7.4 shows the training process for a subset of the word-level models that were trained, with the validation loss graphed per epoch of training. Following is a high-level description of the models to gain a basic understanding of them. The description excludes the input layer and output layer:

- **4x512LSTM:** Four LSTM layers with 512 memory units each.
- **Conv_BiLSTM:** One convolution layer with 1024 filters, one max-pooling layer, one bi-directional LSTM layer with 256 memory units in each direction, and one fully connected layer with 128 neurons.
- **2x512_256LSTM:** Two LSTM with 512 memory units and one LSTM with 256 memory units.
- **BilSTM:** One Bi-directional LSTM layer with 256 memory units in each direction.
- **3xConv_BiLSTM:** Three convolution layers with 256 filters each and one bi-directional layer with 256 neurons in each direction.

Unlike when processing characters, the convolutional bi-directional network (Conv_-BiLSTM) did not perform as well here. In this case, the simpler bi-directional network (BiLSTM) has a steadier training pace, converging to a lower loss value.

Vocabulary Size

The word-level model is trained using a set of most frequent words acquired from the training set. Table 7.3 shows the impact the size of the vocabulary has on the validation loss of the neural network model. While a size larger than 25,000 does not seem to benefit the model significantly, a size of 10,000 and below present higher loss values. From an intuitive perspective, as long as no adverse effects are imposed on the loss, larger vocabularies should be preferred as they generally increase the ability to capture more information in the text. On the other hand, they turn out to be expensive with regards to memory. To balance this, a vocabulary size of 50,000 was deemed appropriate for our purposes.

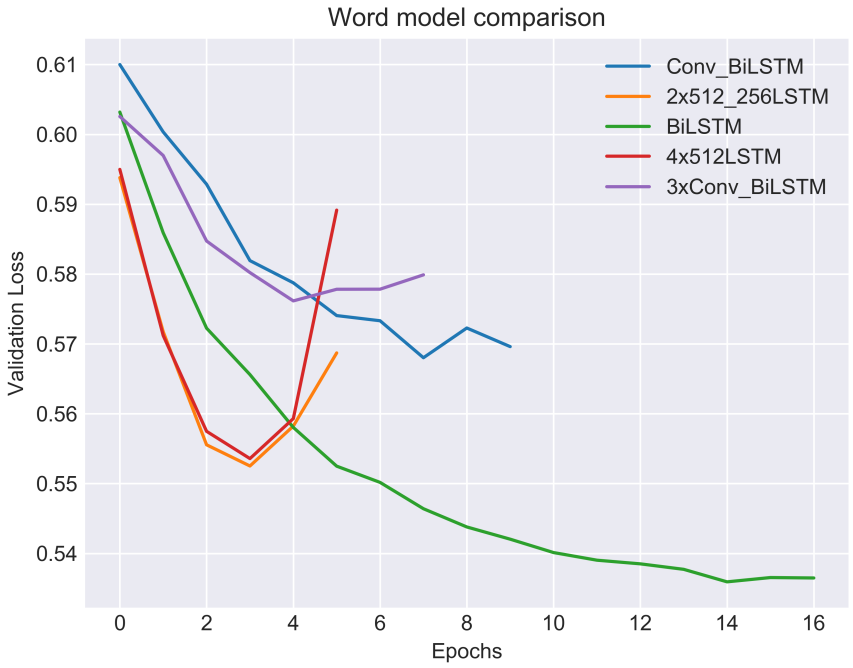


Figure 7.4.: Comparison of a subset of word-level models explored, showing the validation loss for each epoch.

Vocabulary Size	**Val Loss**
5,000	0.560
10,000	0.550
25,000	0.536
50,000	0.536
88,000	0.535

Table 7.3.: Validation loss for different vocabulary sizes in the word-level model.

Regularization

Figure 7.5, on page 80, illustrates the development of training loss, along with the validation loss, during training. We see that after four epochs of training, the training loss surpasses the validation loss, which starts to flatten out not long after. As with the character-level model, this development raises the question of whether it would be possible to lower the validation loss by using regularization methods. With dropout already incorporated, Figure 7.6, on page 80, illustrates the effects of L1 and L2 regularization on the model. Unfortunately, the results are discouraging.

Ablation Study of Pre-Processing Methods with Word-Level Model

Non-Base Methods			*Base Methods*			**Val loss**
Lem.	**R. Emo.**	**R. Punct.**	**R. Stop.**	**Lower**	**Int. tags**	
			x	**x**	**x**	0.536
x			**x**	**x**	**x**	0.538
	x		**x**	**x**	**x**	0.532
		x	**x**	**x**	**x**	0.532
x	**x**	**x**	**x**	**x**	**x**	0.538
	x	**x**	**x**	**x**	**x**	0.534
				x	**x**	0.540

Table 7.4.: Ablation study of pre-processing in the word-level system. The table shows the validation loss when the various pre-processing methods are used in combination or omitted. The abbreviations represent lemmatization, removal of emoticons, removal of punctuation, removal of stopwords, lowercasing of text, and replacement of Internet terms with placeholder tags. The base methods compose the initial pre-processing configuration used to explore various model topologies.

As Table 7.4 illustrates, the pre-processing steps seem to be even less significant for the word-level model, with little variance in the loss values. This is reasonable behavior for emoticons, since they are not present in the GloVe embedding dictionary. Punctuation marks do have corresponding embeddings, but the change in loss value is relatively insignificant, unlike the effects imposed on the character-level model. The little use of lemmatization could be explained by the fact that the inflected forms of words are likely to be located close to each other in vector space, thus imposing little change in the word embeddings that are passed through the neural network. As such, the results will practically be the same. Keeping capital letters was not explored, because the set of pre-trained GloVe embeddings only contains lowercased terms.

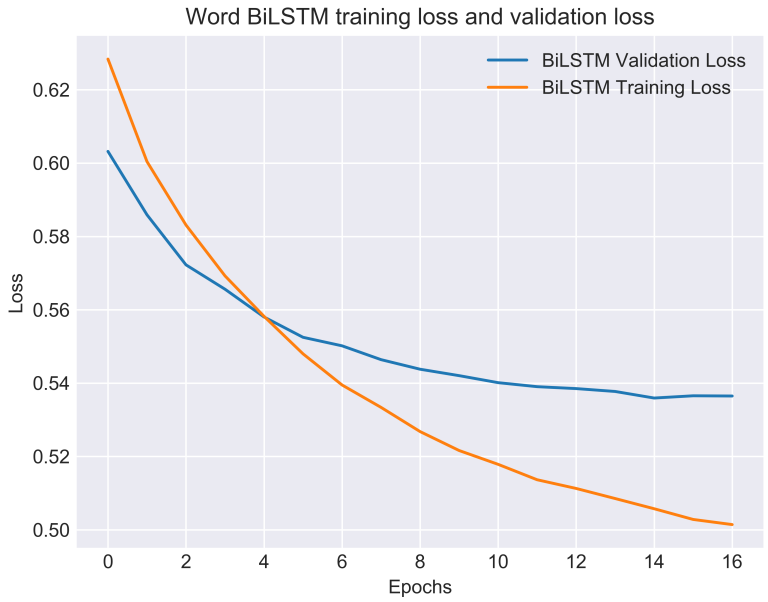


Figure 7.5.: Training and validation loss of word-level model.

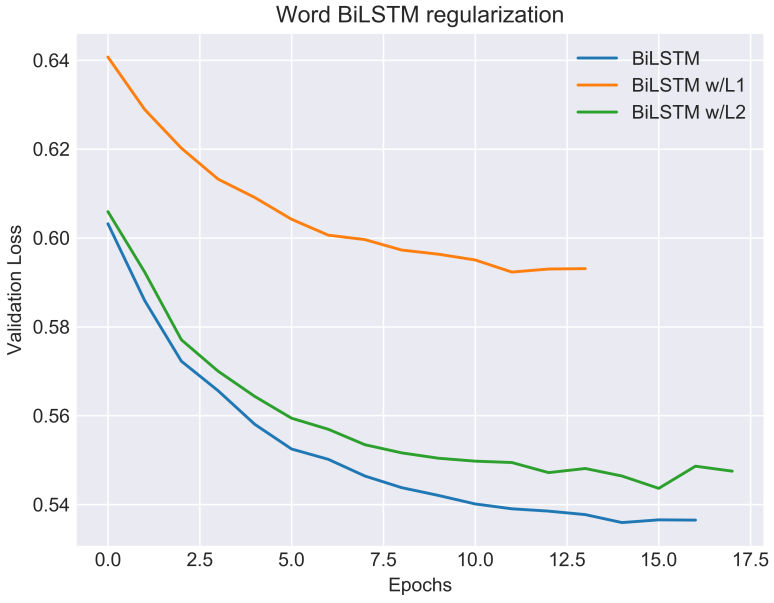


Figure 7.6.: The effects of L1 and L2 regularization on the word-level model.

7.2.3. Document Level Model

Base Model

To be able to evaluate different vocabulary sizes and representations before optimizing the classifier, a base classifier was chosen from a set of ANN models. Since only feed-forward networks were used for the document-level models, this set of models could feasibly be constructed in a structured fashion. This was more challenging for the character-level and word-level models because of the wider array of ANN variants explored and the considerably longer training time. We started simple with one hidden layer containing one neuron, and iteratively added models to the set by increasing the number of layers and neurons. This approach made it possible to see when the topology of the model was too simple to learn and at what point the increased complexity did not lead to anymore improvements. To train and evaluate the set of base models, a Bag of Words representation was used with an initial vocabulary size of the 10,000 most frequent words. This is the simplest representation and was, with a significantly large vocabulary size, deemed a good setup for experimenting with initial models.

Table 7.5, on page 82, shows the topologies of the models and the validation loss they converged to. While the differences between the loss values are within a small range, there are some pointers towards the better choices. The green row contains the model that was used as base model in further experiments. The rows in red further show how the increased number of neurons or layers can be of little use to improving the loss.

Bag of Words

Table 7.6, on page 83, displays the experiments conducted to find an appropriate size for the n-gram vocabulary containing the most frequent words in the training set. Simultaneously, different values for n were tested, these being one, two and three, i.e., unigrams, bigrams and trigrams. Since the size of the vocabulary determines the size of the input vector to the model, it was considered beneficial to keep the vocabulary as small as possible provided that too much information was not lost. Therefore, different sizes were tested to examine for each size how the validation loss was affected. For every vocabulary size, six different n-gram parameters were explored. These parameters define the range of n, e.g, (1,3) means the use of unigrams, bigrams and trigrams. From these results we see that there is relatively little value in anything other than unigrams. The vocabulary size was measured to be best with 10,000 terms. Vocabularies smaller than this threshold seemed to contain less information affecting the validation loss to the worse. There was no improvement with a larger vocabulary, which indicates that there were few additional valuable terms to be extracted.

Topology	Minimum Validation Loss
[1]	0.607
[16]	0.591
[32]	0.588
[64]	0.583
[128]	0.574
[512]	0.565
[1024]	0.555
[2048]	0.553
[4096]	0.553
[16, 16]	0.590
[32, 32]	0.583
[128, 64]	0.565
[256, 128]	0.558
[512, 256]	0.549
[1024, 512]	0.537
[2048, 1024]	0.539
[128, 64, 32]	0.568
[256, 128, 64]	0.556
[512, 256, 128]	0.555
[1024, 512, 256]	0.536
[2048, 1024, 512]	0.533
[2048, 1024, 512, 256]	0.539

Table 7.5.: Validation loss for different possible base models. The green row shows the base model that was chosen. The models marked in red show the more complex models with no performance increase compared to other simpler models.

Vocabulary Size	N-gram	Validation Loss
15,000	(1, 1)	0.539
	(1, 2)	0.541
	(1, 3)	0.542
	(2, 2)	0.628
	(2, 3)	0.632
	(3, 3)	0.649
10,000	(1, 1)	0.533
	(1, 2)	0.546
	(1, 3)	0.550
	(2, 2)	0.634
	(2, 3)	0.636
	(3, 3)	0.651
5,000	(1, 1)	0.560
	(1, 2)	0.566
	(1, 3)	0.565
	(2, 2)	0.644
	(2, 3)	0.645
	(3, 3)	0.657
2,000	(1, 1)	0.595
	(1, 2)	0.596
	(1, 3)	0.598
	(2, 2)	0.653
	(2, 3)	0.655
	(3, 3)	0.662
1,000	(1, 1)	0.622
	(1, 2)	0.623
	(1, 3)	0.626
	(2, 2)	0.660
	(2, 3)	0.662
	(3, 3)	0.667

Table 7.6.: Validation loss for different vocabulary sizes and ranges of n-grams using BoW. The grey rows represent the runs where very little learning occurred, indicating that unigrams are imperative features. The green row shows the best result in the experiment.

Vocabulary Size	N-gram	Validation Loss
10,000	(1, 1)	0.544
	(1, 2)	0.543
	(1, 3)	0.545
	(2, 2)	0.690
	(2, 3)	0.690
	(3, 3)	0.690

Table 7.7.: Validation loss for different vocabulary sizes and ranges of n-grams using TF-IDF. The grey rows represent the runs where very little learning occurred, indicating that unigrams are imperative features.

Term Frequency-Inverse Document Frequency

With basis in the Bag of Words experiments, a less exhaustive set of tests was performed for the Term Frequency-Inverse Document Frequency representation. Because a vocabulary smaller than 10,000 terms performed poorly with BoW and minimal improvement was observed with a size of 15,000, the tests were restricted to 10,000 terms. The results are shown in Table 7.7 and we see that the loss values are slightly higher. By omitting unigrams, the results are even poorer than those presented for Bag of Words.

Dissimilarity Vocabulary

As described in Chapter 6, about the architecture, a method for quantifying the gender-based dissimilarity of terms was tried. A subset of the words with highest dissimilarity scores are displayed in Table 7.8 and here we can see, e.g., the smiley emoticon ':)', which we discovered was more frequently used by women (Chapter 5). Table 7.10 shows the results by using this approach to construct the vocabulary, followed by using BoW and TF-IDF as representation. Unfortunately, the results are relatively similar to those seen before using most frequent terms. This observation begged for further investigation. The greyed out terms in Table 7.8, constituting almost half of the displayed terms, are also found in the set of 100 most frequent terms. A subset of the latter is shown in Table 7.9. In this table we do see some less topic-descriptive terms, such as '&' and 'via'. However, further investigation revealed that the least frequent dissimilarity term in Table 7.8, "basketball", appeared as the 651th most frequent term. Essentially, this implies that as long the vocabulary size is large enough, it is likely that all of the most gender-identifying words will be captured. This substantiates that with a vocabulary size of 10,000, both vocabularies will, for the most part, contain the same terms. Another observation is that the performance of BoW and TF-IDF are nearly identical, which conforms to the expected behavior because the dissimilarity formula already implicitly diminishes the weight of terms which are too common. In addition, there was little difference between these two representations when using most frequent terms. If the vocabularies are practically the same, this is expected.

Dissimilarity Words by Gender			
love	day	today	thank
data	google	happy	please
credit	artist	help	game
tanks	card	pace	christmas
life	horse	:)	amazing
photo	news	web	beautiful
women	painting	basketball	marketing
lovely	man	play	girl

Table 7.8.: Subset of the 100 terms with the highest gender-based dissimilarity score.

Frequent Words in Vocabulary			
new	&	via	like
one	today	day	get
good	great	time	thanks
love	people	know	see
back	think	work	year
go	:)	best	would
us	first	last	really
need	going	happy	make

Table 7.9.: Subset of the 100 most frequent terms

Model	Vocabulary Size	N-gram	Validation Loss
TF-IDF	10,000	(1, 1)	0.543
		(1, 2)	0.553
		(1, 3)	0.555
BoW	10,000	(1, 1)	0.543
		(1, 2)	0.551
		(1, 3)	0.555

Table 7.10.: Validation loss for TF-IDF and BoW using a vocabulary constructed using the most distinguishing words for each gender

Sentiment as Feature

A brief experiment was conducted using text sentiment as an extra feature. The vocabulary size, and type of n-grams used, was based on the best results from the experiments with BoW and TF-IDF, i.e., 10,000 unigrams. The resulting performance of validation loss using sentiment as additional feature did not make any difference compared to the

initial setup. The validation loss for using sentiment as feature ended up being 0.546, which is slightly higher than using the initial setup scoring 0.533.

Ablation Study of Pre-Processing Methods with Base Model

Non-Base Methods			*Base Methods*			**Val loss**
Lem.	**R. Emo.**	**R. Punct.**	**R. Stop.**	**Lower**	**Int. tags**	
			x	x	x	0.533
x			x	x	x	0.542
	x		x	x	x	0.542
		x	x	x	x	0.544
x	x	x	x	x	x	0.542
	x	x	x	x	x	0.545
				x	x	0.542

Table 7.11.: Ablation study of pre-processing in document-level system. The table shows the validation loss when the various pre-processing methods are used in combination. The abbreviations represent lemmatization, removal of emoticons, removal of punctuation, removal of stopwords, lowercasing of text and replacement of Internet terms with placeholder tags. The base methods compose the initial pre-processing configuration used to explore various model topologies.

Similar to the character-level and word-level systems, the use of pre-processing methods to filter the text for certain words made virtually no difference at all. Table 7.11 shows the results. Our intuition was also that the use of lemmatization would increase the model's chance of capturing more content information since the term space would be reduced and generalized. Since this model is to a larger extent restricted by its smaller vocabulary, we hypothesized that lemmatization would provide more use here than for the other models. The results, however, show otherwise.

Dimensionality Reduction with Sparse Autoencoders

A drawback of representing texts using BoW is that the size of the vectors are the same size as the size of vocabulary. This poses a potential problem with regards to sparsity and dimensionality. Our data analysis showed that on average, a tweet in the training set contained approximately twelve words (see Figure 5.4). This implies that with a vocabulary containing 10,000 terms, only twelve of the 10,000 values are non-zero on average, i.e., 99.9% of the values in a tweet representation are zeros on average. According to Xu et al. (2013), such sparse representations can lead to overfitting and reduced generalization. The authors used a dimensionality reduction technique similar to autoencoders to increase document classification accuracy. The minimal differences in the results of the latter experiments also underpin the assumption that the representation

could be too high-dimensional and sparse. Additionally, training would usually stop after a few epochs because of overfitting and the lack of improvement in the validation loss. These tendencies provided the motivation for trying dimensionality reduction of the BoW representation with autoencoders, to represent the same information in a lower dimension, thus increasing the possibility for better generalization.

The sparse autoencoder implemented in this experiment was inspired by the description of sparse autoencoders in "Representation Learning: A Review and New Perspectives" by Bengio et al. (2013, section 7.2) and from the Keras blogg.[1] As it was not initially a part of the plan to use autoencoders, not much time was allocated to optimizing this process. Thus, only three reduction sizes were tested and the depth of the autoencoder was not thoroughly explored. Based on Goodfellow et al. (2016, p.510), a single hidden layer with 1,000 neurons was added between the input layer and the encoding layer. Table 7.12 displays the results for the three different reduction sizes. Recall that the best measured validation loss with base model was 0.533. As we can see the, the dimensionality reduction in all three trials seems to result in information loss during the encoding process, since the loss increases. Figure 7.7 shows the training and validation loss, during training, when using the 300-dimensional encodings and the original 10,000-dimensional representations. We can see that, though there is information loss, overfitting is considerably reduced. However, this could also be related to the model's difficulties with extracting information from the encodings.

Reduction Dimension	**Validation Loss**
1,000	0.589
500	0.587
300	0.587

Table 7.12.: Reduction size from the original representation size and there performance in validation loss using the base model. The reduced size are reduced from the original represenation with BoW 10,000 most frequent terms as vocabulary.

Model Optimization

The previous experiments revealed minimal differences in gender prediction performance, using various feature extraction methods and representations. Disregarding the insignificance of the differences, the Bag of Words representation, with a vocabulary composed of the 10,000 most frequent unigrams, presented the best performance with respect to validation loss. With this setup, an effort was made to optimize the base model classifier, with regards to topology and regularization. The experiments conducted to construct the base model indicated that a model with three hidden layers seemed to be the most viable topology. As shown in Table 7.13, trying more complex structures with four hidden layers and varying the number of neurons led to no improvement,

[1] https://blog.keras.io/building-autoencoders-in-keras.html

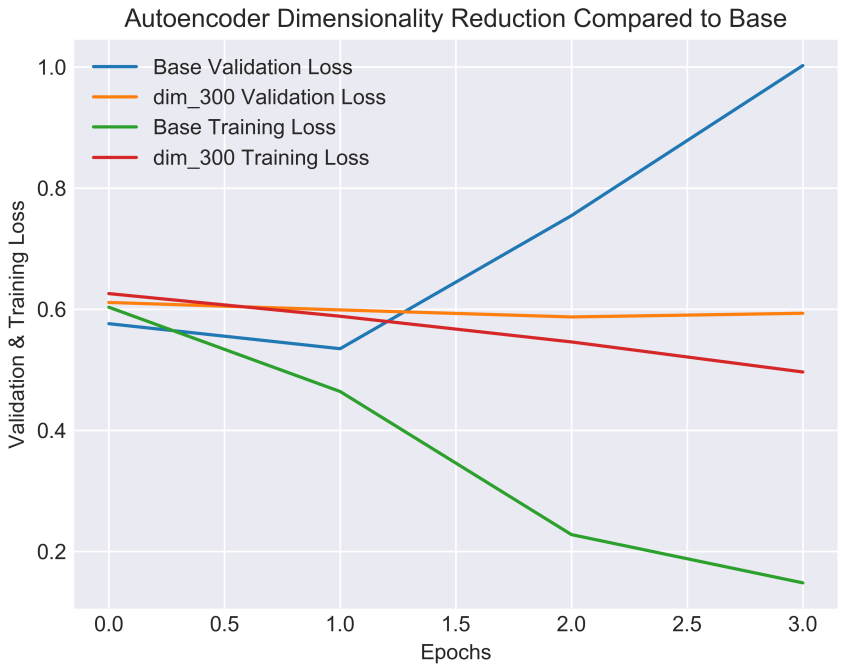


Figure 7.7.: Training and validation loss when using representations with reduced dimension size (dim_300), compared to using the original 10,000-dimensional base representations.

resulting in the base model becomes the final model. Several regularization techniques were tried on the model, but as Figure 7.8 displays, there were no further improvements using regularization. Evidently, the use of batch normalization results in extremely poor performance. This could be related to the level of sparsity in the representations. Furthermore, the plot shows that the rest of the regularization techniques result in similar performance to using no regularization (denoted as simply 'Base' in the figure), until each graph diverges. As previously mentioned, time constraints prevented us from doing extensive experimentation with regularization parameters.

Layers	Minimum Validation Loss
[800, 400, 200]	0.537
[1500, 750, 375]	0.535
[1024, 2048, 1024]	0.542
[512, 512, 512, 512]	0.545
[800, 400, 200, 100]	0.546
[1024, 2048, 1024, 512]	0.543

Table 7.13.: Combination of other topologies and there performance.

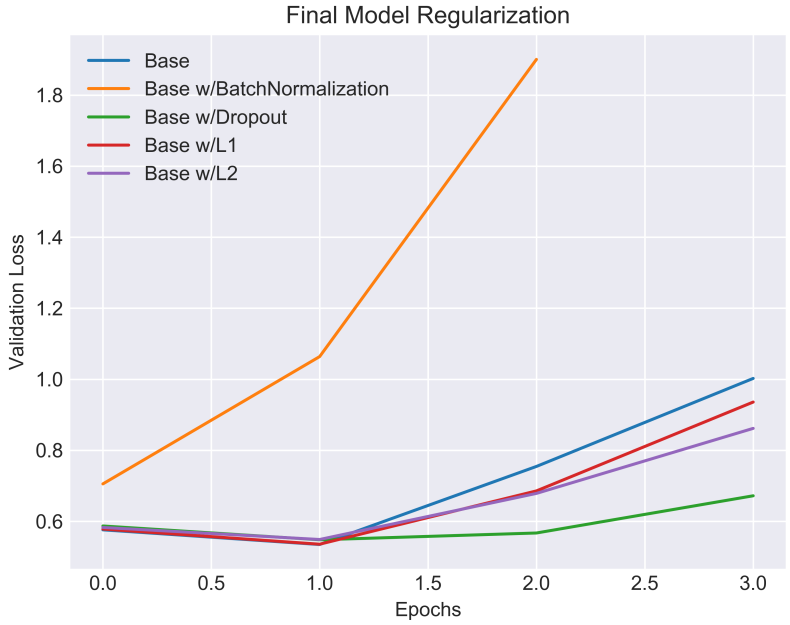


Figure 7.8.: Performance on validation loss comparing regularization techniques with the base model.

7.3. Validation Set Results

Tables 7.14 to 7.16 present the precision, recall and F1-score for the character-, document- and word-level systems on the validation set. A brief summary[2] of the score metrics is provided for convenience, explained with respect to a classification class C:

- Precision: The fraction of relevant instances among the retrieved instances, i.e., the fraction of instances correctly classified as C among the instances classified as C.
- Recall: The fraction of relevant instances retrieved of all relevant instances, i.e, the fraction of instances correctly classified as C of all existing instances of C.
- F1-score: A measure for averaging precision and recall to get a better overall evaluation of the system. More specifically, it is the harmonic mean of precision and recall.

These scores are presented for each gender and the system as a whole, with the values for the latter being identical for precision, recall and F1-score in binary classification, using the micro-average. The micro-average is biased towards the more populated class in the dataset. On the other hand, macro-average is biased towards the least populated class. In our case, the choice was trivial, as the the two metrics presented similar results.

Character-Level			
	Female	**Male**	**Overall**
Precision	0.750	0.733	0.740
Recall	0.652	0.815	
F1-score	0.697	0.772	

Table 7.14.: Validation scores for the character-level system.

Word-Level			
	Female	**Male**	**Overall**
Precision	0.702	0.722	0.713
Recall	0.663	0.757	
F1-score	0.681	0.740	

Table 7.15.: Validation scores for word-level system.

The results correspond to the reported validation loss results, which showed that the character-level model had less error than the word-level and document-level models, with

[2] *A more detailed description about the metrics can be at Scikit-Learn: http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html*

Document-Level			
	Female	**Male**	**Overall**
Precision	0.714	0.733	0.725
Recall	0.671	0.771	
F1-score	0.692	0.751	

Table 7.16.: Validation scores for document-level system.

a value of 0.497. The latter two had similar amounts of validation loss, at approximately 0.540. We can also see that the models are more proficient at predicting males. In the case of the character-level model, recall for males is considerably higher than the precision. In addition, the precision for females is slightly higher than precision for males. This indicates that the model is more careless when producing a prediction as male, i.e., a higher rate of male class predictions.

7.4. Test Set Results

The following tables present the precision, recall and F1-score for the character-, document- and word-level systems on the test set, in addition to the scores of the ensemble models, which combine the previously mentioned sub-systems. For convenience, the overall score for each sub-system, and each variant of stacking model, are provided in Table 7.17. Logistic Regression, Random Forest and Naïve Bayes are used as baseline models and their scores are also included in this table. The text representation used for these models are the same as for the document-level model. More detailed tables will be presented in the following paragraphs.

Sub-Systems

Character-Level			
	Female	**Male**	**Overall**
Precision	0.616	0.570	0.592
Recall	0.559	0.626	
F1-score	0.586	0.597	

Table 7.18.: Test scores for character-level system.

System Type	Overall
Logistic regression	0.591
Random Forest	0.551
Naïve Bayes	0.577
Character-level	0.592
Word-level	0.604
Document-level	0.585
Average Confidence-**CWD**	0.610
Average Confidence-**CW**	0.613
Average Confidence-**CD**	0.598
Average Confidence-**WD**	0.602
Majority-**CWD**	0.606
Maximum Confidence-**CWD**	0.610

Table 7.17.: The overall test performance for each sub-system and ensemble model. The abbreviations **CWD**, **CW**, **CD** and **WD** correspond to the sub-models that have been combined.

Word-Level			
	Female	**Male**	**Overall**
Precision	0.613	0.593	0.604
Recall	0.633	0.572	
F1-score	0.623	0.582	

Table 7.19.: Test scores for word-level system.

Document-Level			
	Female	**Male**	**Overall**
Precision	0.596	0.572	0.585
Recall	0.613	0.554	
F1-score	0.604	0.563	

Table 7.20.: Test scores for document-level system.

Tables 7.18 to 7.20 illustrate the differences in performance between the three sub-systems. Compared to the validation results, we can instantly observe a discrepancy with the results on the test set being considerably worse that on the validation set. This will be further addressed in the next chapter. On a larger scale, the test scores are quite similar. The largest difference in overall performance is around 1.5%, being the difference between the score for the word-level system and the document-level system. However, with respect to the low magnitudes, we can still make some observations to compare the systems.

With regard to the individual classes, the character-level system performs better than the others when predicting male authors, while the word-level system scores best predicting females. None of the systems turn out to be superior at classifying both genders, though the overall performance of the word-level system is better than the other two. The document-level system is the weaker of the three, being outperformed by the word-level model at every single point. Though it is around 2% better at predicting females than the character-level model, its ability to predict males is nearly 3.5% weaker. Thus, the utility of the document-level model could be questioned. However, an important detail is that these results do no indicate whether or not the separate systems are correctly classifying the same tweets, which is the very incentive for utilizing ensemble techniques, such as stacking.

Stacking Systems

Different variants of aggregation functions were tested for the ensemble models, these being majority voting, averaging and maximum of individual confidences. Table 7.21 displays the scores achieved by averaging all three sub-models. As expected, the overall system performance is higher than that achieved by the individual models alone, though not by much. The averaged ensemble model only gained 0.6 % compared to the word-level system, producing an overall score of 0.610.

To gain a better perspective of the contribution made by each sub-model, three additional tests were run where the sub-models were averaged pairwise. These tests reveal that the best performance is achieved by averaging the character-level and word-level systems, resulting in an overall score of 0.613. Table 7.22 displays these results. Averaging the character- and document-level systems produces a slight performance increase from that achieved by the individual models alone, because of their individual ability of better predicting males and females, respectively. However, combining the word-level system with the document-level system actually decreases the score with 0.02%, compared to the word-level model's individual score. This is not unexpected, though, as the word-level system achieves better scores for each metric on its own. Essentially these results indicate that the word-level model is for the most part covering what the document-level system is able to learn. On the other hand, the character-level system's ability to more accurately predict males than the word-level system is likely to be the cause of the 1% increase in performance from the word-level model's individual score.
From Table 7.23, we can see the slight increase in overall score achieved by granting each sub-system a vote. The majority determines the prediction of each tweet. Even though the overall score is increased, the F1-score for each gender is not higher than the maximum score achieved by an individual system. This illustrates how the ensemble model incorporates both the positive and negative aspects of the sub-models. Table 7.24 shows the results from the last aggregation function explored, where the model that is most confident in its prediction is allowed to dominate each classification. This produced similar, but slightly worse results than when averaging the sub-systems, indicating that

Average Confidence - All Systems			
	Female	**Male**	**Overall**
Precision	0.621	0.597	0.610
Recall	0.629	0.590	
F1-score	0.625	0.593	

Table 7.21.: Test scores when stacking all three systems, using confidence average as aggregation function.

Average Confidence - Character- and Word-Level			
	Female	**Male**	**Overall**
Precision	0.627	0.598	0.613
Recall	0.621	0.605	
F1-score	0.624	0.601	

Table 7.22.: Test scores when stacking character- and word-level systems, using confidence average as aggregation function.

the model most confident of its prediction is not always right. Combining only the character- and word-level systems, with the same method, produced exactly the same results as when averaging the two systems, which is the more desired behavior, i.e., that the more confident a model is in its prediction, the more likely should it be that it is correct as well. The extended score tables for some of the tests have been omitted here, but can be viewed in Appendix D.

Majority - All Systems			
	Female	**Male**	**Overall**
Precision	0.620	0.592	0.606
Recall	0.617	0.595	
F1-score	0.618	0.593	

Table 7.23.: Test scores when stacking all three systems, using majority as aggregation function.

Maximum Confidence - All Systems			
	Female	**Male**	**Overall**
Precision	0.621	0.598	0.610
Recall	0.632	0.587	
F1-score	0.626	0.592	

Table 7.24.: Test scores when stacking all three systems, using the maximum individual system confidence as aggregation function.

Gender Exclusive Training

One experiment was conducted where the models were only trained on one gender, but validated and tested on both, to investigate if either gender perhaps held more characteristic information in their tweets than the other. However, as Tables 7.25 and 7.26 show, this only resulted in extreme overfitting and a bias for the gender that was trained on.

Training on Only Females			
	Female	**Male**	**Overall**
Precision	0.517	0.00	0.517
Recall	1.000	0.00	
F1-score	0.682	0.00	

Table 7.25.: Test results by only using females in training set. The results were the same for all three sub-models.

Training on Only Males			
	Female	**Male**	**Overall**
Precision	0.00	0.483	0.483
Recall	0.00	1.000	
F1-score	0.00	0.651	

Table 7.26.: Test results by only using males in training set. The results were the same for all three sub-models.

Correlation Between Confidence and Prediction of the Systems

As shown previously, combining the systems somewhat increased the prediction power compared to that of the individual systems. The results showed that the word-level system was best when predicting females, in addition to essentially covering what the document-level was able to learn. Also, the character-level model was able to predict males more accurately than the word-level system. To investigate if there was a correlation between the performance of the models and how confident they were of their predictions, a brief study was done on this very subject. Thus, we would be able to obtain a better understanding of the behavior of the three models.

Figure 7.9 visualizes the level of confidence the three models have in their predictions of the test set. The figure can be confusing and begs for some explanation as the graphs are smoothed out curves based on histograms for the three sub-models. This was done to visualize and compare the sub-models. The x-axis in the graph represents the degree of confidence, and is measured using Equation (7.1), i.e., the difference between the model's estimated probabilities, or confidences, for each gender. The y-axis represents

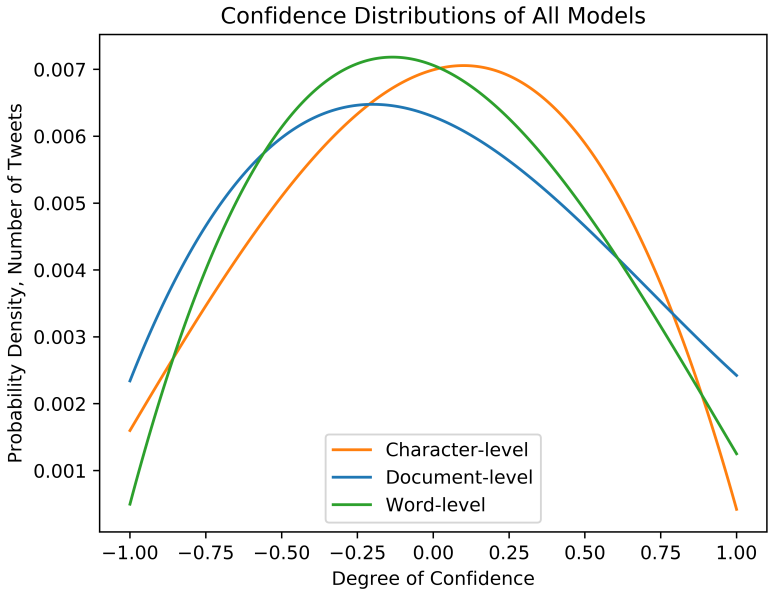


Figure 7.9.: Prediction confidences of each model where the x-axis shows the degree of confidence. Each end-point on the x-axis (-1.00 and 1.00) represent maximum confidence, while 0.00 represents maximum uncertainty. The y-axis represents a normalized measure for the number of texts so that the each graph sums to 1.00, as a probability density.

the probability density as the fraction of the test set that had a confidence difference equal to the value of x. The interval $[0, 1]$ on the x-axis means that the tweet was predicted as a male, while the interval $[0, -1]$ corresponds to the prediction as female. A value closer to 1.00 or -1.00 corresponds to the increased confidence in the prediction, as respectively male and female. As one gets closer to 0.00, the model is more uncertain about its predictions. Since the y-values correspond to fraction of the test set, the area under each curve equals to 1.00.

$$x = P(Male) - P(Female) \tag{7.1}$$

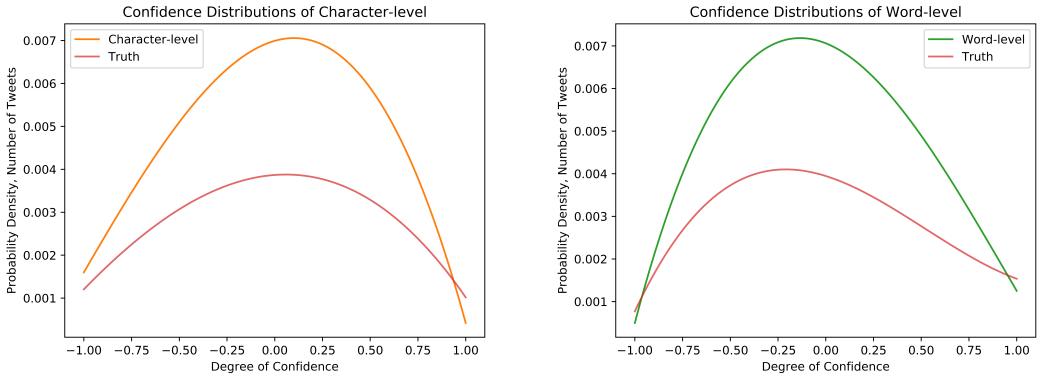
The graph illustrates that the three models have learned to predict each gender with varying level of confidence and when compared to the test result scores for each sub-model, we can learn some interesting aspects. The end-points of the blue curve in the graph show that the document-level model was more confident in its predictions than the other models. However, this stands in contrast to the actual performance of the model which was the less accurate of the three. Intuitively, one would expect the curve to be lower at the end-points and higher around $x = 0.00$, i.e., that the model would be less certain of its predictions, given the weak performance.

The area beneath the green curve, which represents the word-level model, covers almost all of the area beneath the blue curve. This implies the same as noted earlier, that the word-level model may be covering most of the knowledge that the document-level model possesses. It can also be observed that the orange curve of the character-level model is skewed to the right, indicating its higher confidence in male predictions, while the green curve is skewed to the left. This supports the results seen earlier, where the character-level model was superior predicting males, and likewise with the word-level model for prediction of females. Then, adding the area covered by the orange curve and the green curve results in spanning a wider area to the left and right. Hence, the confidences for predicting each gender are increased, supporting the notion of superiority when combining these models, which was also proven by the test results.

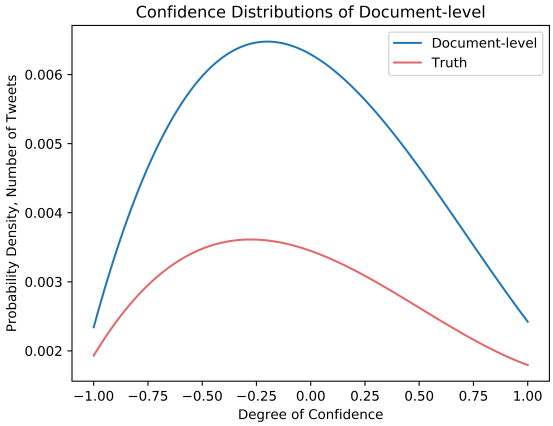
The test scores of the models showed that the accuracy of predicting the genders lie around 60%. However, these values do not show the correlation between the prediction confidences of the models to the true genders. It is desirable to have a correlation such that high confidence implies high accuracy, and likewise for low confidence and low accuracy. Figure 7.10, on page 98, visualizes the correlation between the prediction confidences and the true genders. This is done by plotting the same curves as those in Figure 7.9 along with a curve representing the fraction of these predictions that were correct. Thus, the error can be viewed as the distance between the red line and the confidence curve of the model. A common property of the error rate for the models is that it increases when the model has very low confidence, i.e., gets close to zero. For the predictions the models are most confident of, i.e., when the degree of confidence is at -1.00 or 1.00, it is clear that the error rate is highest for the document-level model because the distance between the curves is higher than for the other models. The word-level is the model with least error when it is most confident. Finally, one can see that the character-level model and the word-level model have the lowest errors for males and females, respectively, which supports the test results.

Tables 7.29 to 7.31, on page 100 and 101, contain some concrete examples of tweets exemplifying the classification abilities of the models. Tables 7.29 and 7.30 show examples of tweets where the models were 100% confident of their predictions. The predictions were only correct for the examples in the first table, while the latter table contains tweets for which the predictions were wrong. The tables show that there are recurring topics authored by both genders. Among the tweets by females, which were predicted correctly, the content is about appearance and statements showing affection. The content in tweets by males is related to sports, such as soccer and basketball, as well as video games. Apparently, the model has learned these stereotypical characteristics. However, as Table 7.30 shows, neither set of of topics is gender-exclusive and both can appear as content in tweets by the opposite gender as well. Thus, when this happens the models tend to predict the incorrect gender.

Keeping these previous observations in mind, Table 7.28, on page 99, further underpins



(a) (b)



(c)

Figure 7.10.: The graphs of Figure 7.9 plotted individually, along with a red curve indicating error rate

these, as it lists the most frequent tokens in both correctly and incorrectly predicted tweets with respect to the gender of the author. In the set of tokens from correctly predicted tweets, the distinct stereotypical tendencies are present. The models show that they are able to use these content-describing words to characterize the respective genders, but are having difficulties when the same terms, or related terms, are used with the opposite gender. Table 7.31 shows examples of tweets when the models are most uncertain, but still were able to predict the correct gender. The common denominator here is the minimal amount of content-descriptive terms. We also discovered that when the models were 100% confident, more than $\frac{2}{3}$ of the predictions were correct, while less than $\frac{1}{3}$ were incorrect as shown in Table 7.27. Corresponding to results presented earlier, it can also be observed that the error rate of the document-level model is higher at 100% confidence.

Model	**Correct**	**Wrong**	**Total**
Char	38	12	50
Word	37	14	51
Doc	60	29	89
Total	135	55	190

Table 7.27.: Number of tweets classified as correct or incorrect by each model, when the models are 100% confident of predicting the correct gender.

Gender	**Frequent Tokens** (Correct Prediction)	**Frequent Tokens** (Wrong Prediction)
Male	league male model season soccer	tina love model artist palette
Female	:) tina makeup love hair	google buseniess basketball player scoring

Table 7.28.: Most frequent tokens in both correctly and incorrectly predicted tweets, with respect to gender.

Model	Gender	Tweet
Char	F	@Michael5SOS I love it! It sounds good and feels good sorry that I'm so cheesy
Word		@TreTre0 @vonebell.. GORGEOUS DRESS by my favorite designer.
Doc		im finna be wearing dress clothes to work lol, i gotta go shopping and start wearing heels
Char	M	About to play WarZone for the 1st time #Halo5 #Halo5Guardians #Xbox https://t.co/RP9qlgSn5V
Word		@jacksmith23456 @MagicalOezil lol Yaya score 20 league goals , second best player in the league that season and comfortably better than Silva
Doc		It's FC Barca, and Lionel Miss-It missed the penalty.

Table 7.29.: Examples of tweets that are predicted correctly with 100% confidence by the three different models.

Model	Gender	Tweet
Char	F	Often | The weeknd Siet \o/
Word		My player for the month is WALTERS striker of Stokes city 4 scoring 2 own goals missed penalty and chelsea won 4-0 last week
Doc		We're not gonna be assaulted by FanDuel and Draft Kings commercials during basketball season, right?
Char	M	@KissFMUK Been at sat in a hotel with liam and shane writing work reports since 4pm, Luckily #kisstory is keeping us going! thankyou
Word		I think Lancome's Mes Incontournables de Parisienne Makeup Essentials Palette is the smartest palette https://t.co/Vx9fHFzrHw
Doc		Halloween costume idea for those unknowing of what to be: any of those food costumes WITH fishnet stockings AND heels.

Table 7.30.: Examples of tweets that are predicted incorrectly with 100% confidence by the three different models. The gender column shows the true label.

Model	Gender	Tweet
Char	F	I'll go, I'll go and then you go, you go out and spill the truth
Word		Esmond is the dumbest man I've ever met.
Doc		@TimWadephul @CBCMusic @Alanis thank you Tim.
Char	M	I bet the girl wearing the Star Wars hoodie is seeing a lot of action recently
Word		For the next 30 days join me in leaving the biggest impression on this city.
Doc		It's sad how it won't ever be the same between us.

Table 7.31.: Examples of tweets that are predicted correctly with 0% confidence by the three different models.

8. Evaluation and Conclusion

8.1. Evaluation

This section will provide an interpretation and evaluation of the results presented in the previous chapter. To provide a neat presentation of the findings, the section will be divided according to the topics that are covered.

Pre-Processing

The results from the ablation studies showed marginal differences for most of the pre-processing methods that were explored. This transpired to be true for all three models. The small effects of lemmatization in the word-level model were expected because inflected forms of words are close to the root form of the word in vector space. However, the insignificance of the method in the character-level model and the document-level model were more surprising as the expected behavior was that it would make it easier for the models to capture information related to the same theme or topic. The cause of this discrepancy could have several explanations. It may be that the inflected forms of the most gender-characterizing terms do not occur frequently enough to make an impact, and if they do occur frequently, they are likely to be a part of the large document-level vocabulary. It is more difficult to interpret the behavior of the character-level model as the features it detects are embedded in the model. It could be that the short distance between a root word and its inflected forms, measured by the number of letter changes, makes it trivial for the model to understand that the terms are related.

Removing stopwords, as defined by NLTK, was motivated by preventing the models from focusing on terms, which in general provide no information of value for classification. The insignificance of this method could be explained by how ANNs can be made robust to noise by training with noisy data, which would be the case of, e.g., including stopwords. The reasoning is similar for the low returns of removing punctuations and emoticons. In addition, the low frequency of emoticons in general could explain the little impact it has to include them, even though two emoticons, the heart and the regular smiley, had diverging distributions across genders.

Model Topology

The different topologies of the models exemplify the level of inter-connectivity in a deep learning based architecture. Although both the character-level and word-level models process the text as sequences, the fact that these sequences are represented differently

have a major impact on which ANN topologies work optimally for the respective models. While the Bidirectional LSTM led to the best performance at word-level, it was the most poorly performing character-level ANN model. By adding a convolution layer, with an accompanying pooling layer, the performance increased considerably. A reason for this could be that the character-level model learns representations from scratch, while the word-level model uses pre-trained embbeddings. This implies that the character-level model has to "work harder" than the word-level model to learn. The bidirectional nature implies that more information has to be processed at a time. Additionally, the character-level model could be more sensitive towards noise because it has no prior knowledge of the text it processes. With these aspects clarified, it is understandable that a topology containing two regular LSTMs perform better than the Bidirectional LSTM, until convolution is incorporated. With the use of convolution and pooling, a preliminary spatial processing is done, summarizing the most important features. By subsequently applying sequential processing of these, with the Bidirectional LSTM, we achieve a different desirable outcome.

The character-level model topology was tried with the word-level system, but led to poorer results. A possible explanation for this is that the feature pooling leads to information loss, which does not occur with the use of a lone Bidirectional LSTM. However, this cannot be stated with certainty without doing further tests. Another observation was that more complex models, e.g., with the addition of convolutional layers, led to poorer validation loss results, which is an indication of overfitting due to model complexity.

Regularization of Models

All of the models showed signs of overfitting, begging the use of regularization to improve generalization. Dropout proved effective for the character-level and word-level models, but rather ineffective for the document-level model. The application of L1 regularization on the weights did not improve the performance on any models. While L2 regularization did reduce overfitting, this came at the cost of the validation loss, i.e., the validation loss was also higher. It is worth mentioning that because of time constraints, we were unable to conduct extensive experiments with the regularization parameters.

Normalization of input values is generally considered a pre-processing method, for enabling faster and increased capacity of learning. However, we incorporated it in the ANN model through a batch normalization layer in the document-level model. The result of using batch-wise normalization were discouraging, causing instantly diverging loss. Explaining why its use would affect the system as such is somewhat difficult, but could be related the sparsity of the BoW representations, implying that normalization is done over vectors containing mainly zeros. This is likely to meddle with the desired effects.

A peculiar observation, concerning the training of all three models, is that the validation loss is lower than the training loss during the first few epochs. It should generally

not be the case that the classification performance on data the model uses for training is worse than the performance on unseen data. This could indicate that the validation set is not representative of the entire tweet population and somewhat biased towards certain tweet types. The models seem to recognize these early in training, which causes the loss discrepancy.

Vocabulary

Based on the training set, separate vocabularies are defined in the word-level and document-level systems, to be used during model training. Two methods were explored to select vocabulary terms in the document-level system, these being the frequency of terms, i.e., selecting the most frequent terms, and the use a metric to measure how distinguishing the terms are, with regard to gender. Even though the dissimilarity measure seemed to rank the terms more appropriately, i.e., the most distinguishing terms were given higher scores, the results showed that a sufficiently large vocabulary was able to capture the same terms, even if ranked differently. To build the word embedding vocabulary, only different numbers of most frequent terms were used. The results of both systems promote the use of larger vocabulary sizes to capture more information. The noise imposed by the less frequent terms can essentially be ignored, because the networks are able to disregard these. The most prominent restriction imposed with regard to increasing the vocabulary size was the amount of memory needed. However, the experiments also indicate that the amount of further information captured by increasing vocabulary size decreases exponentially.

Though the element of using a vocabulary is similar for the word-level and document-level systems, the manner in which words are represented signify an important difference. In the BoW representation, there is no notion of similarity between words as they do not have a vector space representation, in addition to the occurrence of terms being regarded as independent from each other. With word embeddings, correlations between words are captured in the vector representations. An implication of this is that low-frequency terms can be close to other high-frequency terms in vector space. Leveraging this form of word similarity is not possible in the BoW representation, which merely keeps counts of words.

Shortness of Tweets

Our efforts to improve the BoW representation, in the document-level system, using TF-IDF were rather futile. The reasoning behind using TF-IDF was to normalize the frequency counts in the BoW representations, with regard to relevance, but the results show minimal differences. The reason for this may be the short length of tweets. In longer texts, it is reasonable to believe that the frequencies of words pertaining to the topic of the document will be higher than others. With tweets being short in nature, it is reasonable to believe that the words describing the topic of the tweet may as well occur only once or twice, if the entire tweet is only twelve words long, as was found to be

the average. Influenced by low term frequencies in the individual tweets, TF-IDF may not have the same effects as it has with longer texts. Related to this aspect could also be why only unigrams seem to be useful as features, compared to bigrams and trigrams. The likelihood of any bigrams or trigrams occuring more than a few times in the entire tweet set can be hypothesized to be low, which is underpinned by the results.

Quantitative and Qualitative Evaluation of Test Results

The test results reveal that the overall performance of the word-level system is better than the character-level and document-level systems, with an overall F1-score of 0.604 against 0.592 and 0.585, respectively. While the word-level model is superior on the matter of predicting female authors, the character-level system is coincidentally better at predicting male authors. These characteristics were leveraged to create ensemble stacking models, which aggregate the predictions of the sub-models. Our experimental results show that the best performance is produced combining the character-level model and the word-level model using averaging as aggregation function, i.e., the probability distributions over the classes of each model are averaged. This produces a F1-score of 0.613 on the test set. This is a small increase from the word-level system's score of 0.604, but nevertheless an increase, suggesting that the character-level model's slightly better ability to predict males makes a contribution. Given the superiority of the other two models, the document-level model is unable to contribute, resulting in worse scores when combined with the other models.

Interpreting ANN classifiers is a challenging affair. To better understand the behavior of the models, we examined the probability distributions of the predictions made by the models. Ideally, if the model is confident in its prediction, i.e. the model's estimated likelihood of either class is close to 1.0, the given prediction should rarely be wrong. The analysis supports the test results for the character-level and word-level models, i.e. there is a correspondence between their high confidence predictions and their correct predictions. On the other hand, the same analysis underpins the poor quality of the document-level model as it is more confident than the other models, but has a higher error rate simultaneously.

The brief qualitative analysis presented, indicate that the models are able to make connections between tweet content words, related to certain themes and topics, and gender. Stereotypical characteristics were found, such as males tweeting about sports and females writing about affection and appearance. However, there is a clear indication of the models having problems predicting tweets where these characteristics occur with the opposite gender, i.e., a female tweeting about sports.

Dataset Quality

A comparison of the test results and the validation results reveals a discrepancy, namely that the validation results are significantly better than the test results. Normally, one

would expect these to be only moderately different, as neither the validation set or the test set are used to train the model. Along with the earlier described observation concerning how the validation loss is lower than the training loss during the first few epochs of training, this clearly suggests that the validation set is not as representative of the population of tweets as we would like. Additionally, the mentioned aspects indicate that there is some unwanted correlation between the validation set and the training set. This is not unlikely as the training set and the validation set were constructed by collecting multiple PAN Shared Task datasets from the previous years, while the test set was collected from a separate source. However, given that duplicates are removed during construction of the datasets, it is rather peculiar if a correlation is present. In this case, the correlation may be between tweets by the same author, as these may have certain stylistic similarities. The likelihood of this is supported by the fact that the number of tweets per author is not uniform. Nevertheless, we intially made the assumption that in the case of gender prediction, all tweets are independent of each other, given that they are not duplicates from the same source.

8.2. Discussion

The goal of this thesis was to construct a deep learning system for predicting the gender of tweet authors based on linguistic differences. To define the necessary steps for reaching this goal, four research questions were formulated. This section will address these research questions, presenting the merits, as well as the limitations, of the work.

Research Question 1 *What does the literature establish as the most useful gender identifying linguistic traits?*

With regards to Research Question 1, the literature presents various observations regarding how men and women express themselves differently in writing. These include multiple mentions of how frequencies of certain POS categories diverge, in addition to differing structural aspects, such as lengths of words and sentences. However, our analysis of the dataset did not support these claims for the domain of tweets. Considering how the concept of tweets promote short burst of information, it is not all that surprising that the distribution of tweet lengths is uniform across genders. Therefore, our focus was turned to mainly capturing semantic and content-based features. Most studies seem to agree that these features contribute significantly in author profiling systems, with n-grams of content terms being popular. Our work reflects this as all of the classification systems focus on tweet content, even though the character-level and word-level models may implicitly capture structural elements, such as lengths of sentences, as a result of how they process text as sequences of characters and words, respectively. Corresponding to the findings of previous studies presented in Chapter 4, our models made use of content terms to draw stereotypical connections between certain topics and genders.

Research Question 2 *How can texts be represented in deep learning systems to capture meaningful information?*

On the matter of text representation, which concerns Research Question 2, multiple approaches have been explored. The implemented systems focus on the content of tweets at different granularities, these being at the level of characters, words and the document as a whole. Different numerical representations have been used to represent the features of the texts. While characters are represented as one-hot vectors, pre-trained GloVe embeddings have been used to represent words. A more old-fashioned approach was followed for the document-level system, for which BoW representations were used. The motivation for constructing multiple systems with different foundations was how the particular approaches could be able to capture different aspects of the tweets, thus characterizing different traits of the genders. Our results do indeed indicate that the models have varying competence with regard to predicting instances of each gender, exemplified by how the character-level model is better at predicting female authors than the other models, while the character-level excels at predicting male authors. Additionally, we are able to compare the old-fashioned architecture of the document-level model with the more advanced architectures of the character-level and word-level models, showing that tediously handcrafted features are outperformed by features learned by deep learning models.

Research Question 3 *What types of deep learning models are viable for processing and classifying tweets?*

Based on reviewed ANN theory and research, Convolutional Neural Networks and LSTMs have been used in the architectures because of their viability in NLP. This pertains to Research Question 3. With the mentioned network types requiring spatial and sequential representation of the data, respectively, they are only used in the character-level and word-level models. In the document-level model a regular feedforward network is used to process the BoW representations of tweets, disregarding the dependencies between the terms. This latter aspect of the document-level model may be a major cause of the weak performance, compared to the other models.

Research Question 4 *Can multiple deep learning architectures be used in combination to recognize different characteristics of tweets?*

Pertaining to Research Question 4, the results showed that the models can be used collaboratively to increase the predictive power, and the overall performance is better than the performance of the baseline models. The F1-score difference between the best stacking model and Logistic Regression, Naïve Bayes and Random Forest are respectively 2.6%, 6.2% and 3.6%. Though we can observe that the models are able to learn, it

can be argued that they are not accurate enough for real-world applications, considering the results. Previous studies have shown higher accuracy for texts of other genres, but tweets, in particular, have the quality of being short by nature. The question then arises as to if the goal of creating a model applicable for the real world, in the domain of tweets, is unrealistic. Intuitively, one may argue that tweets are too short to extract gender-exclusive information, or more precisely a set of characteristics providing enough information to predict gender. The problem may be more feasible if a stricter threshold is set for the length of tweets to be predicted, e.g., by disregarding any tweet less than the average or median length of the dataset, though one can always argue that better models can possibly overcome such restrictions.

Though usable results have been produced in the course of this project, there are a number of aspects that have limited our research. The issues related to the dataset suggest that it could have been further quality controlled. In addition to the points made in the previous section, there is the fact that even though the training data obtained from the PAN Shared Task were labelled as English, during parsing of the data we found traces of several other languages. To further investigate if a correlation between tweets by the same author is present, it is possible to construct a separate dataset, containing a single tweet per author, and train a model, though this was not tested. On the matter of selecting features based on the statistic analysis of the data, an explicit threshold should have been defined to make the selection process more structured and rigorous.

In the case of word embeddings, GloVe was used, though other pre-trained embeddings could have been explored, such as pre-trained word2vec on tweets and FastText[1] by Facebook. Additionally, experiments could have been conducted for determining the optimal sequence length limit in the character-level and word-level models. Though a limit was determined based on the average character length and word length of tweets, it could be that the noise imposed by padding text representations can be disregarded when compared to the extra information gained by increasing the sequence length to maximum tweet length.

It is worth mentioning that we may have miscalculated the work involved with developing three fundamentally different deep learning models. Thus, because of time constraints, we were unable to perform extensive hyperparameter testing to perform a structured optimization procedure. This may have impaired the quality of the models, including the autoencoder for dimensionality reduction. However, the use of autoencoders was not initially planned and this path was pursued after discovering the level of sparsity in the document-level representations. With regard to the baseline models, it may seem odd that Support Vector Machines (SVMs) were not used when they are clearly present in state-of-the-art implementations. The fact is that we encountered some difficulties with our SVM implementation, which made us leave out the model. Additionally, it should be noted that the baseline models were not optimized. Thus, there is a form of bias

[1] https://research.fb.com/projects/fasttext/

when comparing the deep learning models to these.

8.3. Contributions

As a part of this thesis, three different classification systems, processing text at different granularities, were developed to classify gender of tweet authors. These can be summarized as follow:

1. A Convolutional Bidirectional LSTM model processing tweets at the character-level, representing these as one-hot vectors. This model achieved a F1-score of 0.592.

2. A Bidirectional LSTM model using pre-trained GloVe word embeddings to process tweets at the word-level. This model achieved a F1-score of 0.604.

3. A feedforward model which processes tweets at the document-level. This model focuses more on feature engineering, for which it is developed using more traditional NLP methods and Bag of Words is used as tweet representation. This model achieved a F1-score of 0.585.

Additionally, several ensemble models, utilizing stacking, were explored to combine the predictive powers of the three separately trained sub-models. The gender predictions of the sub-models were combined using aggregation functions, these being averaging the predictions of the sub-models, choosing the most confident prediction and voting. The overall best model was produced by combining the character-level and word-level models to achieve an overall F1-score of 0.613, beating the baseline models, of which Logistic Regression had the best result with a score of 0.587. Thus we show that deep learning architectures can learn different aspects of tweets and be combined to collaboratively produce higher quality predictions.

Our statistical analysis of tweets stand in contrast to previous findings in text of other genres, where it has been shown that, e.g., frequencies of certain part-of-speech can be considered gender-specific. The results and the development process indicate that models based on manual feature engineering are inferior to well-constructed models that are able to implicitly learn representations. Our experiments with the character-level and word-level models have shown that provided well-defined representations of texts and well-constructed deep learning architectures, better gender-specific features can be learned, as opposed to manually handpicking features based on statistical analysis. The latter was done for the document-level model.

With ANNs, interpretation of results is challenging, as the models make implicit abstractions. We provide a qualitative analysis of the models, showing that the level of confidence the models have in their predictions is connected to their success rate. We have also composed a review of studies on linguistic gender-based differences in tweets and a study of the current state-of-the-art approaches.

8.4. Future Work

In addition to the results produced, we encountered several challenges during the course of this thesis which inspired ideas that can be explored further. Additionally, with the vast amount of possibilities within Natural Language Processing, we were forced to prioritize the methods and some had to be omitted. Based on these aspects, this section presents suggestions for how the conducted research can be further improved upon.

Named Entity Recognition

Named Entity Recognition (NER) is a method for identifying and classifying proper names into predefined categories. It would be interesting to explore the use NER, for the purpose of using frequencies of named entities as features. For simplicity, this can be done by using existing NER systems adapted for tweets. Ritter et al. (2011) provide a system, which is able to categorize up to 10 types of entities, these being: *person*, *geo-loc*, *company*, *facility*, *product*, *band*, *sportsteam*, *movie*, *tv-show*, *other*.

Autoencoders

Autoencoders were used in our work to reduce the dimension of document-level feature representations, though we barely scratched the surface of the area. This was motivated by the amount of sparsity in the high-dimensional Bag of Words representations. By creating a compressed representation of the tweets, we hoped to increase the classification accuracy. However, we were not able to produce lower-dimensional representations improving the results and further work can be done in this area. Additionally, it could also be worth applying autoencoders in combination with the character-level and word-level models to produce higher quality representation of tweets.

Utilizing Tweet Specific Characteristics

Hashtags and mentions are considered term specific to the domain of Internet and Twitter. In our architecture, such terms are replaced with placeholders, acknowledging their presence, but disregarding the specific term. Since these are integral parts of the Twitter domain, they should be leveraged for what they are worth. Being a construct designed to describe topics of tweets, hashtags are often used and shared by a wide range of users. Thus, it could be worth exploring the usefulness of these, with regard to predicting gender. Another commonly occurring element on the Internet, tweets included, is elongation, or repeated characters in words. In the proposed models, this could be a source of noise, motivating the process of eliminating elongated terms by transforming them to the corresponding dictionary words. On the other hand, it can also be the case that elongation can be used as a feature if it happens to be gender-specific.

Training a Combined Learner

The stacking model developed as a part of this thesis only aggregated the predictions of pre-trained models using aggregation functions with the class probability distribution as input. However, when stacking is employed, it is common to further train the combined learner, using a simple model. It would be interesting to optimize this process using deep learning methods. Related to this is the possibility of merging the different deep learning architectures into one architecture, training the sub-models and the final model simultaneously.

Further Optimization of Models

Since we were unable to perform a structured optimization of all models, it can be considered trivial to suggest this. In addition, it would be desirable to compare the deep learning models with a Support Vector Machine model, which we were unable to do as a part of this thesis. To be able to further substantiate the superiority of deep learning methods, the baseline models should be optimized as well.

Improving Quality of Handcrafted Features

The basis of the feature engineering process in this thesis was laid out in Chapter 5, where a statistic analysis of the training data was performed. Using these results, certain features of the tweets were disregarded, such as part-of-speech counts. An important aspect of this is that the data analysis only showed how much such features would contribute by themselves, and not alongside other features. Considering this, it is possible to improve the feature selection process by calculating the pairwise correlation between features, with regard to gender prediction. This implies that a feature, which shows no potential on its own, may do so if considered together with another feature.

Prediction of other Author Attributes

In this thesis, only prediction of gender was explored, though author profiling includes other personal attributes of authors, such as age, personality traits and native language. Given appropriately annotated datasets, the architectures of the character-level and word-level models should enable us to train them for predicting other author characteristics, without major adjustments, though optimal performance is dependent on hyperparameter tuning. This is a result of the models learning representations of tweets implicitly in the ANN, without manual feature engineering.

Bibliography

Madhulika Agrawal and Teresa Gonçalves. Age and gender identification using stacking for classification. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September*, pages 785–790, 2016. URL `http://ceur-ws.org/Vol-1609/16090785.pdf`.

Shlomo Argamona, Moshe Koppelb, Jonathan Finec, and Anat Rachel Shimonib. Gender, genre, and writing style in formal written texts. *Text*, 23:3, 2003.

Shaina Ashraf, Hafiz Rizwan Iqbal, and Rao Muhammad Adeel Nawab. Cross-genre author profile prediction using stylometry-based approach. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September*, pages 992–999, 2016. URL `http://ceur-ws.org/Vol-1609/16090992.pdf`.

Yoshua Bengio, Patrice Simrad, and Paolo Frasconi. *Learning Long-Term Dependencies with Gradient Descent is Difficult*, 1994.

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

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE - Institute of Electrical and Electronics Engineers Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

Ivan Bilan and Desislava Zhekova. Caps: A cross-genre author profiling system, 2016.

Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009. ISBN 0596516495, 9780596516499.

Ann Bodine. Sex differentiation in language. *Language and sex: Difference and dominance*, pages 130–151, 1975.

Konstantinos Bougiatiotis and Anastasia Krithara. Author profiling using complementary second order attributes and stylometric features. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September*, pages 836–845, 2016. URL `http://ceur-ws.org/Vol-1609/16090836.pdf`.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

John D. Burger, John C. Henderson, George Kim, and Guido Zarrella. Discriminating gender on twitter. In *EMNLP - Conference on Empirical Methods in Natural Language Processing, Edinburgh, UK, 27-29 July, 2011.*, pages 1301–1309. Association for Computational Linguistics, 2011.

Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *EMNLP - Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October, 2014.*, pages 103–111. Association for Computational Linguistics, 2014.

François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 05-09 July, 2008.*, volume 307 of *Association for Computing Machinery International Conference Proceeding Series*, pages 160–167. Association for Computing Machinery, 2008.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20 (3):273–297, 1995.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.

Li Deng, Dong Yu, et al. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387, 2014.

Rodwan Bakkar Deyab, José Duarte, and Teresa Gonçalves. Author profiling using support vector machines. In *CLEF - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016.*, volume 1609 of *CEUR Workshop Proceedings*, pages 805–814. CEUR-WS.org, 2016.

Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, and William W. Cohen. Tweet2vec: Character-based distributed representations for social media. *CoRR - Computing Research Repository*, abs/1605.03481, 2016.

Daniel Dichiu and Irina Rancea. Using machine learning algorithms for author profiling in social media. In *CLEF - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016.*, volume 1609 of *CEUR Workshop Proceedings*, pages 858–863. CEUR-WS.org, 2016.

Sara El Manar El Bouanani and Ismail Kassou. Authorship analysis studies: A survey. *International Journal of Computer Applications*, 86(12), 2014.

Aparna Garimella and Rada Mihalcea. Zooming in on gender differences in social media. In *Proceedings of the Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media, Osaka, Japan, 12 December, 2016.*, pages 1–10, 2016.

Felix A Gers and Jürgen Schmidhuber. Recurrent nets that time and count. In *Proceedings of the IJCNN'2000 - , International Joint Conference on Neural Networks, Como, Italy, 24-27 July 2000*, volume 3, pages 189–194. Institute of Electrical and Electronics Engineers, 2000.

Maite Giménez, Delia Irazú Hernández, and Ferran Pla. Segmenting target audiences: Automatic author profiling using tweets. In *Proceedings of CLEF - Conference and Labs of the Evaluation forum, Toulouse, France, 8-11 September, 2015.*, 2015.

Lee Gomes. Machine-learning maestro Michael Jordan on the delusions of big data and other huge engineering efforts, 2014. URL `http://spectrum.ieee.org/robotics/artificial-intelligence/machinelearning-maestro-michael-jordan-on-the-delusions-of-big-data-and-other-huge-engineering-efforts`. [Online; accessed 03-May-2014].

Bruno Gonçalves and David Sánchez. Crowdsourcing dialect characterization through twitter. *PloS one*, 9(11):e112074, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016.

Alex Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer, 2012.

Adelaide Haas. Male and female spoken language differences: Stereotypes and evidence. *Psychological Bulletin*, 86(3):616, 1979.

Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

Sepp Hochreiter. *Untersuchungen zu dynamischen neuronalen Netzen*. PhD thesis, Diploma thesis, T.U. München, 1991.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Clayton J. Hutto and Eric Gilbert. A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the Eighth International Conference on Weblogs and Social Media, ICWSM 2014, Ann Arbor, Michigan, USA, June 1-4, 2014.*, 2014. URL `http://www.aaai.org/ocs/index.php/ICWSM/ICWSM14/paper/view/8109`.

Diederik Kingma and Jimmy Ba. A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Moshe Koppel and Jonathan Schler. Exploiting stylistic idiosyncrasies for authorship attribution. In *Proceedings of IJCAI'03 Workshop on Computational Approaches to Style Analysis and Synthesis*, volume 69, page 72, 2003.

Moshe Koppel, Shlomo Argamon, and Anat Rachel Shimoni. Automatically categorizing written texts by author gender. *Literary and Linguistic Computing*, 17(4):401–412, 2002.

Cheris Kramer. Women's speech: Separate but unequal? *Quarterly Journal of Speech*, 60(1):14–24, 1974.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Yann Le Cun, LD Jackel, B Boser, JS Denker, HP Graf, I Guyon, D Henderson, RE Howard, and W Hubbard. Handwritten digit recognition: Applications of neural network chips and automatic learning. *IEEE Communications Magazine*, 27(11):41–46, 1989.

Yann LeCun. *Modèles connexionnistes de l'apprentissage*. PhD thesis, Paris 6, 1987.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

Matthias Liebeck, Pashutan Modaresi, and Stefan Conrad. Evaluating safety, soundness and sensibleness of obfuscation systems. In *CLEF - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016*, volume 1609 of *CEUR Workshop Proceedings*, pages 920–928. CEUR-WS.org, 2016.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*, 2015.

Adrian Pastor Lopez-Monroy, Manuel Montes-y Gómez, Hugo Jair Escalante, Luis Villasenor-Pineda, and Esaú Villatoro-Tello. INAOE's - National Institute of Astrophysics, Optics and Electronics participation at PAN'13: Author profiling task. In *CLEF - Conference and Labs of the Evaluation forum, Valencia, Spain, 23-26 September, 2013*, 2013.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

Ilia Markov, Helena Gómez-Adorno, Grigori Sidorov, and Alexander Gelbukh. Adapting cross-genre author profiling to language and corpus. In *Proceedings of the CLEF - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016.*, pages 947–955, 2016.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b.

Marvin Minsky and Seymour Papert. Perceptrons., 1969.

Anthony Mulac, Lisa B Studley, and Sheridan Blau. The gender-linked language effect in primary and secondary students' impromptu essays. *Sex Roles*, 23(9-10):439–470, 1990.

Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

Mart Busger op Vollenbroek, Talvany Carlotto, Tim Kreutz, Maria Medvedeva, Chris Pool, Johannes Bjerva, Hessel Haagsma, and Malvina Nissim. Gronup: Groningen user profiling, 2016.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP - Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October, 2014.*, volume 14, pages 1532–1543, 2014.

Oliver Pimas, Andi Rexha, Mark Kröll, and Roman Kern. Profiling microblog authors using concreteness and sentiment, 2016.

R Rajendra Prasath. Learning age and gender using co-occurrence of non-dictionary words from stylistic variations. In *International Conference on Rough Sets and Current Trends in Computing, Warsaw, Poland, June 28-30, 2010.*, pages 544–550. Springer, 2010.

Francisco Rangel, Paolo Rosso, Ben Verhoeven, Walter Daelemans, Martin Potthast, and Benno Stein. Overview of the 4th author profiling task at pan 2016: cross-genre evaluations. *Working Notes Papers of the CLEF - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016.*, 2016.

T Raghunadha Reddy, B Vishnu Vardhan, and P Vijayapal Reddy. A survey on authorship profiling techniques. *International Journal of Applied Engineering Research*, 11 (5):3092–3102, 2016.

Alan Ritter, Sam Clark, and Oren Etzioni. Named entity recognition in tweets: an experimental study. In *EMNLP - Conference on Empirical Methods in Natural Language Processing, Edinburgh, UK, 27-29 July, 2011.*, pages 1524–1534. Association for Computational Linguistics, 2011.

Robin Lakoff. Language and woman's place. *Language in society*, 2(01):45–79, 1973.

Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, DTIC Document, 1961.

Hassan Saif, Miriam Fernandez, Yulan He, and Harith Alani. On stopwords, filtering and data sparsity for sentiment analysis of twitter, 2014.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W Pennebaker. Effects of age and gender on blogging. In *AAAI spring symposium: Computational approaches to analyzing weblogs, Palo Alto, California, 27–29 March, 2006*, volume 6, pages 199–205, 2006.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

Shamane Siriwardhana. *Why Convolutional Neural Networks.* `https://www.linkedin.com/pulse/why-convolutioanl-neural-networks-shamane-siriwardhana`, 2016. [Online; accessed 01-May-2017].

Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Efstathios Stamatatos. A survey of modern authorship attribution methods. *JASIST - Journal of the Association for Information Science and Technology*, 60(3):538–556, 2009.

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

Paul John Werbos. Beyond regression: new tools for prediction and analysis in the behavioral science. *Ph. D. Thesis, Harvard University*, 1974.

Zhixiang Eddie Xu, Minmin Chen, Kilian Q. Weinberger, and Fei Sha. An alternative text representation to TF-IDF and bag-of-words. *CoRR*, abs/1301.6770, 2013.

Xiang Zhang and Yann LeCun. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*, 2015.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.

A. Artificial Neural Network Theory

Activation Functions

ReLU

$$relu(z) = \max(0, z) \tag{A.1}$$

Softmax

$$softmax(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{A.2}$$

Sigmoid

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{A.3}$$

Tanh

$$tanh(z) = 2\sigma(2z) - 1 \tag{A.4}$$

Where $\sigma(z)$ is the sigmoidal function defined in Equation (A.3).

Loss Function

Cross Entropy

$$\mathcal{L}(X, Y) = -\frac{1}{n} \sum_x \sum_j [y_j^{(i)} ln\ a_j(x^{(i)}) + (1 - y_j^{(i)}) ln(1 - a_j(x^{(i)})] \tag{A.5}$$

The summations are aggregated over training samples and the individual neurons in the output layer. X represent the input samples and Y are the true labels. $a(x)$ is the output of the neurons in the neural network given the input x:

$$a = \sum_j w_j x_j \tag{A.6}$$

B. Libraries, API´s and Hardware

In this chapter, a brief overview is provided of the most important libraries and API´s that were used for implementations in this work. Additionally, a specification of the hardware utilized for model training is described.

Keras

Keras (Chollet et al., 2015), is a high-level open source API for constructing, training and optimizing Artifical Neural Networks (ANNs). The API is written in Python and can run on top of Tensorflow[1] or Theano.[2] Keras provides higher-level abstractions of complex ANN architectures, which allows for faster and easier prototyping. It facilitates standard feedforward neural networks, as well as more complex network such as Convolutional Neural Networks and Recurrent Neural Networks. For this thesis, Tensorflow was used as the backend engine.

Natural Language Toolkit (NLTK)

The Natural Language Toolkit (NLTK) by Bird et al. (2009) is a Natural Language Processing toolkit for Python. It provides an interface to various corpora and a myriad of libraries for tokenization, stemming and other NLP methods and models, such as Bag of Words and Term Frequency-Inverse Document Frequency.

Scikit-learn (sklearn)

Scikit-learn (Pedregosa et al., 2011) is an open source Machine Learning library for the programming language Python. Scikit-learn contains support a range of topics within Machine Learning. The library has been used for constructing the base line models respectively Logistic Regression, Naïve Bayes and Random Forests in the master thesis.

[1] https://www.tensorflow.org/
[2] http://deeplearning.net/software/theano/

Hardware

The Artifical Neural Network models in this thesis were trained using NVIDIA Titan X GPUs[3].

[3]https://www.nvidia.com/en-us/geforce/products/10series/titan-x-pascal/

C. Special Words and Abbreviations

Stopwords

Stopwords				
a	do	into	ours	too
about	does	is	ourselves	under
above	doesn	isn	out	until
after	doing	it	over	up
again	don	its	own	ve
against	down	itself	re	very
ain	during	just	s	was
all	each	ll	same	wasn
am	few	m	shan	we
an	for	ma	she	were
and	from	me	should	weren
any	further	mightn	shouldn	what
are	had	more	so	when
aren	hadn	most	some	where
as	has	mustn	such	which
at	hasn	my	t	while
be	have	myself	than	who
because	haven	needn	that	whom
been	having	no	the	why
before	he	nor	their	will
being	her	not	theirs	with
below	here	now	them	won
between	hers	o	themselves	wouldn
both	herself	of	then	y
but	him	off	there	you
by	himself	on	these	your
can	his	once	they	yours
couldn	how	only	this	yourself
d	i	or	those	yourselves
did	if	other	through	

Table C.1.: List of stopwords.

Part-of-speech tags

Table C.2 displays abbreviations of POS tags used with NLTK tagger module. The example are acquired from the available webpage[1] from the book "Natural Language Processing with Python" (Bird et al., 2009, chapter 5) under the terms: Creative Commons Attribution Noncommercial No-Derivative-Works 3.0 US License.[2]

POS-tags		
Tag	**Meaning**	**English Examples**
ADJ	adjective	new, good, high, special, big, local
ADP	adposition	on, of, at, with, by, into, under
ADV	adverb	really, already, still, early, now
CONJ	conjunction	and, or, but, if, while, although
DET	determiner	the, a, some, most, every, no, which
NOUN	noun	year, home, costs, time, Africa
NUM	numeral	twenty-four, fourth, 1991, 14:24
PRT	particle	at, on, out, over per, that, up, with
PRON	pronoun	he, their, her, its, my, I, us
VERB	verb	is, say, told, given, playing, would
.	punctuation marks	. , ; !
X	other	ersatz, esprit, dunno, gr8, univeristy

Table C.2.: Abbreviation of a set of POS tags used with NLTK tagger module.

[1] http://www.nltk.org/book_1ed/
[2] https://creativecommons.org/licenses/by-nc-nd/3.0/us/

POS-tags	
Tag	**Description**
$:	dollar
”:	closing quotation mark
(:	opening parenthesis
):	closing parenthesis
,:	comma
–:	dash
.:	sentence terminator
:	colon or ellipsis
CC:	conjunction, coordinating
CD:	numeral, cardinal
DT:	determiner
EX:	existential there
FW:	foreign word
IN:	preposition or conjunction, subordinating
JJ:	adjective or numeral, ordinal
JJR:	adjective, comparative
JJS:	adjective, superlative
LS:	list item marker
MD:	modal auxiliary
NN:	noun, common, singular or mass
NNP:	noun, proper, singular
NNPS:	noun, proper, plural
NNS:	noun, common, plural
PDT:	pre-determiner
POS:	genitive marker
PRP:	pronoun, personal
PRP$:	pronoun, possessive
RB:	adverb
RBR:	adverb, comparative
RBS:	adverb, superlative
RP:	particle
SYM:	symbol
TO:	"to" as preposition or infinitive marker
UH:	interjection
VB:	verb, base form
VBD:	verb, past tense
VBG:	verb, present participle or gerund
VBN:	verb, past participle
VBP:	verb, present tense, not 3rd person singular
VBZ:	verb, present tense, 3rd person singular
WDT:	WH-determiner
WP:	WH-pronoun
WP$:	WH-pronoun, possessive
WRB:	Wh-adverb
“:	opening quotation mark

Table C.3.: Abbreviation of all POS tags used with NLTK tagger module.

D. Additional Experimental Results and Figures

Data Characteristics

Part-of-speech tags

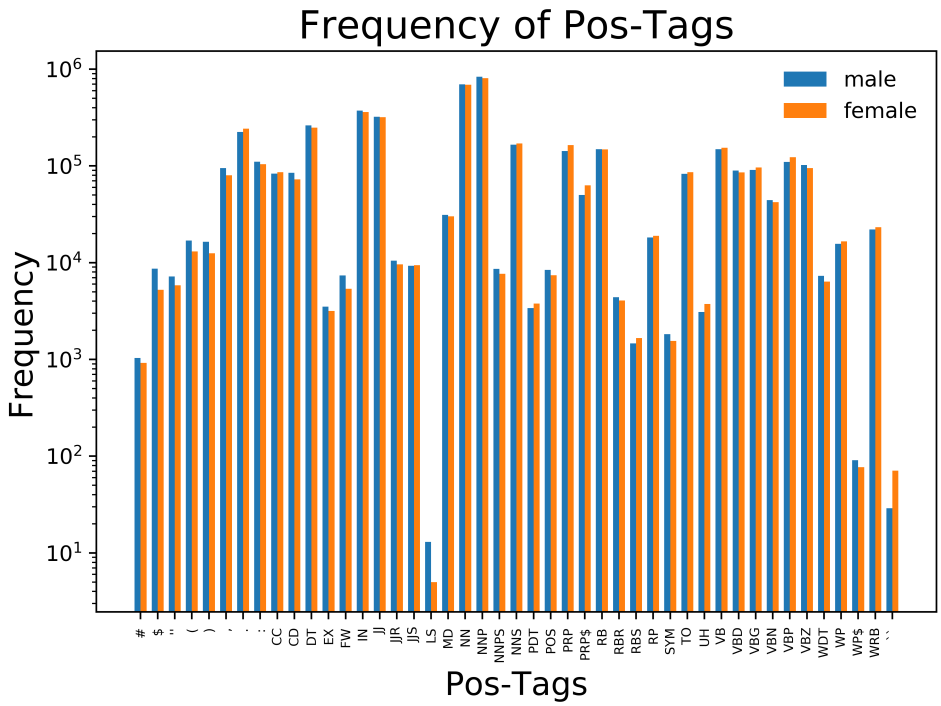


Figure D.1.: Frequency of POS tags among gender.

Stopwords

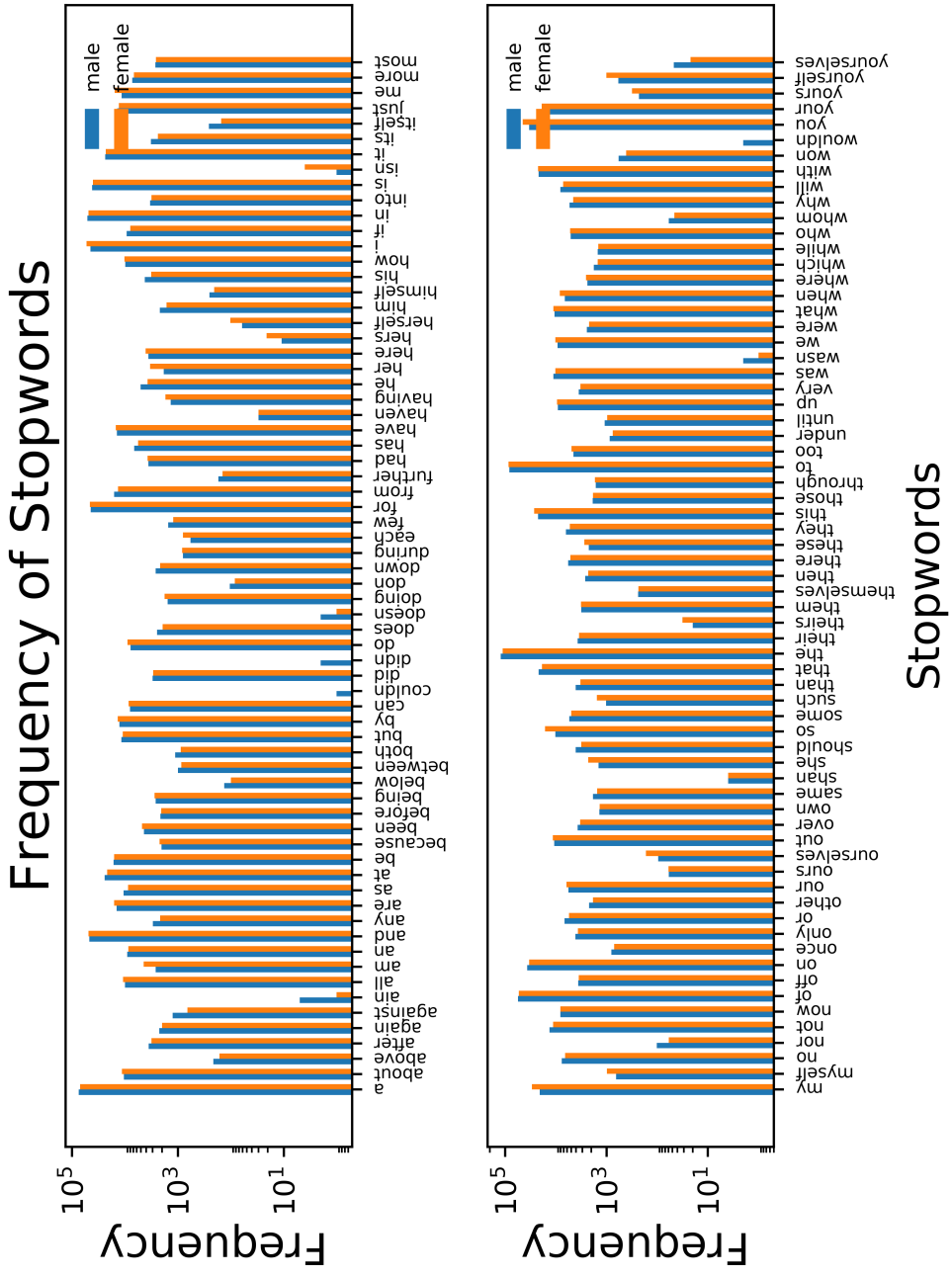


Figure D.2.: Frequency of stopwords among gender.

Test Results

Average Confidence - Character- and Document-Level			
	Female	**Male**	**Overall**
Precision	0.613	0.583	0.598
Recall	0.607	0.589	
F1-score	0.610	0.586	

Table D.1.: Test scores when stacking character- and document-level systems, using confidence average as aggregation function.

Average Confidence - Word- and Document-Level			
	Female	**Male**	**Overall**
Precision	0.611	0.592	0.602
Recall	0.636	0.567	
F1-score	0.623	0.579	

Table D.2.: Test scores when stacking word- and document-level systems, using confidence average as aggregation function.

Logistic Regression			
	Female	**Male**	**Overall**
Precision	0.616	0.564	0.591
Recall	0.538	0.641	
F1-score	0.574	0.600	

Table D.3.: Test scores of Logistic Regression.

Naïve Bayes			
	Female	**Male**	**Overall**
Precision	0.556	0.599	0.577
Recall	0.805	0.311	
F1-score	0.658	0.410	

Table D.4.: Test scores of Naïve Bayes.

Random Forests			
	Female	**Male**	**Overall**
Precision	0.573	0.529	0.551
Recall	0.502	0.599	
F1-score	0.535	0.561	

Table D.5.: Test scores of Random Forests.

Visualization of GloVe Embeddings with t-SNE

This section presents visualizations of the GloVe embeddings used in the vocabulary of the word-level system. For this t-Distributed Stochastic Neighbor Embedding (t-SNE) is used. t-SNE is a machine learning algorithm for dimensionality reduction developed by Geoffrey Hinton and Laurens van der Maaten (Maaten and Hinton, 2008). The method is suitable for mapping high-dimensional data to low-dimensional space, such as 2D or 3D, for more convenient visualization. In contrast to Principal Component Analysis (PCA), which is also a highly used dimensionality reduction method, t-SNE is non-linear. An important aspect of dimensionality reduction is to preserve structure and relations between data points when mapped to a lower dimension. Briefly put, t-SNE constructs one similarity matrix for the original high-dimensional data points and another matrix for the points in the low-dimensional map. Then it proceeds to minimize the Kullback-Leibler divergence Kullback and Leibler (1951) between the two matrices. This is essentially a measure of difference between probability distributions.

Figure D.3 visualizes the word embeddings of the 500 most frequent terms in the vocabulary of the word-level system. The large vector space and the many data points makes it difficult to see the relations between the terms. Therefore, Figure D.4 is provided for convenience. Here, we can see how semantically similar terms are located close to each other in embedding space.

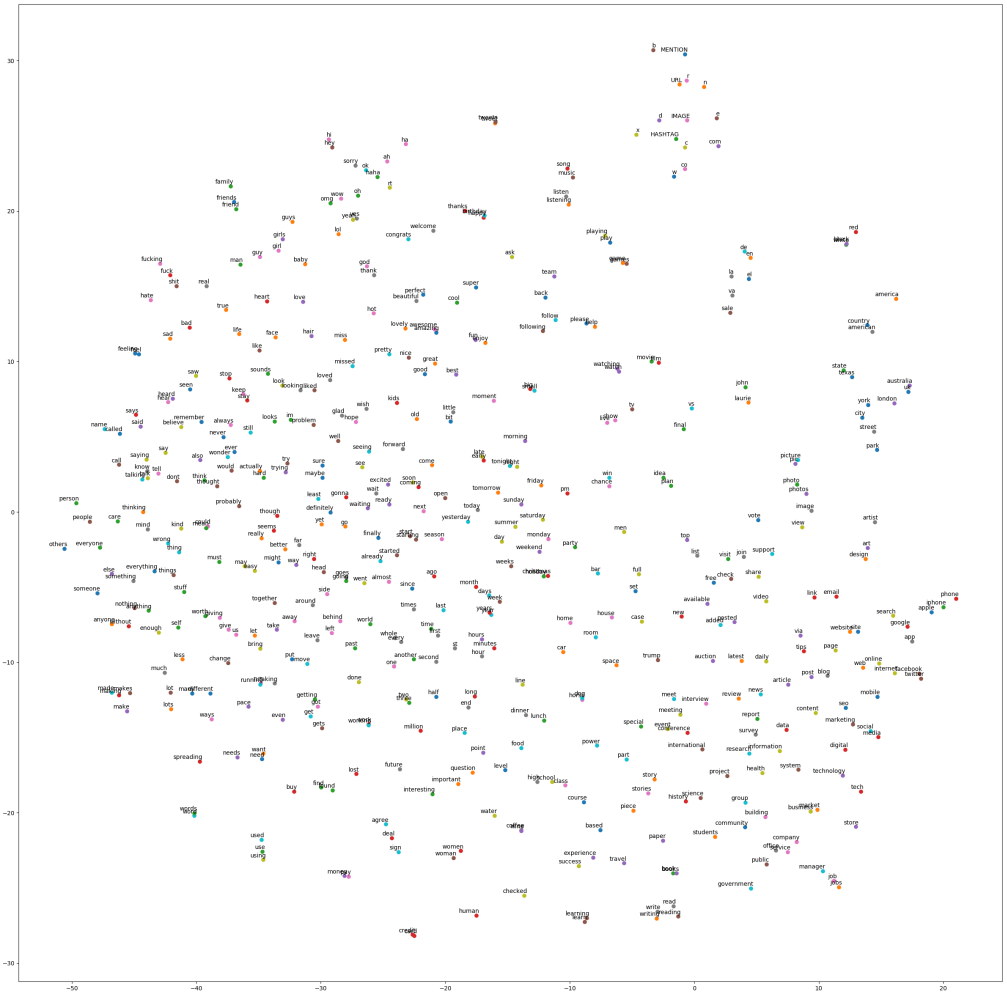


Figure D.3.: Visualization of the 500 most frequent words in the vocabulary of the word-level system. The high-dimensional vectors is mapped to 2-dimensional space using t-SNE.

Figure D.4.: Subset of word embedding clusters from Figure D.3.