User Modeling in Social Media: Gender and Age Detection

by

Saman Daneshvar

A thesis submitted in conformity with the requirements
for the degree of Master of Science

Systems Science Program
School of Electrical Engineering and Computer Science
University of Ottawa

User Modeling in Social Media: Gender and Age Detection

Saman Daneshvar

Master of Science

Systems Science Program
School of Electrical Engineering and Computer Science
University of Ottawa

2019

Abstract

Author profiling is a field within Natural Language Processing (NLP) that is concerned with identifying various characteristics and demographic factors of authors, such as gender, age, location, native language, political orientation, and personality by analyzing the style and content of their writings. There is a growing interest in author profiling, with applications in marketing and advertising, opinion mining, personalization, recommendation systems, forensics, security, and defense.

In this work, we build several classification models using NLP, Deep Learning, and classical Machine Learning techniques that can identify the gender and age of a Twitter user based on the textual contents of their correspondence (tweets) on the platform.

Our SVM gender classifier utilizes a combination of word and character n-grams as features, dimensionality reduction using Latent Semantic Analysis (LSA), and a Support Vector Machine (SVM) classifier with linear kernel. At the PAN 2018 author profiling shared task, this model achieved the highest performance with 82.21%, 82.00%, and 80.90% accuracy on the English, Spanish, and Arabic datasets, respectively. Our age classifier was trained on a dataset of 11,160 Twitter users, using the same approach, though the age classification experiments are preliminary.

Our Deep Learning gender classifiers are trained and tested on English datasets. Our feedforward neural network consisting of a word embedding layer, flattening, and two densely-connected layers achieves 79.57% accuracy, and our bidirectional Long Short-Term Memory (LSTM) neural network achieves 76.85% accuracy on the gender classification task.

Dedication

To my parents, brother and sister

for their immeasurable love…

and for always being there for me.

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Dr. Diana Inkpen for her invaluable research direction and mentorship, continued patience, and generous support throughout my research. I will forever be indebted to her for believing in my capabilities, and for being a role model of intellect, hard work, and humility.

Special thanks to Dr. Kenton White for his mentorship, for sharing his extensive knowledge and creative ideas, and for providing the dataset for this work. Thanks to Shainen Davidson from Advanced Symbolics Inc. (ASI) for his assistance with obtaining and preparing the dataset.

Many thanks to Ehsan Amjadian, Parinaz Sobhani, Ali Pesaranghader, and Jelber Sayyad Shirabad for their priceless insights and guidance, and for setting an exceptional example for other researchers. Your wisdom and dedication has inspired me every step of the way.

Thanks to Dr. Chris Tanasescu for his gracious support, and to Prasadith Buddhitha, Haifa Alharthi, Vaibhav Kesarwani, and all my fellow members of the NLP lab, for their help and encouragement.

Special thanks to my committee members, Dr. Amal Zouaq and Dr. WonSook Lee, for their time and attention to this dissertation, and their constructive feedback.

Thanks to all of the guest speakers at the TAMALE seminars and NLP group meetings, namely Dr. Xiaodan Zhu, and Dr. Saif Mohammad from the National Research Council of Canada (NRC), for sharing their cutting-edge research in the field of NLP and ML.

Thanks to the Natural Sciences and Engineering Research Council of Canada (NSERC) and the University of Ottawa for funding this research. This research was enabled in part by support provided by Compute Canada[1].

Last but not least, I would like to thank my family for providing me with unfailing support and continuous encouragement throughout my academic journey, and for giving meaning to my life. None of my accomplishments would have been possible without them.

[1] https://www.computecanada.ca/

Table of Contents

List of Tables

List of Figures

List of Appendices

Abbreviations

ANN	Artificial Neural Network
API	Application Programming Interface
ARC	Advanced Research Computing
BERT	Bidirectional Encoder Representations from Transformers
BoW	Bag of Words
CLEF	Conference and Labs of the Evaluation Forum
CNN	Convolutional Neural Network
DL	Deep Learning
GIF	Graphics Interchange Format
GloVe	Global Vectors
GRU	Gated Recurrent Unit
IR	Information Retrieval
LDSE	Low Dimensionality Statistical Embedding
LIWC	Linguistic Inquiry and Word Count
LOOCV	Leave-One-Out Cross-Validation
LR	Logistic Regression
LSA	Latent Semantic Analysis
LSTM	Long Short Term Memory
ML	Machine Learning
NB	Naïve Bayes

NLP	Natural Language Processing
NLTK	Natural Language Toolkit
NRC	National Research Council of Canada
POS	Part-of-speech
ReLU	Rectified Linear Unit
RF	Random Forest
RNN	Recurrent Neural Network
SVM	Support Vector Machines
TAMALE	Text Analysis and Machine Learning
TF-IDF	Term frequency–inverse document frequency
URL	Universal Resource Locator
XML	Extensible Markup Language

Chapter 1

1 Introduction

1.1 Motivation

The rise of social media in the past decade has introduced new forms of social interaction. With daily active users reaching 126 million on Twitter [Shaban 2019] and 1.56 billion on Facebook ["Company Info" 2019], these social media platforms have become an invaluable source of data for studying the human society at scale. The increasing amount of unstructured textual data generated on the social media created the need for Natural Language Processing to extract useful information for various applications ranging from security and defense (forensics), marketing, and personalization, to research in psychology and sociology [Farzindar and Inkpen 2015].

Twitter is a social networking service, where users can share messages known as *tweets*. These messages may consist of text, images, or videos, and are known for their limited length. Initially, textual tweets were limited to 140 characters; however, as of November 2017, this limit was extended to 280 characters [Rosen 2017]. On Twitter, users can *follow* other users—a glossary of Twitter terms can be found in Appendix A. Unlike many other social networking services, the act of following another user on Twitter does not require reciprocation. In addition, tweets are publicly visible by default, unless a user decides to change the settings of their account to restrict the visibility of their tweets.

All of this makes Twitter a unique environment, which provides researchers with an abundance of data to carry out text analytics using Natural Language Processing (NLP) and analytical methods. It is possible to learn some traits of social media users, such as gender and age, solely based on the textual content that they share on social media platforms.

1.2 Problem Statement

The objective of this thesis is to build a classification model using Natural Language Processing (NLP) and Machine Learning (ML) techniques that can identify the gender and age of the social media users based on the linguistic differences in their public, textual correspondence on Twitter (i.e., tweets).

We approach the gender identification task as a binary classification problem, and the age identification task as a multiclass classification problem, in accordance with the available datasets. A range of deep learning architectures and classical machine learning algorithms will be investigated and compared. The models will be trained and tested on two datasets to examine their generalizability.

1.3 Research Hypotheses

We hypothesize that:

1. Gender and age of Twitter users can be predicted given the textual tweets they have posted.

2. The available corpus of textual tweets contains sufficient information for the classification model to learn the relationship between the tweets and the gender and age of the user.

1.4 Practical Applications

Modeling gender and age is an important user modeling application. The results of this research can be used in the following areas.

1.4.1 Marketing and Advertising

Presenting advertisements to a specific demographic that is most receptive to it improves the effectiveness of an advertising campaign. In the recent years, targeted advertising using demographic profiling techniques has been proven to be more effective and less expensive than non-targeted "run of network" advertising [Beales 2009].

Therefore, an accurate and dependable understanding of the demographics of potential customers can translate into more effective targeted advertising campaigns.

1.4.2 Opinion Mining

Sentiment analysis (also known as *opinion mining*) helps organizations assess the opinion of different communities on a specific topic. It helps businesses get feedback from their customers about their products and services and adjust them to their needs. It is also a powerful tool for the government organizations, helping policy makers decide about new policies that can be beneficial to different demographics of the society.

1.4.3 Personalization and Recommendation Systems

Today, recommendation systems have a strong presence in our lives. Product recommenders on Amazon, content recommenders on social networking services such as Facebook and Twitter, and music and video recommenders on Spotify, YouTube, and Netflix are all examples of such systems. Gender and age are notable attributes in these systems. Therefore, the ability to identify the gender and age of a user when such information is not provided by the user or is not reliable, can improve the performance of personalization and recommendation systems.

1.4.4 Forensics, Security, and Defense

Forensic linguistics (also known as *legal linguistics*) is a branch of applied linguistics which deals with the forensic context of language, law, crime investigation, trial, and judicial procedure. An example would be finding the linguistic profile of the author of a suspicious or threatening note, and detecting specific characteristics of the author as linguistic evidence.

1.5 Outline

The upcoming chapters are organized as follows:

Chapter 2 presents a quick introduction to natural language processing and machine learning, and will review some of the most widely used supervised learning algorithms, deep learning architectures, and text vectorization techniques.

Chapter 3 reviews other research efforts related to this work, including a survey of the studies on gender and age identification from written documents, and the current state-of-the-art in social media author profiling.

Chapter 4 describes the two datasets used in this thesis to train and test the classification models: The dataset obtained from the PAN 2018 author profiling shared task, and the ASI dataset which was prepared as part of this work.

Chapter 5 describes our participation in the author profiling shared task at PAN 2018 as part of the Conference and Labs of the Evaluation Forum (CLEF) 2018 conference.

Chapter 6 describes our gender and age classification experiments on the ASI dataset, and the architectures of the classification systems developed.

Chapter 7 offers a conclusion and discusses potential future work.

Chapter 2

2 Background

2.1 Natural Language Processing

Natural Language Processing (NLP) is a branch of computer science, artificial intelligence, and computational linguistics directed toward the interaction of computers with human language.

NLP has various applications in today's world, including text analytics, question answering, machine translation, sentiment analysis, and text summarization, to name a few. NLP enables us to extract meaningful information from large amounts of textual data, without the need for direct human intervention.

2.2 Machine Learning

Machine learning (ML) is a subset of artificial intelligence (AI), which deals with algorithms and statistical models that enable computers to perform a specific task without the use of explicit instructions. In 1959, Arthur Samuel described ML as "the field of study that gives computers the ability to learn without being explicitly programmed." A more modern definition is offered by Tom Mitchel: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Machine learning tasks can be divided into three main paradigms:

- *Supervised learning*, where a labeled dataset is provided, and the classification model can be trained based on that dataset. Supervised learning problems are mostly categorized into *regression* and *classification* problems. In regression problems, the input data is mapped to a continuous function, whereas in classification problems we have discrete outputs.
- *Unsupervised learning*, where no training data is available, and the algorithm provides a better understanding of the data based on the relationships among the variables in the data. C*lustering* and *dimensionality reduction* are two well-known categories of unsupervised learning.

- *Reinforcement learning*, where an *agent* learns to choose actions in its *environment* to maximize some cumulative *reward*. AlphaGo, the computer program developed by Google DeepMind that plays the board game Go, is an example of reinforcement learning.

In what follows, we review some of the most widely used supervised learning algorithms. Although deep learning is a subfield of machine learning, we have dedicated a separate section to it, and will discuss it in section 2.3.

2.2.1 Linear Regression

Linear Regression estimates real values by mapping continuous input variables to output variables using an optimum linear function. This optimum linear function is called the *regression line* and is represented as $y = ax + b$.

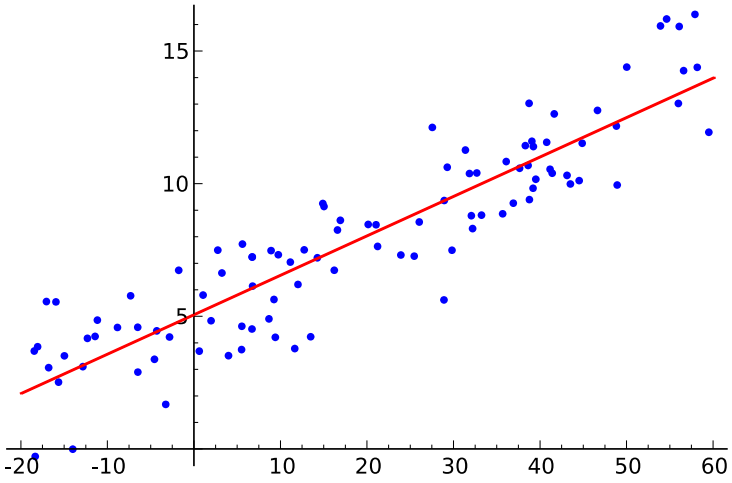

Figure 1. Random data points and their regression line. Public domain image obtained from Wikimedia Commons[2].

2.2.2 Logistic Regression

Despite how its name suggests, Logistic Regression (LR) is an algorithm used for classification rather than regression. Logistic regression can be *binomial*, dealing with two unordered output

[2] https://commons.wikimedia.org/wiki/File:Linear_regression.svg

classes, *multinomial*, dealing with three or more unordered classes, or *ordinal*, dealing with ordered, dependent variables.

2.2.3 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a commonly used supervised learning algorithm, which falls under the *kernel methods*. Originally introduced by Vladimir N. Vapnik and Alexey Y. Chervonenkis in 1963, the algorithm has since been subject to a series of improvements. Boser, Guyon, and Vapnik [1992] suggested a way to create non-linear classifiers using the *kernel trick*. The modern formulation of SVM was developed by Cortes and Vapnik [1995].

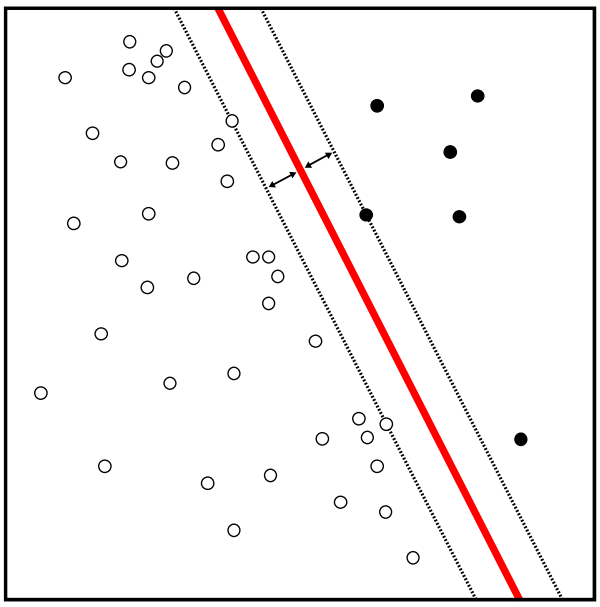

Figure 2. SVM maximum margin hyperplane. Image obtained from Wikimedia Commons[3], with some alteration. Used under the CC BY-SA 4.0 license[4].

SVM tries to solve binary classification problems by finding a good *decision boundary*. A decision boundary is a surface, separating data points into two groups in the space. Each side of the decision boundary would represent one of the two classes. To achieve this, SVM would first construct a

[3] https://commons.wikimedia.org/wiki/File:Kernel_Machine.svg

[4] https://creativecommons.org/licenses/by-sa/4.0/deed.en

model, mapping the training data to a new high-dimensional representation, where the decision boundary can be a *hyperplane* (a multi-dimensional plane).

Figure 2 demonstrates an SVM model. The points closest to the hyperplane are called the *support vectors*. The region bounded by the hyperplane and the support vectors is the *margin*. SVM would try to maximize the margin, by maximizing the distance between the hyperplane and the support vectors. A new data point can then be classified by being mapped into the same space and falling on either side of the hyperplane.

Calculating the coordinates of the points in the high-dimensional space is compute-intensive. SVM handles this problem using a kernel trick. A kernel function maps any two points in the initial space to the distance between those points in the target space, bypassing the explicit computation of the new representation space.

SVMs have long been very popular and have proven strong in simple classification problems. However, they do not perform well on perceptual problems, such as image processing, and are hard to scale to large datasets, due to high training time.

2.2.4 Naïve Bayes

The Naïve Bayes (NB) algorithm is a classifier based on the Bayes Theorem, which works with the assumption that all features are conditionally independent. In other words, the contribution of a feature is the same regardless of all other features. Take document D as represented by a vector of features $F = (f_1, f_2, \dots, f_n)$. The probability of D belonging to class C_j can be presented as:

$$P(C_j|F) = \frac{P(C_j)P(F|C_j)}{P(F)}$$

Due to the conditional independence of the features, the equation can be formulated as:

$$P(C_j|f_1, \dots, f_n) = P(C_j)\prod_{i=1}^{n} P(f_i|C_j)$$

The Naïve Bayes algorithm uses the above probability model and the maximum posteriori decision rule, to classify each test sample to the class with the highest probability.

2.2.5 Tree-based Classifiers

Random forests, decision trees, and gradient boosting machines are some of the most prominent tree-based classification algorithms. Decision trees are flowchart-like structures and are known for their interpretability. Decision trees build a large tree and prune it until a cost-complexity function is minimized. Random forests construct various decision trees and ensemble their outputs, which makes them a great tool for a wide range of classification problems. Similar to a random forest, a gradient boosting machine operates by constructing an ensemble of decision trees, and improving the model by training new models that focus on the areas where the previous model performed poorly.

2.3 Deep Learning

Deep learning is a subfield of machine learning based on Artificial Neural Networks (ANNs), with a new approach to learning representations from data by learning successive layers of increasingly meaningful representations [Chollet 2017]. ANNs are inspired by our understanding of the brain; however, there is no evidence that our brains operate using learning mechanisms similar to modern deep learning models.

In what follows we will offer an introduction to some of the main terms in deep learning. We will then briefly review some of the most widely used ANN architectures.

- **Forward propagation**

  Forward propagation is the process of calculating the activation values for all the neurons in the neural network based on the weights and inputs of the previous layer, starting with the first (input) layer and moving forward all the way to the last (output) layer.

- **Backpropagation**

  The backpropagation algorithm is used to calculate the partial derivatives of the cost function in order to minimize it. For each training sample, the following two steps are followed:

  1. Starting at the first layer, the activation value of all neurons in the neural network is calculated using forward propagation.
  2. Starting at the last layer and moving back to the first layer, the error value for each neuron is calculated.

- **Activation function**

  In order to achieve a richer hypothesis space and learn more complex representations, an activation function, also referred to as *transfer function*, is used in some neurons. The activation function determines the output of a neuron based on its inputs. Some of the most widely used activation functions are Rectified Linear Unit (ReLU), hyperbolic tangent (tanh), sigmoid, and softmax.

- **Batches**

  When training a neural network, forward propagation is performed on a *batch* (a number of training samples, determined by a hyper-parameter called *batch size*), and then the error for each neuron is calculated by performing backpropagation on the same batch. Based on this error, the weights of the model are then updated. When batch size is one sample, the learning algorithm is called *stochastic gradient descent*. When batch size is the same as the size of the training set, the learning algorithm is called *batch gradient descent*. Finally, when batch size is a value between one and the size of the training set, the learning algorithm is called *mini-batch gradient descent*.

- **Epoch**

  An epoch is a single training iteration of all batches (forward and backward propagation). At the end of each epoch, every training sample has been seen by the model. The number of epochs is a hyper-parameter that determines how many times the learning algorithm will go through the entire training set and update the weights of the model accordingly.

2.3.1 Feedforward Neural Networks

Feedforward neural network was the first and simplest type of neural network created. In a feedforward network, information always moves in the forward direction, and never goes backwards. The main goal of a feedforward network is to minimize a cost function $J(\Theta)$, by finding an optimal set of weights Θ.

The most basic form of a feedforward neural network is a single-layer perceptron, which consists of a single output layer, where inputs are directly fed into the output layer, with a series of weights. A multi-layer perceptron consists of multiple layers of neurons, where the first layer is the input layer, the last layer is the output layer, and the rest of the layers are called *hidden* layers. The

purpose of hidden layers is to increase the non-linearity of the model and achieve a better representation of the data. Figure 3 displays a multi-layer perceptron with only one hidden layer.

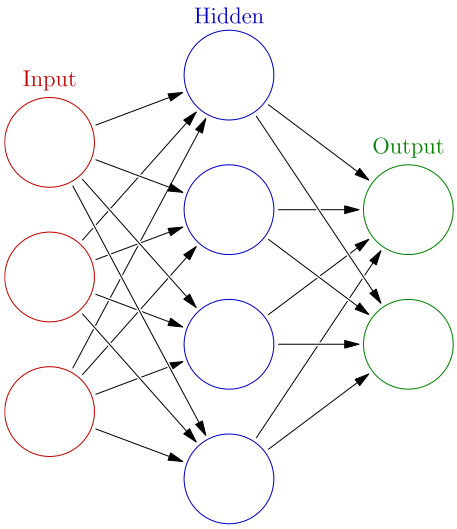

Figure 3. A feedforward neural network with one hidden layer. Image obtained from Wikimedia Commons[5]. Used under the CC BY-SA 3.0 license[6].

2.3.2 Recurrent Neural Networks (RNNs)

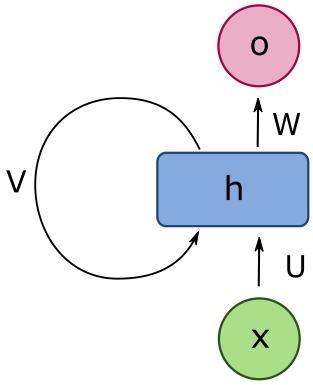

Figure 4. A one-unit recurrent neural network (RNN). Image obtained from Wikimedia Commons[7], with some alteration. Used under the CC BY-SA 4.0 license[8].

---

[5] https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg

[6] https://creativecommons.org/licenses/by-sa/3.0/deed.en

[7] https://commons.wikimedia.org/wiki/File:Recurrent_neural_network_unfold.svg

[8] https://creativecommons.org/licenses/by-sa/4.0/deed.en

A recurrent neural network (RNN) is a neural network with an internal loop, which maintains a state containing information about the previous inputs that the network has processed.

A one-unit RNN is demonstrated in Figure 4, with input connection x, output connection o, and hidden state h. The same network is viewed in Figure 5 as unfolded over time. U, V and W are the weights of the network and x_t, o_t and h_t are the input, output, and hidden state at timestep t.

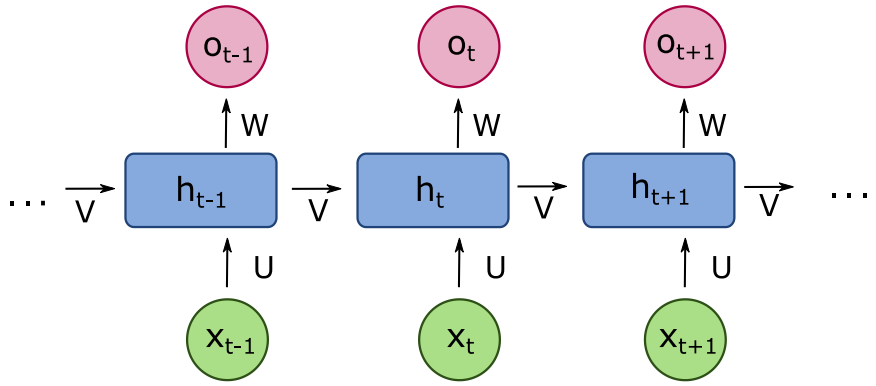

Figure 5. A one-unit recurrent neural network (RNN), unfolded over time. Image obtained from Wikimedia Commons[7], with some alteration. Used under the CC BY-SA 4.0 license[8].

The hidden state at each timestep is calculated based on the input of that timestep and the hidden state of the previous timestep. As a result, at each timestep t, the output tensor contains information from timesteps 0 to t in the input sequence. Therefore, having only one recurrent layer in the network, you will not need the full sequence of outputs, rather only the last output, since it already carries information about the entire sequence.

2.3.3 Long Short-Term Memory Networks (LSTMs)

Long Short-Term Memory Network (LSTM) is a specific type of RNN, which is capable of learning long-term dependencies. Although simple RNN is able to retain the information it has previously seen, in practice, it lacks the ability to maintain long-term dependencies—the information about the inputs seen in many timesteps before. This is referred to as the *vanishing gradient problem* [Bengio, Simard, and Frasconi 1994].

The LSTM algorithm was developed by Hochreiter and Schmidhuber [1997] to tackle the vanishing gradient problem by allowing past information to be carried forward across timesteps.

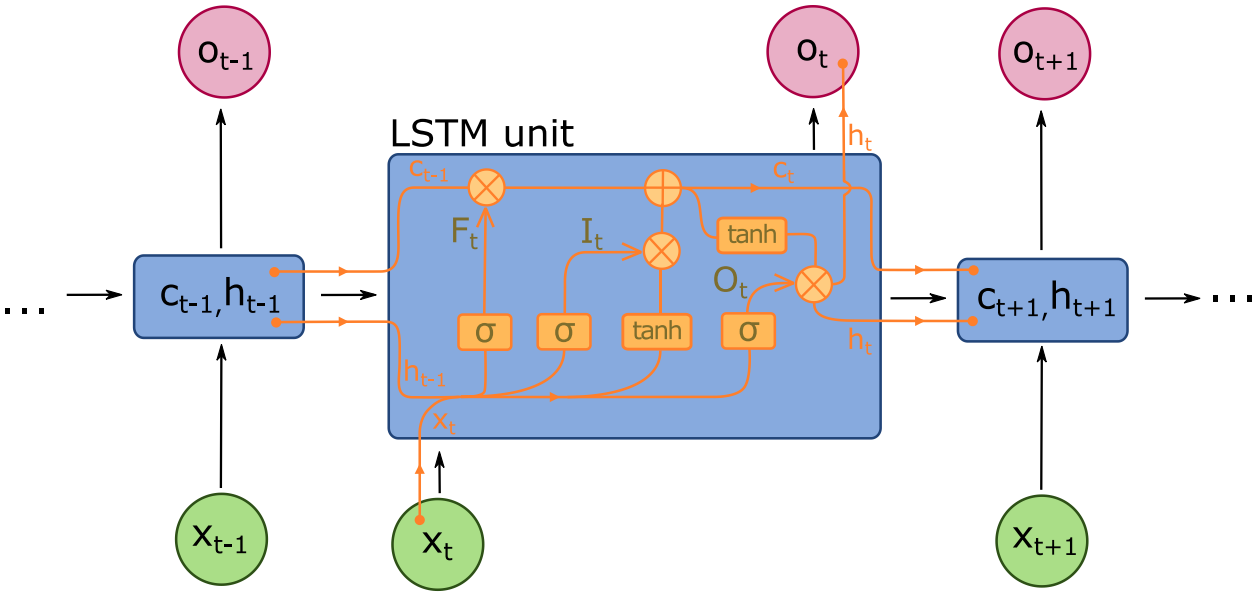

Figure 6. A one-unit LSTM network, unfolded over time. Image obtained from Wikimedia Commons[9]. Used under the CC BY-SA 4.0 license[10].

Figure 6 demonstrates a one-unit LSTM network, unfolded over time. Similar to what we previously saw for the RNN architecture, x_t, o_t and h_t are the input, output, and the hidden state at timestep t. c_{t-1} is the information carried from timestep $t-1$ to timestep t, which will be combined with the input connection x_t and hidden state h_{t-1} to form the hidden state h_t and the carry c_t which will be sent to the next timestep.

A bidirectional LSTM is a specific type of the LSTM network, consisting of two LSTM layers, which process the input sequence in different directions: chronologically and anti-chronologically. These two representations are then merged and form the output of the bidirectional LSTM. As a result, bidirectional LSTM would capture patterns in the data that a unidirectional LSTM may not.

2.4 Text Vectorization

To train a machine learning model on textual data, the data first needs to be represented using numeric values. That is, a vector representation of the document is created, which the classification model can then be trained on. Ideally, this vector would contain the main features of the textual

[9] https://commons.wikimedia.org/wiki/File:Long_Short-Term_Memory.svg

[10] https://creativecommons.org/licenses/by-sa/4.0/deed.en

corpus. In what follows we will present a brief introduction to some of the most widely used text vectorization techniques.

2.4.1 Bag of Words

A simple, yet effective text vectorization technique is *bag of words* (BoW). To create a numeric representation of a text, using this technique, the following steps are followed:

- Create a vocabulary of unique tokens (words).
- Count the number of occurrences of each token in each document.

As an example, consider the following two sentences as our two documents:

1. "The sun is shining in the sky."
2. "The earth revolves around the sun."

A set of every unique word token in the two documents is created as shown in Table 1.

Table 1. Bag-of-words vocabulary

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Token	the	sun	is	shining	in	sky	earth	revolves	around

Next, the numeric representation of the documents is created as a matrix, where each row represents a document, each column represent a token in the vocabulary, and each cell contains the number of occurrences of the corresponding token in the corresponding document:

$$\begin{bmatrix} 2 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

As you can see, this technique removes the notion of time from the documents. In other words, the order in which the words appear in a document is not reflected in the above representation. That is precisely why this technique is given the name *bag of words*.

2.4.2 N-grams

In Natural Language Processing, *n*-gram refers to a consecutive sequence of words or characters of length *n*. In its simplest form, word n-grams with n=1 are called *word unigrams*, which is the equivalent of the bag of words feature set.

Take the first sentence in the previous section as an example: “The sun is shining in the sky.” The word bigrams (n=2) feature set for this document would be as follows:

{“the sun”, “sun is”, “is shining”, “shining in”, “in the”, “the sky”}.

The advantage of word n-grams over the bag-of-words feature set is that it captures the order in which the words appear in the document. The same process can be done for characters of a document to construct character n-grams. Often, a combination of word and character n-grams is used as features.

2.4.3 Term Frequency–Inverse Document Frequency

In the field of Information Retrieval (IR), Term Frequency–Inverse Document Frequency (tf-idf) is a numerical statistic, which reflects the importance of a word in a document. Tf-idf can also be used as a feature weighting technique, to reduce the impact of the terms that occur in many documents.

For term t in document d, tf-idf is calculated as:

$$tf_{t,d} \times \log\left(\frac{N}{df_t}\right)$$

$tf_{t,d}$, is the *term frequency*, which is the number of times term t occurs in document d. N is the total number of documents in the corpus, and df_t is the *document frequency*, which is the number of documents where the term t appears.

2.4.4 One-hot Encoding

A basic, yet common text vectorization technique is one-hot encoding. For a vocabulary consisting of $|V|$ words, each word in the vocabulary is associated with a unique, binary vector of size $|V|$. In each of these binary vectors, only one element is 1, and the rest of the elements are 0.

One-hot encoding results in very high-dimensional, *sparse* vectors. Sparse vectors are vectors that consist of mostly zeros.

2.4.5 Word Embeddings

Word embeddings are a novel approach to associate a numeric vector with a word, using dense word vectors. While one-hot encoding results in high-dimensional, sparse, binary vectors, word embeddings associate words with low-dimensional vectors consisting of floating point numbers.

Word embeddings are meant to map natural language into a geometric space [Chollet 2017], where words with similar meanings would end up closer to each other in the word-embedding space. This embedding space is learned through an unsupervised machine learning algorithm. To learn this embedding space for a specific dataset, an initial, random vector is associated with each word. These vectors will then be learned through backpropagation, the same way that the weights of a neural network are learned.

An alternative to learning a new word-embedding space with every new task, is to use pre-trained word embeddings. Some examples of pre-trained word embeddings are:

- **Word2Vec**

  Word2Vec [Mikolov et al. 2013] is a predictive word-embedding model trained on a Google News corpus. Word2Vec generates a single word-embedding representation for each word in the vocabulary.

- **GloVe**

  Global Vectors (GloVe) [Pennington, Socher, and Manning 2014] is a word-embedding model similar to Word2Vec, developed at Stanford. GloVe is trained on aggregated word-word co-occurrence statistics from a corpus, consisting of the English Gigaword Fifth Edition[11] and the 2014 Wikipedia dump.

- **FastText**

  Developed at Facebook, FastText [Bojanowski et al. 2016] is an extension to Word2Vec, where instead of feeding the words into the neural network, words are broken into several

[11] https://catalog.ldc.upenn.edu/LDC2011T07

sub-words (character n-grams). The word-embedding vector for each word is then calculated as the sum of its sub-words.

- **BERT**

  Bidirectional Encoder Representations from Transformers (BERT) [Devlin et al. 2018] is a *bidirectional*, *contextual*, pre-trained language representation developed at Google. Unlike context-free models such as Word2Vec and GloVe, BERT takes the other words in a sentence into account when generating the representation of each word.

- **ELMo**

  ELMo [Peters et al. 2018] is another bidirectional, contextual, pre-trained word representation model. The representations in ELMo are purely character-based, enabling it to form a representation for out-of-vocabulary tokens.

2.5 Performance Measures and Evaluation Protocols

2.5.1 Performance Measures

For classification problems in machine learning, different metrics can be used to evaluate the performance of a classifier. Accuracy, precision, recall, and F-measure are among the most widely used performance measures. To illustrate each of these metrics, take a binary classification problem with *C1* and *C2* as the classes. The results of the classifier can be summarized into a table, known as the *confusion matrix*, as demonstrated in Table 2.

Table 2. Confusion matrix

	Predicted: C1	**Predicted: C2**
Actual: C1	True Positive (TP)	False Negative (FN)
Actual: C2	False Positive (FP)	True Negative (TN)

FP (short for false positive) represents the number of samples in the test set which belong to the C2 class, but are *falsely* classified as C1. In this example, the C1 class is regarded as the *positive* class. In binary classification, the class that is more uncommon or of more interest, is often considered to be the positive class, such as a spam email, or a disease in medical testing.

In what follows, we illustrate some of the most commonly used performance measures, using the established notation.

Accuracy: The proportion of the test set samples that are correctly classified by the model.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Error rate: The proportion of the test set samples that are incorrectly classified by the model.

$$error\ rate = 1 - accuracy = \frac{FP + FN}{TP + TN + FP + FN}$$

Precision: Precision answers the question, "What proportion of the selected samples are relevant?"

$$precision = \frac{TP}{TP + FP}$$

Recall: Recall answers the question, "What proportion of the relevant samples are selected?"

$$recall = \frac{TP}{TP + FN}$$

F-measure (F1 score): The harmonic mean of precision and recall. F-measure can provide a more realistic measure of the classification performance, by considering both precision and recall.

$$F_1 = \left(\frac{precision^{-1} + recall^{-1}}{2}\right)^{-1} = \frac{2 \times precision \times recall}{precision + recall}$$

2.5.2 Evaluation Protocols

The most commonly used evaluation protocols are *hold-out validation* and *k-fold cross-validation*.

Hold-out validation

In this approach, the dataset is randomly separated into two mutually exclusive subsets: a training set and a validation set. The model is fit on the training set and tested on the validation set. The hyper-parameters of the model can be tuned accordingly and the process can be repeated several times. One problem that can occur in this setting, is that every time you use feedback from the

validation set to tune the model, you *leak* information about the validation set into the model. This can ultimately cause the model to overfit on the validation set.

To prevent this from happening, the dataset can be separated into three subsets: a training set, a validation set, and a test set. Similar to the previous setting, the model would be trained on the training set and tested on the validation set. Finally, when the model is tuned to a suitable configuration, it will be tested on the test set, to assure it has not overfit on the validation set.

The hold-out method is the simplest evaluation protocol, but it has two drawbacks when used with small datasets:

1. The validation and test sets may be too small to be statistically representative of the whole dataset. As a result, the performance measures may be highly variable, depending on which samples are selected for the validation and test sets.
2. The training set would be smaller than the whole dataset, and as a result, the model trained on the training set would perform worse than a model trained on the whole dataset.

_k_-fold cross-validation

In this approach, the dataset is randomly separated into k mutually exclusive subsets of equal size, here called *folds*. The model is trained and tested k times. Each time, one of the k subsets is held out as a validation set and the model is trained on the remaining $k-1$ folds. The performance of the model can be reported using the average and other summary statistics of the performance measures of all k folds.

While this approach takes more computational power, it resolves the two drawbacks of the hold-out method, especially for small datasets. When k is chosen to be equal to the number of samples in the dataset, this approach is the equivalent of the *leave-one-out cross-validation* (LOOCV) method [James et al. 2014].

Chapter 3

3 Related Work

The abundance of user-generated data on social media has encouraged researchers all over the world to try to extract meaningful information from the social media data. In this chapter, we review some of the work that has been done with respect to identifying the gender and age of the social media users based on textual information.

3.1 Gender and Age Identification

Researchers have explored the identification of various traits of authors based on their writings. Earlier works focused on formal written documents, such as books, essays, and newspapers, with the size of the corpora varying from dozens to a few hundred documents. On the other hand, more recent studies have utilized social media data, including blogs, online forums, and social networks. Different content-based and style-based features have been employed, including word n-grams and part-of-speech (POS).

Pennebaker, Mehl, and Niederhoffer [2003] studied the link between language and demographic, psychological, and social traits of a person, including age and gender. Along with his colleagues, Pennebaker [2013] developed the Linguistic Inquiry and Word Count (LIWC) tool, which exploits word counts and word categories to identify traits of authors. LIWC has since become one of the most used tools in author profiling.

Koppel, Argamon, and Shimoni [2002] classified gender on a genre-controlled corpus of formal written documents (e.g., books and newspaper articles) with an accuracy of about 79%. They started with a total of 1,081 features to represent the documents, including part-of-speech (POS) n-grams, and a list of 405 function words. Afterwards, they performed feature selection and trained their classifier on the most important features.

Koppel et al. [2002] showed that for a test set containing fiction, training on the combination of fiction and non-fiction genres diminishes the classification accuracy, compared to training on a dataset only containing fiction. Moreover, training on the fiction corpus and testing on non-fiction resulted in an accuracy of 50%, which was the same as a majority class baseline. This points out another complication in gender identification: the linguistic markers in an author's writings can

differ across genres. As a result, classifying an author's gender by seeking patterns in their style of writing may represent a completely different task in each genre.

Argamon et al. [2003] probed the variation of writing styles between male and female on a corpus of formal texts and achieved approximately 80% accuracy. Similarly to Koppel et al., they identified syntactical and lexical differences pertaining to gender. They reported negations (e.g., 'not') and pronouns (I, you, she, her, their, myself, yourself, herself) as strong indicators for the female class, and determiners (a, the, that, these) and quantifiers (one, two, more, some) as indicators for the male class. They interpreted this as female writing style being more 'involved' and male writing style being more 'informational'. The terms 'involved' and 'informational' were used in accordance with [Biber 1995]: 'Involved' documents contain features which typically indicate interaction between the speaker/writer and the audience, such as first and second person pronouns. As for 'informational' features, prepositions are among the most notable.

With blogs becoming more popular in the early years of 2000s, a massive amount of public data became available and shifted the focus of some researchers to social media. Schler et al. [2006] compiled a corpus of about 1.4 million blog entries from 37,478 blogs on *blogger.com*, which contained about 295 million words. In comparison, Argamon et al. [2003] used a corpus of 600 documents containing 25 million words and Mulac, Studley, and Blau [1990] exploited as few as 96 essays. Schler et al. [2006] explored the writing style differences in users of different age groups and genders and reported significant differences in both style and content. They considered three age groups as 10s (13–17), 20s (23–27), and 30s (33–43). They omitted the intermediate ages to allow a clear discrimination and to account for the time difference between blog posts of a blogger, which was a few years in some cases.

While the rise of blogging and social media brought an abundance of data, it also introduced some new challenges. Blog posts are typically shorter in nature, compared to formal literature. In addition, the language used in blog posts is often more spontaneous and less formal, containing emoticons and "blog words", such as 'lol', 'haha', and 'u' (in place of 'you').

Schler et al. [2006] used style-based features—consisting of the above-mentioned "blog words", part-of-speech (POS) and hyperlinks—along with the 1000 unigrams with the highest information gain. They obtained an accuracy of 80 percent on gender classification, using the Multi-Class Real Winnow (MCRW) learning algorithm, which they claimed to be more efficient than SVM, with

comparable results. They found that women used “blog words” more frequently, while there was a higher occurrence of hyperlinks in posts by men. Additionally, females made more use of pronouns and negations, while males used more articles (e.g., a, and, the) and prepositions. This supports the findings of Argamon et al. [2003] on formal, non-blog corpora. Schler et al. [2006] also reported that male bloggers write more about politics, technology, and money, while female bloggers share more about their personal lives.

Garimella and Mihalcea [2016] also used blogs to detect semantic and psycholinguistic word classes that distinguish each gender. They reported that some of the predominant word classes for women were about sweetness, touch, sourness, and texture, suggesting that women have a greater sense of perception of the world. Family-related words were also seen more in female authors. The main word classes for men were those related to religion, sports, science, and work. The study also investigated whether polysemous words—words with multiple meanings—were used differently by men and women, by performing a gender-based word sense disambiguation. However, the results proved that the chosen word senses were similar for the most part.

Burger et al. [2011] undertook gender classification of Twitter users and achieved 76% accuracy, when trained their model only on tweets, using word unigrams and bigrams and character 1- to 5-grams as features. They adopted the Balanced Winnow2 learning algorithm, which they claimed to have a better combination of accuracy, speed, and robustness than Naïve Bayes and linear SVM. Furthermore, they assessed the performance of 130 human annotators on Amazon Mechanical Turk, annotating the gender of Twitter authors solely based on their textual tweets, and reported that only 5% of the human annotators were able to out-perform the classification model.

3.2 State-of-the-art in Social Media Author Profiling

PAN is an annual series of shared tasks on digital text forensics, which organizes tasks on plagiarism detection, credibility analysis, author profiling, author identification, and author obfuscation among others. The workshop invites Computer Science researchers and practitioners to work on the said problems of interest to improve the state-of-the-art.

Author profiling has been undertaken as a shared task at PAN annually since 2013, with the focus being mostly on gender and age identification on social media data. This has substantially contributed to the body of work in this area.

In this section, we review the state-of-the-art in the Author Profiling task at PAN, with a focus on gender and age identification. Most of the research in this area can be organized into four steps:

1. Data collection
2. Pre-processing
3. Feature extraction
4. Classification

3.2.1 Data Collection

Data collection in this context consists of retrieving and annotating textual data to build a corpus for training and testing a machine learning model as a profiling system. We will describe the data collection process later in Chapter 4, where we discuss the datasets used in this work.

3.2.2 Pre-processing

Pre-processing is the process of preparing and cleaning the textual dataset by removing or replacing certain words and characters that contribute little to no valuable information to the classification model that predicts the labels (e.g., gender or age) of the test examples.

Pre-processing can be described as measures taken to prepare the text corpus for optimal extraction of features. This is achieved by filtering out parts of the text that can be regarded as noise. Pre-processing is a crucial step in every machine learning task, as it directly affects the learning ability of the model.

Burger et al. [2011] opted for a simple pre-processing approach. They tokenized the words using non-alphanumeric characters as separators, which is reasonable, since they only exploited n-gram word and characters as features.

The datasets provided by PAN are in the XML format. XML is a ubiquitous markup language providing a well-formed document, which makes it a suitable choice for interchanging data. As done by Ucelay et al. [2016], Bilan and Zhekova [2016], and Arroju, Hassan, and Farnadi [2015], the general first step of PAN participants is to remove the XML tags from the dataset, as the tags are not part of the tweets and are considered noise.

The most common preprocessing step among all participants of the shared task is removing or normalizing Twitter-specific elements, i.e., username mentions (*@username*), hashtags (*#hashtag*), and URLs [Rangel et al. 2018]. Here, by *normalizing* we mean replacing all user mentions, for example, with a specific token such as '<user_mention>'. This would considerably decrease the total number of tokens in the dataset [Schaetti 2018; Däniken, Grubenmann, and Cieliebak 2018].

Some participants removed character flooding [Ciccone et al. 2018; Raiyani et al. 2018], which is another way to normalize tokens in datasets collected from social media. Character flooding is the repetition of characters, which on social media is used as a demonstration of excitement or strong feelings about something. For example, "Soooooooo happy to be back!!!!!!!"

Lowercasing the words was another common practice among participants, as done by Däniken, Grubenmann, and Cieliebak [2018] and Veenhoven et al. [2018], among others. In addition, Weren [2015] applied stemming while Bougiatiotis and Krithara [2016] performed lemmatization, with no improvement reported in their results.

Most participants removed punctuations [Ciccone et al. 2018; HaCohen-Kerner et al. 2018; Veenhoven et al. 2018; Martinc et al. 2017; Modaresi, Liebeck, and Conrad 2016] and numbers [Bougiatiotis and Krithara 2016; Markov et al. 2016], with Schaetti [2017] removing both as well as out-of-alphabet words. Nowson et al. [2015] removed all character sequences representing emojis.

Stop-words were removed in [HaCohen-Kerner et al. 2018; Veenhoven et al. 2018; Kheng, Laporte, and Granitzer 2017; Martinc et al. 2017; Agrawal and Gonçalves 2016]. Some participants removed short tweets based on word or character count. Kheng, Laporte, and Granitzer [2017] removed tweets which contain less than 10 characters, including special characters. Contractions and abbreviations were expanded in [Stout, Musters, and Pool 2018; Raiyani et al. 2018].

3.2.3 Feature Extraction

To train a machine learning model on textual data, the data first needs to be represented using numeric values. In other words, a vector representation of the document is created, which the

learning algorithm can work with. Ideally, this vector would contain the main features of the textual corpus. These features can be grouped into content-based features and style-based features.

Word and character n-grams are widely used in PAN shared tasks [Tellez et al. 2018; Kosse, Schuur, and Cnossen 2018]. On the other hand, style features have been used by some participants. Counts of stop-words, punctuation marks, emoticons, and slang words were used in [Patra, Das, and Das 2018]. HaCohen-Kerner et al. [2018] used the average number of characters and the average number of words per tweet. Däniken, Grubenmann, and Cieliebak [2018] and Martinc et al. [2017] used counts of emojis/emoticons and Orts [2018] used the skewness calculated from a variation of the Low Dimensionality Statistical Embedding (LDSE) [Rangel, Franco-Salvador, and Rosso 2017]. A combination of word and character n-grams with bag-of-terms was used by Aragón and López-Monroy [2018] and character flooding was used by Martinc et al. [2017].

The best performing team at PAN 2017 used word unigrams and bigrams, along with character 3- to 5-grams as features [Basile et al. 2017]; however, some teams experimented with word and character *n*-grams with values of *n* up to 3 and 7 respectively [Rangel et al. 2017].

At PAN 2015 [Rangel et al. 2015], the best performing team adopted Latent Semantic Analysis (LSA), and reported a 4% increase in accuracy in their English dataset, compared to bag-of-words (BOW) [Álvarez-Carmona et al. 2015].

In deep learning implementations, Martinc, Skrlj, and Pollak [2018], Veenhoven et al. [2018], López-Santillán, González-Gurrola, and Alonso [2018], and Takahashi et al. [2018] among others, used word embeddings, whereas Schaetti [2018] and Franco-Salvador et al. [2017] used character embeddings. Miura et al. [2017] combined word and character embeddings, and Poulston, Waseem, and Stevenson [2017] also used traditional features such as tf-idf n-grams. Combination of word embeddings and stylistic features were used by Patra, Das, and Das [2018]. As for feature weighting techniques, Term frequency–inverse document frequency (tf-idf) was commonly used among participants.

Similar to word embeddings, Dhingra et al. [2016] created embeddings of tweets based on the raw textual content. They built a character-level bidirectional Gated Recurrent Units (GRU) neural network, which takes raw text as input to classify hashtags of tweets. The characters were processed as one-hot character vectors, similar to one-hot word vectors. The motivation behind

this architecture was to capture the common out of vocabulary words in tweets, which word-level models simply ignore, because these words do not end up in the embedding dictionary, as they do not occur frequently enough. The challenge of such systems is that they need to learn the syntactic and semantic structures of text from the ground up, whereas word-level architectures rely on prior knowledge of words.

In neural networks models, feature extraction and classification become parts of the same model, as the data representation is implicitly done in the neural network.

3.2.4 Classification

A variety of approaches have been explored towards the gender and age identification task. Some of the classification algorithms that have been used include SVMs, logistic regression, random forest, deep learning (CNNs and RNNs), and Naïve Bayes, as well as ensemble methods which combine the above [Rangel et al. 2017].

Support Vector Machine (SVM) can be considered a state-of-the-art model for author profiling. Basile et al. [2017] achieved the highest performance in the author profiling task at PAN 2017, using a linear SVM classifier. Many other participants have also opted to use SVMs [Aragón and López-Monroy 2018; Ciccone et al. 2018; Patra, Das, and Das 2018; Tellez et al. 2018, 2017; Veenhoven et al. 2018; López-Monroy et al. 2017].

Logistic regression was another learning algorithm that was used by many participants, including Martinc et al. [2017], Dias and Paraboni [2018], HaCohen-Kerner [2018], and Nieuwenhuis and Wilkens [2018]. Other classical machine learning algorithms that were tried by the participants include multi-layer perceptron [HaCohen-Kerner et al. 2018], a basic feed-forward network [Kosse, Schuur, and Cnossen 2018], and distance-based methods [Tellez et al. 2018; Khan 2017; Kocher and Savoy 2017].

With regard to deep learning methods, Takahashi et al. [2018], Bayot and Gonçalves [2018], and Kodiyan et al. [2017] used Recurrent Neural Networks (RNN), while Convolutional Neural Networks (CNN) were used in [Aragón and López-Monroy 2018; Schaetti 2018; Sezerer et al. 2018; Martinc, Skrlj, and Pollak 2018; Schaetti 2017; Sierra et al. 2017]. Miura et al. [2017] who ranked third in gender identification at the PAN 2017 competition, probed both RNN and CNN with attention mechanism, max-pooling layer, and a fully-connected layer. Veenhoven et al.

[2018] used a Bidirectional LSTM model with attention mechanism, and reported that this model performs better compared to a CNN model.

As for ensembling methods, Stout, Musters, and Pool [2018] applied an ensemble of Naïve Bayes and RNN, while Poulston, Waseem, and Stevenson [2017], and Ciobanu et al. [2017] used an ensemble of different configurations of their algorithms. Poulston, Waseem, and Stevenson [2017] combined logistic regression with a Gaussian process trained on word embeddings. Franco-Salvador et al. [2017] used Deep Averaging Networks and Raiyani et al. [2018] used a dense neural network, but did not obtain exceptional results.

While traditional machine learning algorithms such as SVM and logistic regression have consistently achieved the highest performance on the task of gender and age classification, deep learning methods have gained increasing attention in the recent years. With a capacity to solve complex problems on large datasets, deep learning is an essential area to investigate.

Chapter 4

4 Data

In this work, two datasets were used. The first dataset was prepared by PAN for the author profiling task of 2018. We obtained the second dataset from Advanced Symbolics Inc. (ASI) and prepared it according to our needs for this research. In this chapter, we describe the details of the collection and preparation of both datasets.

4.1 Author Profiling Task at PAN 2018

The CLEF (Conference and Labs of the Evaluation Forum) Initiative has provided a corpus for the author profiling task at PAN 2018, which is a subset of the corpus for the same task at PAN 2017 [Rangel et al. 2018]. In this section, we describe the steps taken by the organizers of the task to create the corpus, as explained in [Rangel et al. 2017] and [Rangel et al. 2018].

4.1.1 Selecting Target Languages and Dialects

The first step in collecting the dataset was to select the target languages and varieties (dialects) in each language. The following languages and dialects were selected:

- Arabic: Egypt, Gulf, Levantine, Maghrebi
- English: Australia, Canada, Great Britain, Ireland, New Zealand, United States.
- Spanish: Argentina, Chile, Colombia, Mexico, Peru, Spain, Venezuela.

The PAN 2017 author profiling task also included tweets in the Portuguese language (consisting of Brazilian and Portuguese dialects), however, this language did not makes its way to the PAN 2018 workshop.

4.1.2 Retrieving Tweets Based on Region

For each dialect, major cities within the region where the dialect is used were selected. These cities consisted of the capital city and in some cases the cities with the most population. For each city, tweets posted in the radius of 10 kilometers from the center of the city were retrieved. The list of selected cities for each language variety is shown in Table 3.

Table 3. Selected cities for each language variety

Language	Variety	City
Arabic	Egypt	Cairo
	Gulf	Abu Dhabi, Doha, Kuwait, Manama, Mascate, Riyadh, Sana'a
	Levantine	Amman, Beirut, Damascus, Jerusalem
	Maghrebi	Algiers, Rabat, Tripoli, Tunis
English	Australia	Canberra, Sydney
	Canada	Toronto, Vancouver
	Great Britain	London, Edinburgh, Cardiff
	Ireland	Dublin
	New Zealand	Wellington
	United States	Washington
Spanish	Argentina	Buenos Aires
	Chile	Santiago
	Colombia	Bogota
	Mexico	Mexico
	Peru	Lima
	Spain	Madrid
	Venezuela	Caracas

4.1.3 Identifying Unique Users

In the retrieved dataset, as described in the previous section, for each Twitter user, their timeline was also retrieved, which consists of some meta-data, including the user's full name, location (as stated by the user in the form of free text), and language (as stated by the user). However, the language described by the user on their Twitter timeline may not match the actual language their tweets are written in.

4.1.4 Discarding Users with Few Tweets

In this step, any author who had less than 100 tweets in their corresponding language (Arabic, English, or Spanish) was removed from the dataset. These tweets do not include retweets.

4.1.5 Annotating Gender

Gender annotation was carried out in two steps:

1. Automatically, by means of a dictionary of proper names, excluding names with ambiguous genders. In the final dataset, the names of the authors were discarded and authors were coded with a random alpha-numeric ID.
2. Manually, having annotators visit each Twitter profile and check if the automatically generated gender label corresponds to the user's photo, profile description, and so on.

4.1.6 Creating the Final Dataset

The final dataset for the PAN 2018 author profiling task is balanced (stratified) on gender labels and contains 100 tweets and 10 images for each user. For each user, the 100 latest tweets were selected for the corpus. As a result, the time frame of the tweets may vary from days to months, depending on how frequently a user tweets.

Regarding the images, according to PAN, they retrieved all the images posted on each user's timeline, taking into account at least the last 1,000 tweets. They then randomly selected 10 of the images. Therefore, the time frame of the images does not correspond to that of the textual tweets. Moreover, users with less than 10 images posted on their timeline were discarded.

Table 4. Number of users in the corpus, per language

Language	Training set	Test set	Total
Arabic	1,500	1,000	2,500
English	3,000	1,900	4,900
Spanish	3,000	2,200	5,200

4.2 Advanced Symbolics Inc. (ASI)

Advanced Symbolics Inc. (ASI) is a company based in Ottawa, Canada, which provides scientific research regarding public opinion on a specific topic. ASI collects and maintains a dataset of tweets posted by a sample of Twitter users that is statistically representative of the population of Canada.

In this section, we describe ASI's methodology in collecting and annotating this dataset, as well as the steps that we took to select a subset of this dataset and prepare it for the experiments in this work.

4.2.1 Sampling Users Based on Location

ASI collects a random sample of Twitter users using Conditional Independence Coupling (CIC)[12]. CIC [Li 2016] is a sampling algorithm which crawls online social networks and builds a sample of users which is statistically representative of the population. The resulting sample is mathematically proven to converge to the *stationary distribution*[13] of the population.

There are two ways a Twitter user may explicitly share their location:

- Geolocation (GPS information)
- Self-declared location on the user's profile

Sharing the GPS location information is an opt-in feature on Twitter, meaning it is disabled by default and each user needs to enable it to use it. After that the precise location (latitude and longitude) of the user is detected using the GPS chip on their device and sent along with their tweet.[14]

A user can also provide their location on their Twitter profile, as a textual description. This is also optional and the location provided on a profile does not necessarily reflect the current location of the user.

If a user has opted to enable the GPS location information, ASI uses this information to determine the user's location. Since a user will probably post tweets from different locations, all geolocation points are clustered using a k-means algorithm. The cluster with the most number of points is selected as the user's home location.

Often, the GPS location information is not enabled by the user. In these cases, the self-declared location on the user's profile is used. This information is often the name of a city or state/province, and in some cases a street address. ASI performs an address search using Microsoft's Bing Maps. Bing Maps will then return a geographical place corresponding to the provided textual address or

[12] https://www.advancedsymbolics.com/wp-content/uploads/2018/03/ASI-METHODOLGY-SUMMARY.pdf

[13] A special distribution of a Markov chain

[14] https://help.twitter.com/en/safety-and-security/tweet-location-settings

detect if the self-declared address is invalid—for instance a user might declare their location as 'Earth' or 'Somewhere on this planet'.

CIC can be restricted to sample only from a specific geographic region, such as a country, province, or city. This is done by restricting the network that CIC is crawling to the users that are located in the desired geographical area. In cases where neither the GPS location information nor a valid self-declared location is available, CIC cannot assign a geographical place to the individual. The sample created by CIC is statistically representative of the Canadian population, without the need to weight the sample based on demographics.

The sample of Twitter users that ASI maintains consists of nearly 250,000 users from across Canada. Table 5 and Figure 7 demonstrate the distribution of the users based on municipality. For brevity, we have only shown the 17 most populated municipalities here. For the complete list, refer to Table 27 in Appendix B.

Table 5. Distribution of users based on municipality

Municipality	**Users**	**% Users**
Toronto	35,506	17.6%
Vancouver	26,466	13.1%
Montréal	18,981	9.4%
Ottawa & Gatineau (Ontario part)	14,863	7.4%
Calgary	11,816	5.9%
Edmonton	10,788	5.3%
Halifax	6,693	3.3%
Hamilton	5,398	2.7%
Winnipeg	4,933	2.4%
Kitchener, Cambridge & Waterloo	4,841	2.4%
Victoria	4,036	2.0%
London	3,509	1.7%
Québec	2,659	1.3%
Saskatoon	2,478	1.2%
Regina	2,235	1.1%
St. Catharines & Niagara	2,191	1.1%
New Glasgow	2,114	1.0%
Other	42,268	20.9%
Grand Total	**201,775**	**100.0%**

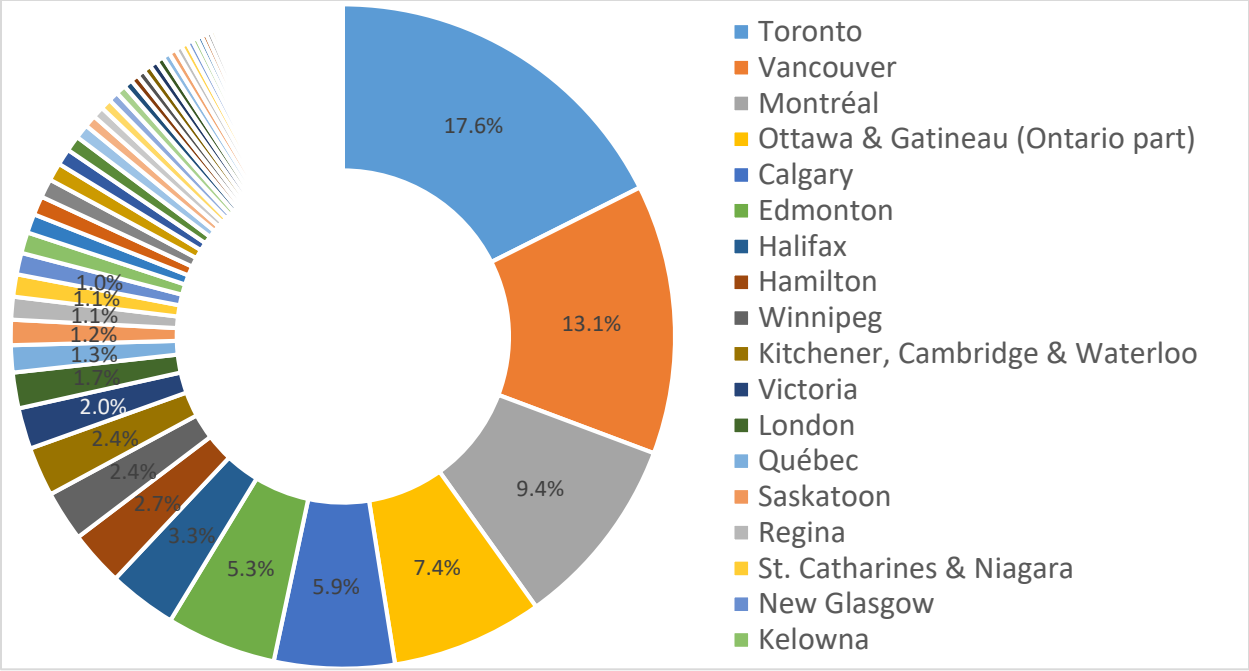

Figure 7. Distribution of users based on municipality

As mentioned in section 4.2.1, in cases where neither GPS information nor a valid self-declared location is available for a user, the individual is left without a municipality label, hence the difference between the grand total in Table 5 (201,775) and the total number of users in ASI's sample (248,932).

4.2.2 Calculating Age and Gender Probability Distributions

The age and gender of each user is automatically determined using a combination of AI systems, based on the user's first name and profile picture.

Birth records in US and Canada record how many male and female babies were given a specific first name in each year for the past 150 years. Moreover, actuarial tables (also known as *life table* or *mortality table*) show, for each age, the probability that a person of that age will still be alive after their next birthday. The first name of the user is examined against the combination of birth records and the actuarial tables and a probability distribution for the individual's age and gender is obtained. Actuarial tables are used to adjust the probability distribution based on the likelihood that an individual of a certain age would still be alive.

The user's profile picture is analyzed using the Face++[15] image analysis software. Face++ is a deep learning-based image recognition and analysis service, capable of detecting face attributes such as age, gender, ethnicity, and emotion. The URL of the profile photo of each Twitter user in the dataset is given to the Face++ API and it calculates a probability distribution of age and gender based on the features of the image.

ASI then combines these two sets of probabilities to deduce the probability distribution for age and gender of each user. Gender is grouped into *female* and *male*, and age is categorized into the following six ranges:

- [0, 25] Less than 25 years old
- [25, 34] Between and including 25 and 34 years old
- [35, 44] Between and including 35 and 44 years old
- [45, 54] Between and including 45 and 54 years old
- [55, 64] Between and including 55 and 64 years old
- [65, …] 65 years old or more

4.2.3 Privacy Practices

According to ASI, all of their data is retrieved from publicly available websites, and in compliance with each website's terms and conditions. Privacy is preserved through *data anonymization*. Data anonymization is the process of removing *personally identifiable information* from datasets. According to ASI, once the demographics probability distributions are computed, all the personally identifiable information of users is discarded, including the user's name, profile picture, self-declared location, and geolocation information.

4.2.4 Age and Gender Annotation: Selecting a Subset of the Users

As described in section 0, each user in ASI's sample is assigned a set of probabilities for age and gender. Figure 8 and Table 6 demonstrate the probability distribution of gender for the ASI sample. For each user, the sum of the two probabilities of female and male is equal to 1.

[15] https://www.faceplusplus.com

Table 6. Probability distribution of gender

Female Probability	Male Probability	Users	% Users
0–0.02	0.98–1	71,568	28.75%
0.02–0.04	0.96–0.98	6,070	2.44%
0.04–0.06	0.94–0.96	1,335	0.54%
0.06–0.08	0.92–0.94	1,355	0.54%
0.08–0.1	0.9–0.92	1,127	0.45%
0.1–0.12	0.88–0.9	803	0.32%
0.12–0.14	0.86–0.88	354	0.14%
0.14–0.16	0.84–0.86	1,753	0.70%
0.16–0.18	0.82–0.84	303	0.12%
0.18–0.2	0.8–0.82	405	0.16%
0.2–0.22	0.78–0.8	196	0.08%
0.22–0.24	0.76–0.78	457	0.18%
0.24–0.26	0.74–0.76	281	0.11%
0.26–0.28	0.72–0.74	801	0.32%
0.28–0.3	0.7–0.72	408	0.16%
0.3–0.32	0.68–0.7	258	0.10%
0.32–0.34	0.66–0.68	181	0.07%
0.34–0.36	0.64–0.66	175	0.07%
0.36–0.38	0.62–0.64	39	0.02%
0.38–0.4	0.6–0.62	164	0.07%
0.4–0.42	0.58–0.6	224	0.09%
0.42–0.44	0.56–0.58	180	0.07%
0.44–0.46	0.54–0.56	172	0.07%
0.46–0.48	0.52–0.54	235	0.09%
0.48–0.5	0.5–0.52	269	0.11%
0.5–0.52	0.48–0.5	89,470	35.94%
0.52–0.54	0.46–0.48	248	0.10%
0.54–0.56	0.44–0.46	111	0.04%
0.56–0.58	0.42–0.44	216	0.09%
0.58–0.6	0.4–0.42	197	0.08%
0.6–0.62	0.38–0.4	358	0.14%
0.62–0.64	0.36–0.38	180	0.07%
0.64–0.66	0.34–0.36	320	0.13%
0.66–0.68	0.32–0.34	121	0.05%
0.68–0.7	0.3–0.32	96	0.04%

0.7–0.72	0.28–0.3	240	0.10%
0.72–0.74	0.26–0.28	233	0.09%
0.74–0.76	0.24–0.26	485	0.19%
0.76–0.78	0.22–0.24	461	0.19%
0.78–0.8	0.2–0.22	371	0.15%
0.8–0.82	0.18–0.2	562	0.23%
0.82–0.84	0.16–0.18	588	0.24%
0.84–0.86	0.14–0.16	1,130	0.45%
0.86–0.88	0.12–0.14	1,097	0.44%
0.88–0.9	0.1–0.12	551	0.22%
0.9–0.92	0.08–0.1	672	0.27%
0.92–0.94	0.06–0.08	908	0.36%
0.94–0.96	0.04–0.06	659	0.26%
0.96–0.98	0.02–0.04	2,151	0.86%
0.98–1	0–0.02	58,394	23.46%
Grand Total		**248,932**	**100.00%**

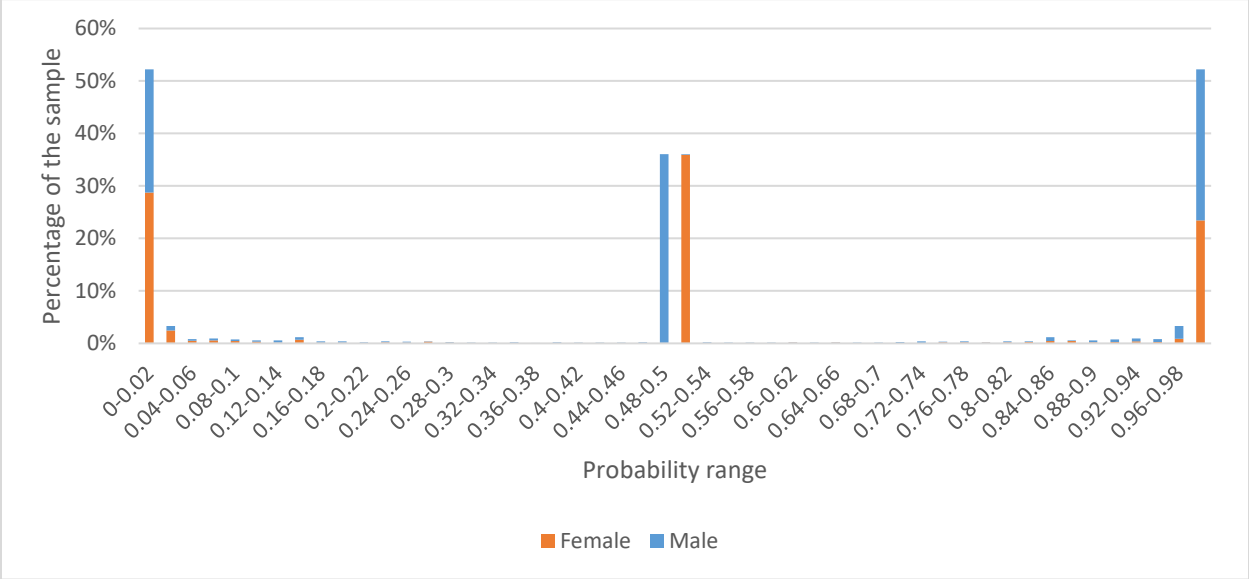

Figure 8. Stacked probability distribution of gender groups

For about 52% of the users in the sample, the probability of one gender is between 0.98 and 1. These are the individuals with first names that are strictly female or male names, and profile photos that the Face++ API can identify as that same gender as their first name, with a high confidence. For about 36% of the users in the sample, the probability of either gender groups lie in the range of 0.50 ± 0.02. These are the users with first names that cannot be strongly attributed to one gender,

and profile photos that cannot be identified as female or male with a high confidence. The remaining 18% of the users lie somewhere in between.

To create a dataset with high-quality gender labels, we opted to keep only the users with a gender probability in the range of [0.98, 1], and avoid the rest of the sample.

On the other hand, the probability distribution of age groups was not as clear-cut as for gender.

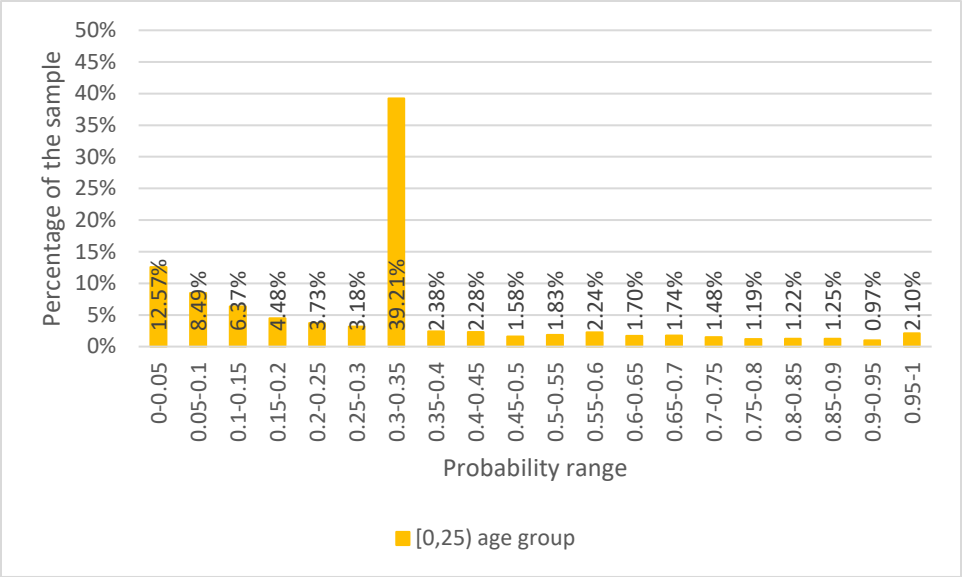

Figure 9. Probability distribution of age group [0, 25]

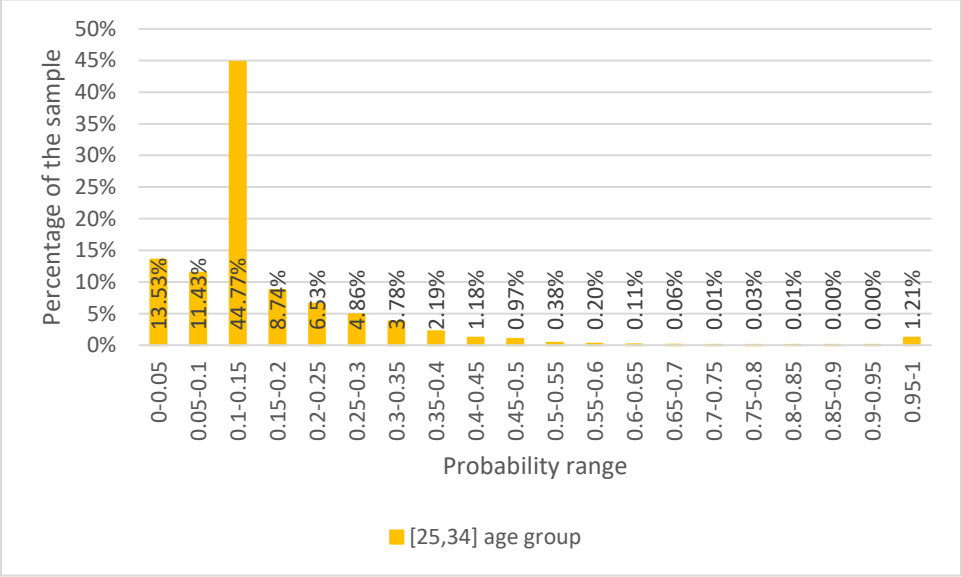

Figure 10. Probability distribution of age group [25, 34]

Figure 9 through Figure 14 demonstrate the probability distribution for each age group. The corresponding tables can be found in Appendix B (Table 28 through Table 33). Similar to gender, for each user, the sum of the probability of all six age groups is equal to 1.

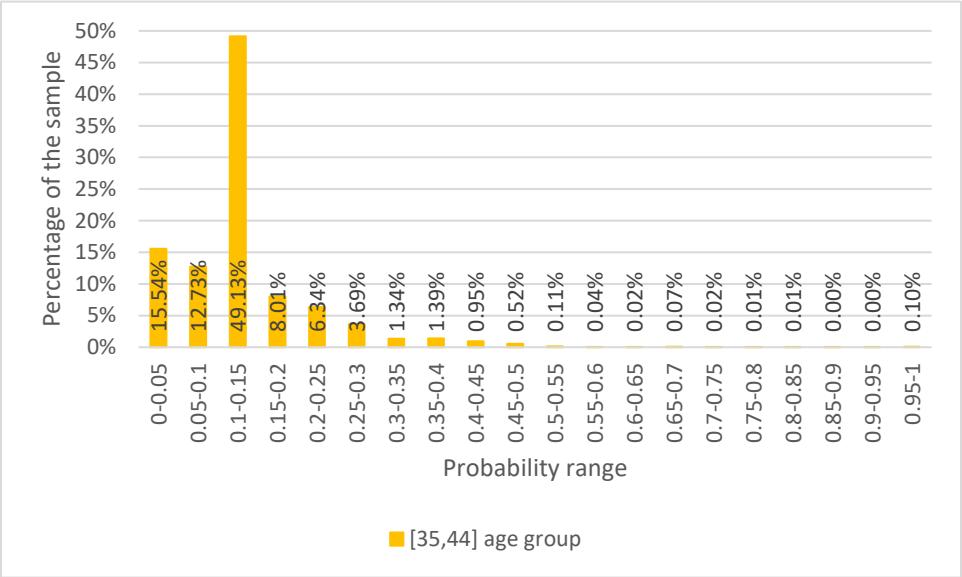

Figure 11. Probability distribution of age group [35, 44]

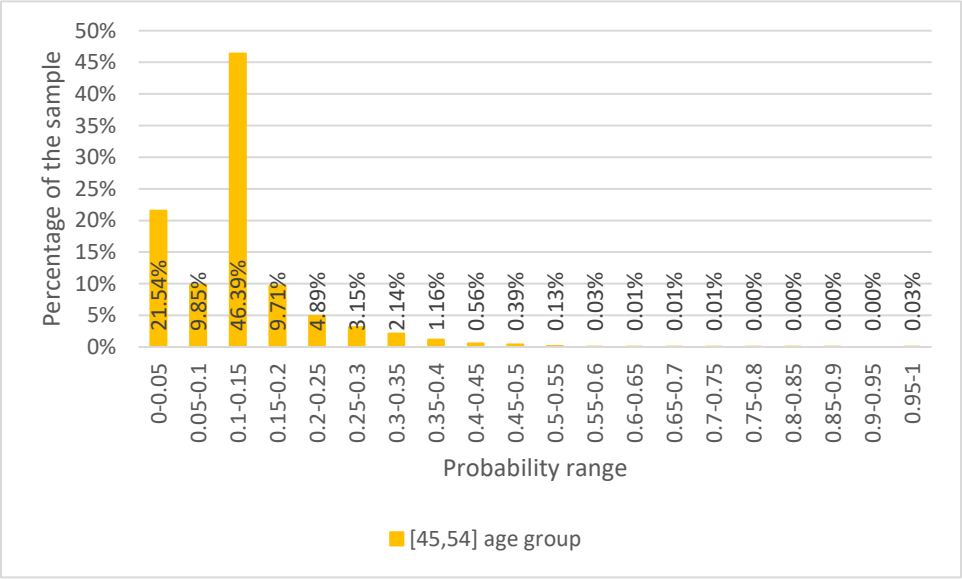

Figure 12. Probability distribution of age group [45, 54]

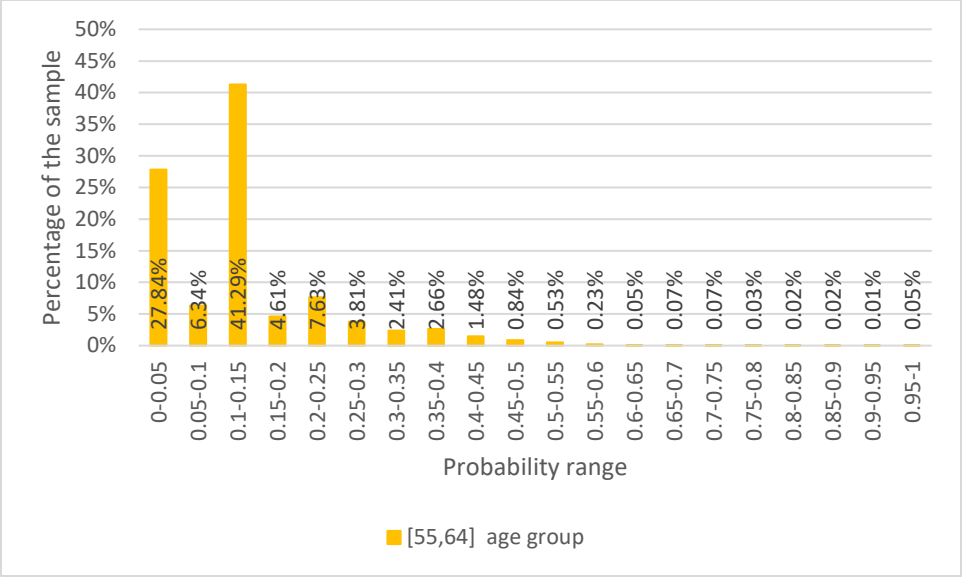

Figure 13. Probability distribution of age group [55, 64]

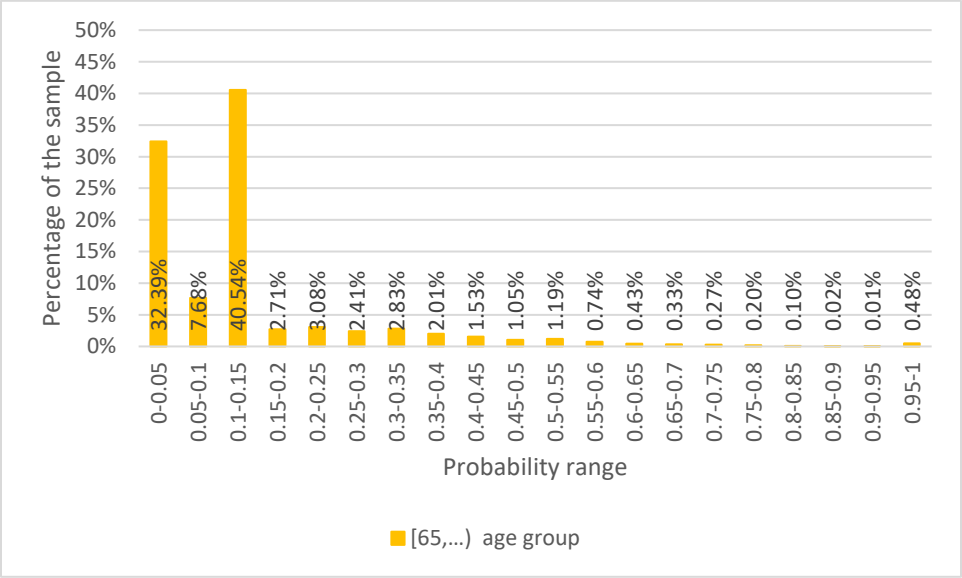

Figure 14. Probability distribution of age group [65, …]

For age groups, we initially chose to keep users which have a probability of more than 0.50 for one of the age groups, and discard the rest of the sample. In this case, for each remaining user one age group would have a probability of $p > 0.5$ and the rest of the five age groups would have a total probability of $1 - p < 0.5$. However, by doing this, the resulting sample would not have represented the population of Canada.

Statistics Canada conducts a national census in Canada every five years[16]. The 2016 Census Program is the latest of its kind and provides statistical data of the demographics of Canada. Figure 15 and Table 7 demonstrate the population of Canada by age and gender, according to the 2016 Census Profile [Statistics Canada 2017].

Table 7. Population of Canada by age and gender, 2016 Census

Age Group	Female Population	Male Population	Total Population	% Population	% Population
0 to 4 years	925,760	973,030	1,898,790	5.40%	28.76%
5 to 9 years	983,445	1,034,685	2,018,130	5.74%	
10 to 14 years	937,445	985,200	1,922,645	5.47%	
15 to 19 years	986,940	1,039,215	2,026,155	5.76%	
20 to 24 years	1,098,200	1,144,495	2,242,695	6.38%	
25 to 29 years	1,141,515	1,144,470	2,285,985	6.50%	13.13%
30 to 34 years	1,181,110	1,148,290	2,329,400	6.63%	
35 to 39 years	1,169,730	1,118,635	2,288,365	6.51%	12.93%
40 to 44 years	1,150,695	1,104,440	2,255,135	6.42%	
45 to 49 years	1,202,210	1,157,760	2,359,970	6.71%	14.33%
50 to 54 years	1,359,320	1,318,755	2,678,075	7.62%	
55 to 59 years	1,335,055	1,285,190	2,620,245	7.45%	13.97%
60 to 64 years	1,175,630	1,114,885	2,290,515	6.52%	
65 to 69 years	1,019,405	953,075	1,972,480	5.61%	16.89%
70 to 74 years	742,900	677,975	1,420,875	4.04%	
75 to 79 years	552,300	469,550	1,021,850	2.91%	
80 to 84 years	423,880	325,765	749,645	2.13%	
85 years and over	501,990	268,790	770,780	2.19%	
Grand Total	**17,887,530**	**17,264,205**	**35,151,735**	**100.00%**	**100.00%**

In order to create a dataset that represents the population of Canada, we randomly selected about 30,000 users from the remaining users in the ASI sample, such that the percentage of users in each age group corresponds with that of the 2016 Census—as seen in the last column of Table 7.

[16] https://www12.statcan.gc.ca/census-recensement/index-eng.cfm

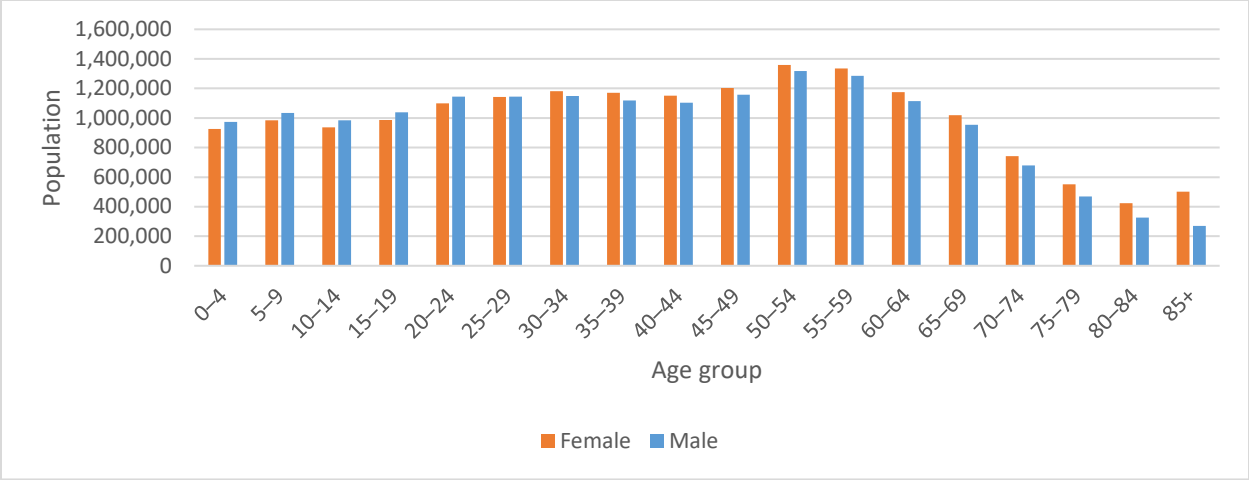

Figure 15. Population of Canada by age and gender, 2016 Census

As mentioned earlier, all of the selected users have an assigned gender probability of 0.98 or more. As for age, we needed to make some compromises in three of the age groups, namely [35, 44], [45, 54], and [55, 64]. As there were not enough users with an assigned age probability of more than 0.5 for these three age groups, we used a probability of more than 0.40, which is still significantly higher than the $\frac{1}{6} \approx 0.16$ random baseline. On the other hand, for the [0, 25] age group we selected users with a probability of more than 0.70.

In the end, we assigned a gender label and an age label to each of the 30,000 selected users based on the class with the highest probability. The results of our experiments, discussed in Chapter 6, show that the quality of the gender labels of the ASI dataset are comparable to that of the PAN 2018 dataset.

4.2.5 Cleaning the Corpus

After selecting the users, we downloaded all of their public tweets posted in the span of 2012 to 2018. We then performed the following cleaning and filtering process on the corpus:

1. Remove all retweets from the list of tweets of each user. The Twitter API specifies retweets with an 'RT' string at the beginning of the tweet text.

2. Remove all URLs from tweets.

3. Remove all *@username* mentions from tweets.

4. Lowercase all tweets.

5. Remove character flooding.

   Character flooding is the repetition of characters, sometimes used to demonstrate excitement. We used NLTK's TweetTokenizer to replace repeated character sequences longer than 3 with sequences of length 3. For instance, 'Sooooo happy' would be replaced with 'Sooo happy'.

6. Detect and remove all non-English tweets.

   For language detection, we used the *polyglot*[17] library which depends on the *pycld2*[18] library, which itself depends on the *Compact Language Detector 2 (CLD2)*[19] library.

7. Remove all users with less than 100 words in all their tweets combined.

   We first counted the word tokens in each tweet (not including the URLs and username mentions, which were removed in the previous steps, and not including punctuation), and then summed the count of word tokens over all tweets of each user. Users who had less than 100 words in all their tweets combined, were discarded from the dataset.

4.2.6 Creating the Final Corpus

We saved the corpus in the XML format. XML is a ubiquitous markup language providing a well-formed document, which makes it a suitable choice for interchanging data. During all gender classification experiments, the largest possible subset of the corpus that was stratified on gender classes was randomly selected, using fixed random seeds, resulting in 22,220 users. The same was done for age classes during age classification experiments, which resulted in 11,160 users in the corpus for age classification experiments.

[17] https://polyglot.readthedocs.io

[18] https://pypi.org/project/pycld2

[19] https://github.com/CLD2Owners/cld2

Chapter 5

5 Gender Classification at PAN 2018

5.1 Introduction

In this chapter, we describe our participation in the author profiling shared task at PAN 2018 as part of the Conference and Labs of the Evaluation Forum (CLEF) 2018 conference. Our approach takes advantage of different types of word and character n-grams as features, dimensionality reduction using Latent Semantic Analysis (LSA), and a linear Support Vector Machine (SVM) classifier.

Our model achieved the highest performance in textual classification among the 23 participating teams at the PAN 2018 author profiling shared task, with the accuracy of 0.8221, 0.8200, and 0.8090 on the English, Spanish, and Arabic test set, respectively.

Our approach was published in [Daneshvar and Inkpen 2018] as part of the CLEF 2018 Working Notes. Moreover, we have made the source code to our approach publicly available at: https://github.com/SamanDaneshvar/pan18ap

5.2 Methodology

Our approach consists of text preprocessing, feature extraction, and classification model construction. In this section, we describe each of these steps and the tools used, as well as our performance measure and evaluation protocol.

5.2.1 Text Preprocessing

As described in section 4.1, the dataset of the PAN 2018 author profiling task consisted of 100 tweets and 10 images for each user—a total of 1,260,000 tweets for the 12,600 users in all three languages. In this work, we only exploited the text of the tweets (not the images). All retweets were already removed from the dataset by the organizers of PAN.

For the preprocessing step, we used the *TweetTokenizer* module from the Natural Language Toolkit (NLTK) library [Bird, Loper, and Klein 2009] as well as some regex procedures. In our submitted system, we followed the preprocessing steps as described below:

1. **Removing the XML tags**

   The datasets provided by PAN were in XML format. XML is a ubiquitous markup language providing a well-formed document, which makes it a suitable choice for interchanging data. Our first step was to parse the XML files into strings containing the tweets of users.

2. **Merging the tweets of each user**

   All 100 tweets of each user were merged into one string, with an '<EndOfTweet>' tag added to the end of each tweet.

3. **Lowercasing**

   It is assumed that normalizing the case of characters will help capture the content-related features. As discussed in section 3.2.2, lowercasing is among the most popular preprocessing steps.

4. **Removing character flooding**

   Character flooding is the repetition of characters, sometimes used to demonstrate excitement. We used NLTK's *TweetTokenizer* to replace repeated character sequences longer than 3 with sequences of length 3. For instance, 'Sooooo happy' would be replaced with 'Sooo happy'.

5. **Normalizing URLs**

   All URLs were replaced with the '<URLURL>' tag.

6. **Normalizing username mentions**

   All *@username* mentions (Twitter handles) were replaced with the '<UsernameMention>' tag. By replacing a Twitter-specific element (e.g., username mention) with a single token, we capture the information regarding the presence of the element in the text, while discarding the specific username. This will help reduce noise and lower the number of features.

7. **Punctuation removal**

   The *TfidfVectorizer* function of the scikit-learn [Pedregosa et al. 2011] library ignores punctuation at all times and treats punctuations as token separators.[20]

[20] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

8. **Stop words removal**

   Stop words are the most common words of a language, and do not offer much useful information in terms of class discrimination. Removing stop words helps the classification algorithm focus on more important features.

   Stop words were detected and eliminated in the feature extraction step, based on document frequency. Any n-gram that occurred in all documents was considered a stop word and was removed from the feature set.

In the development phase, we also explored with stemming and removing Twitter *#hashtags*; however, they were not used in our final system, as they were found to hurt the results rather than improve them. The results of these experiments can be found in section 5.3.3.

5.2.2 Feature Engineering

We found a combination of word and character *n*-grams to be most effective at capturing the important features for this task. We used the *TfidfVectorizer* function from the scikit-learn library [Pedregosa et al. 2011] to extract the features.

To find the best values for *n*, we performed several experiments, some manually and some using scikit-learn's grid search function. As the three languages (Arabic, English, and Spanish) had three separate datasets and needed separate classification models, we treated them as three independent tasks, and found the optimum set of word and character n-grams for each language separately.

We ended up using two different systems; one for the English dataset and another for both the Arabic and Spanish datasets. The following features were used in the final model that was submitted to the shared task.

For the English language:

- Word unigrams, bigrams, and trigrams
- Character 3- to 5- grams
- Dimensionality reduction using latent semantic analysis (LSA) was performed on the n-grams

For the Arabic and Spanish languages:

- Word unigrams and bigrams
- Character 3- to 5-grams

The following hyper-parameters were used for the vectorizer for all three datasets:

- ***min_df*: 2**

  Ignore terms (tokens) with a document frequency strictly lower than 2. Setting this value to 2 helps eliminate typos and misspellings.

- ***use_idf*: True (default)**

  Use tf-idf weighting

- ***sublinear_tf*: True**

  Apply sublinear tf scaling. Enabling this would replace tf with $1 + \log(tf)$.

- ***max_df*: 1.0 (default)**

  Tokens that occur in all documents will be regarded as stop words and left out of the dictionary of vocabulary.

Topic modeling using latent semantic analysis (LSA) was applied only on the English dataset. We performed linear dimensionality reduction by means of truncated singular-value decomposition (SVD), using scikit-learn's *TruncatedSVD* function. LSA is an unsupervised learning technique which was first introduced in [Dumais et al. 1988] as a method to improve information retrieval (IR) by reducing the dimension of the IR problem. LSA starts by performing singular-value decomposition (SVD) on the tf-idf matrix, and as a result, representing the documents and terms in a reduced-rank space. This reduced space is called the *semantic* space, since it captures the relationships among words and documents [Dumais 2005].

We experimented with different ranks for the LSA reduced space, and concluded that setting the number of dimensions to 300 seems to capture the most relevant information of the corpus, as found by [Wiemer-Hastings 2004].

5.2.3 Learning Algorithm

We experimented with Naïve Bayes, logistic regression, and SVM classifiers, and found SVM with a linear kernel to achieve the highest performance as measured by classification accuracy.

Although hyper-parameter optimization was done on the classifier, we found the accuracy improvement to be minimal. Therefore, in our submitted model, we used the default hyper-parameters of scikit-learn's implementation of linear SVM.

5.2.4 Performance Measure and Evaluation Protocol

We have used accuracy as our measure of success, since the task is framed as a binary classification problem, in accordance with the available dataset. Due to the relatively small size of the dataset, we used 10-fold cross validation as our evaluation protocol. For the final system, in addition to cross validation, we trained the model on the whole training set and tested it on the official test set of the shared task.

5.3 Experiments and Results

In this section, we discuss all the experiments that were conducted in order to build the best classification model for the gender identification task on the PAN 2018 author profiling dataset. We will then present the final system and discuss how it holds up against other systems in related work.

In order to prevent over-fitting on the training set, and to avoid having to set aside a considerable portion of the training set for validation, we evaluated the accuracy of our models using stratified 10-fold cross-validation, in all of our experiments. There is a bias-variance trade off in choosing the value of k in k-fold cross-validation. $k = 5$ and $k = 10$ have been shown empirically to give decent results [James et al. 2014].

During the shared task, the official test set was not made available to the participants. Therefore, for our experiments, we randomly selected 40% of the shared task's training set and kept it aside for test. With the exception of our final system (explained below), during all experiments, our model never saw this part of the dataset and was evaluated by 10-fold cross validation on the 60% portion of the dataset: 1,800 authors for each of the English and Spanish datasets and 900 authors for the Arabic dataset. This randomly selected subset was balanced (stratified) on gender labels.

To evaluate our final system, we performed 10-fold cross-validation on the shared task's training set (3,000 authors for the English dataset, and so on). Our submitted model was trained on the

whole training set and tested on the task's test set [Stamatatos et al. 2018], on the TIRA platform [Potthast et al. 2014]. The results are shown in Table 14.

The reported confidence interval (CI) of the experiments is the CI for the accuracy values of the 10-fold cross-validations. We calculated the 95% confidence interval as 2σ, where σ is the population standard deviation, computed using the following expression:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Where $n = 10$, x_i is the accuracy score of the i-th fold, and $\bar{x}$ is the mean of the 10 accuracy scores.

Loosely speaking, this means that 95% of the cross-validation scores lie within the range of $\bar{x} \pm CI$. As seen in the rest of this section, this range is quite considerable. This is an indication of the variance in the dataset, and suggests that testing the model on a single text set cannot be considered the ultimate judgement on its performance.

5.3.1 Experimental Plan

In order to examine the effect of each decision in the preprocessing, feature construction, and classification steps, we need to evaluate each step separately, while holding the other steps constant. Therefore, for each of the three steps—preprocessing, feature construction, and classification—we followed a procedure where a simple baseline setting was chosen for the other two steps and the effect of different options in the step under examination was evaluated. The best performing option was selected and carried forward.

Starting with the classification algorithm, different classification algorithms were evaluated on a system with simple lowercasing and tokenization preprocessing and bag-of-words as features. SVM was found to be the best performing classifier.

Next, the effect of different preprocessing techniques was evaluated on an SVM system with bag-of-word features and a combination of them was selected and carried forward.

The bulk of the experiments were carried out on feature extraction, selection, and transformation. Various reweighting and feature exclusion measures were examined on bag-of-words features using grid search, including idf reweighting, sublinear tf scaling, and stop-words threshold. The optimum combination was carried forward. Next, various combinations of word- and character-level n-grams were investigated, as well as topic modeling using LSA with different ranks. Offensive expressions were also tried as features, with the help of a dictionary of offensive language.

5.3.2 Comparison of Learning Algorithms

To compare the performance of different classifiers, we used a baseline setting for preprocessing and feature extraction. We preprocessed the tweets by lowercasing and tokenization using NLTK's *TweetTokenizer*, and used bag-of-words as features, weighted by absolute frequency. We used scikit-learn's *TfidfVectorizer* as our vectorizer, with default parameters.

Table 8 demonstrates the results obtained from Naïve Bayes, logistic regression and SVM algorithms.

Table 8. Baseline performance of different classifiers

Classifier	**Mean Accuracy ▼**	**Confidence Interval**
SVM	79.11%	2.38%
Logistic Regression	77.82%	2.64%
Naïve Bayes	76.17%	3.22%
Majority Class Baseline	50%	N/A

As it can be seen in the table, although Naïve Bayes has been effective for various NLP applications in the past, it performed poorly at this task. Logistic Regression performed slightly better, and the best results were obtained using SVM. For this experiment, we used SVM with a linear kernel, with the default parameters of *scikit-learn*'s implementation.

5.3.3 Effect of Preprocessing Techniques

Using SVM, which performed better than the other classification algorithms, and the same implementation of bag-of-words as features, we examine the effect of various preprocessing techniques. Table 9 shows the results of these experiments. The baseline is a simple preprocessing using lowercasing and tokenization, same as the experiments in the previous section.

As seen in Table 9, normalizing the URLs had an improvement on the accuracy, while stemming had a slight negative impact on the system. Moreover, removing hashtags hurt the results rather than improve them, which indicates that hashtags may contain information that can be useful in discriminating gender.

Moving forward, we combined the preprocessing measures that seemed to work best: lowercasing, removing character flooding, normalizing URLs and username mentions, punctuation removal, and stop words removal. In section 5.2.1, we have described these steps in detail.

Table 9. Effect of preprocessing methods on a bag-of-words SVM system

Preprocessing	Mean Accuracy ▼	Confidence Interval
Normalizing URLs	79.43%	2.61%
Removing character flooding	79.32%	3.57%
Removing stop words	79.24%	2.44%
Removing punctuation	79.13%	2.18%
Baseline	79.11%	2.38%
Stemming	78.76%	3.24%
Removing hashtags	77.43%	3.04%

5.3.4 Effect of Weighting and Exclusion Hyper-parameters

In the previous sections, we found the classification algorithm and preprocessing steps that were most effective for this task. In this section, we use those settings along with bag-of-words as

features as a baseline to find the best hyper-parameters for the vectorizer. We use scikit-learn's *TfidfVectorizer* for feature extraction.

We examined the following variations of hyper-parameters of the vectorizer using cross-validated grid-search to find the optimized setting. In what follows, by *document*, we mean all tweets of each user concatenated into a single text string.

- ***min_df*: 1 (default), 2, 5, 10**

  Any token with a document frequency lower than this threshold (also referred to as the cut-off value) will be ignored when building the vocabulary. In other words, any term (token) that has occurred in fewer documents than this threshold will not be included in the dictionary of vocabulary.

- ***use_idf*: True (default), False**

  Enable inverse document frequency (idf) reweighting. In other words, use term frequency–inverse document frequency (tf-idf) instead of term frequency (tf).

- ***sublinear_tf*: False (default), True**

  Apply sublinear tf scaling. Enabling this would replace tf with $1 + \log(tf)$.

- ***max_df*: 1.0 (default), 0.95, 0.9, 0.8**

  Any token with a document frequency proportion higher than this threshold will be ignored when building the vocabulary. This will identify the corpus-specific stop words (most common tokens in the corpus).

The best performance was achieved with the following values:

- ***min_df*: 2**

  Ignore terms (tokens) with a document frequency strictly lower than 2. Setting this value to 2 will help eliminate typos and misspellings.

- ***use_idf*: True (default)**

  Use tf-idf weighting

- ***sublinear_tf*: True**

  Apply sublinear tf scaling. This parameter had a significant impact on the results, improving the accuracy by about 2%.

- ***max_df*: 1.0 (default)**

  Tokens that occur in all documents will be regarded as stop words and left out of the dictionary of vocabulary.

5.3.5 Word and Character N-grams and LSA

As discussed in section 5.2.2, to find the optimum values for *n*, we performed several experiments, some manually and some using scikit-learn's grid search function. We treated the three languages (Arabic, English, and Spanish) as three independent tasks, and ended up with two different systems; one for the English dataset and another for both the Arabic and Spanish datasets.

For the Arabic and Spanish languages:

- Word unigrams and bigrams
- Character 3- to 5-grams

Table 10 demonstrates the cross-validation scores on the Arabic and Spanish datasets for word and character n-grams feature sets and their combination.

Table 10. Cross-validation scores for word and character n-grams on the Arabic and Spanish datasets

	Arabic		**Spanish**	
Features	**Mean Ac.**	**CI**	**Mean Ac.**	**CI**
Word 1- and 2-grams	78.22%	7.52%	77.06%	7.01%
Character 3- to 5-grams	77.89%	7.40%	77.11%	5.30%
Word 1- and 2-grams + Character 3- to 5-grams	80.33%	6.82%	78.17%	6.83%

For the English language:

- Word unigrams, bigrams, and trigrams
- Character 3- to 5- grams
- Dimensionality reduction using latent semantic analysis (LSA) was performed on the n-grams

Table 11 shows the cross-validation scores on the English dataset for word and character n-grams and their combination, as well as the effect of LSA with a dimension of 300.

Although performing dimensionality reduction using LSA resulted in a 0.94% increase in the mean accuracy, we found this improvement to be statistically insignificant. The p-value was calculated as $p = 0.29$, which was bigger than the intended 5% threshold and failed to reject the null hypothesis of equal averages, meaning that the samples are drawn from the same distribution.

Table 11. Cross-validation scores on the English dataset for word and character n-grams with and without dimensionality reduction

Features	Mean Accuracy	CI
Word 1- to 3-grams	80.67%	4.15%
Character 3- to 5-grams	81.33%	4.80%
Word 1- to 3-grams + Character 3- to 5-grams	81.89%	5.40%
(Word 1- to 3-grams + Character 3- to 5-grams), LSA 300	82.83%	5.50%

We carried out hypothesis testing on the cross-validation results of the n-grams without LSA and n-grams with LSA (for the English dataset), using the *normaltest*[21] and *ttest_ind*[22] functions from the SciPy library [Jones, Oliphant, and Peterson, n.d.].

The *normaltest* function is an implementation of D'Agostino and Pearson's test [D'Agostino 1971; D'Agostino and Pearson 1973] which combines skew and kurtosis to produce an omnibus test of normality. This test showed that our samples come from a Gaussian (normal) distribution.

The *ttest_ind* function calculates Student's t-test and Welch's t-test for the means of two independent samples of scores.

[21] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.normaltest.html

[22] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html

Performing LSA on the Arabic and Spanish datasets did not show any significant improvements.

5.3.6 Offensive Language as Features

In this section, we discuss our experiments with using the frequency of offensive expressions as features, on the English dataset.

It is a rather intuitive hypothesis that men use profanities more frequently than women. We investigated this hypothesis using the Flame dictionary of offensive language expressions [Razavi et al. 2010]. The dictionary[23] consists of 2,648 expressions, marked with a flame level from 1 (least offensive) to 5 (most offensive).

We started by training our SVM classifier on the expressions of each flame level separately, so that we can compare the information gain of the expressions in each flame level. We constructed the features with the following two steps:

1. Count the number of times each expression occurred in each document (all 100 tweets of each user combined).

2. Transform the count matrix into a normalized tf or tf-idf representation, using the *TfidfVectorizer* function from the scikit-learn library.

In the first set of experiments, the un-normalized count matrix (tf) was used. In the second set of experiments, l^2-normalization was performed on the count matrix. With l^2-normalization, each row of the matrix (representing each document, i.e., the tweets of each author combined) is normalized to have a sum of squares equal to 1.

Table 12 demonstrates the results of these experiments, measured by 10-fold cross-validation. CI refers to the 95% confidence interval for the accuracy values of the cross-validation, as described in the opening of section 5.3.

With the majority class baseline being 50%, we can see that the flame level 3, 4, and 5 expressions contain almost no information capable of discriminating gender. The highest performance was

[23] http://www.site.uottawa.ca/~diana/resources/

achieved by the combination of the expressions in all five flame levels, with l^2-normalization, with an accuracy of 65.94% ± 7.18%.

We repeated the same experiments, this time using the inverse document frequency weighting (tf-idf matrix instead of tf), with and without normalization. However, the results were lower than the above.

Table 12. Cross-validation scores for offensive expressions feature set, with and without normalization

	No Normalization		**l^2-Normalization**	
Flame Level	Mean Accuracy	CI	Mean Accuracy	CI
1	57.67%	5.32%	59.67%	5.99%
2	61.72%	6.39%	63.00%	5.88%
3	52.83%	5.17%	53.89%	5.79%
4	54.28%	8.56%	54.56%	9.79%
5	52.56%	4.51%	51.44%	4.90%
1 & 2	61.17%	7.36%	65.33%	5.43%
All (1–5)	61.28%	6.90%	65.94%	7.18%

Table 13. Cross-validation scores for the combination of all offensive expressions and word and character n-grams

Features	**Mean Accuracy**	**CI**
N-grams	81.89%	5.40%
N-grams, LSA 300	82.83%	5.50%
N-grams, LSA 300 + Offensive expressions	82.72%	6.80%
N-grams, LSA 300 + Offensive expressions, LSA 300	82.61%	6.87%
N-grams, LSA 300 + Offensive expressions, LSA 10	82.67%	4.79%
N-grams, LSA 300 + Offensive expressions, LSA 5	82.78%	5.04%

Lastly, we combined the offensive expressions feature set with word and character n-grams, after performing dimensionality reduction on them using LSA. We also experimented with performing LSA on the offensive expressions, with different number of dimensions. The performance of the model was not improved with any of these experiments.

Table 13 shows the results.

5.3.7 Final System

Our final system, described in section 5.2, was submitted to the shared task through the TIRA platform (Potthast et al., 2014). TIRA is an isolated virtual machine designed to ensure the validity of the results achieved by participants of PAN shared tasks. The participants run their code on this environment without having access to the official test set of the shared task.

We carried out a final 10-fold cross-validation on the whole training set of the task (3,000 authors for the English dataset, and so on). Our submitted model was trained on the whole training set and tested on the official test set of the task (1,900 authors for the English dataset and so on) on the TIRA environment. Results are shown in Table 14.

Table 14. Accuracy scores of 10-fold cross-validation and the task's test set

	Cross-validation		**Test Set**
Dataset	**Mean Accuracy**	**CI**	**Accuracy**
Arabic	81.53%	6.40%	80.90%
English	82.73%	5.22%	82.21%
Spanish	79.87%	4.17%	82.00%

5.3.8 Generalizability Testing

To assess the generalizability of our final English model, we tested it on a portion of the ASI dataset. The model was trained on the English training set of PAN (3,000 authors), as explained in section 5.3.7. For the test set, we randomly selected a stratified subset of the ASI dataset, consisting of 4,444 users (*Test set 1* as shown in Table 16).

This experiment resulted in an accuracy of 80.73%, which is comparable to the 82.21% accuracy obtained from the PAN test set. This shows that the model trained on the PAN dataset is generalizable to the ASI test set.

5.4 Discussion

In this work, gender identification was framed as a binary classification problem. Gender identities outside of the binary of female and male fall under the umbrella terms of *non-binary* genders or *genderqueer*. There is an increasing recognition of non-binary gender identities in the legal, medical, and psychological systems [Richards et al. 2016]. However, in this work, due to the available dataset and the methodology that was used to annotate it, we classified gender into two classes of female and male.

A relatively simple system using word and character n-grams and a linear SVM classifier proved strong for gender identification at the PAN 2018 author profiling shared task, which highlights the value of hyper-parameter tuning for the feature sets. For the English dataset, we were able to improve the accuracy by about 1 percentage point by performing dimensionality reduction using LSA on the tf-idf matrix.

Regarding preprocessing, removing character flooding (repeated character sequences) helped improve the accuracy and lower the number of features. This preprocessing measure is especially helpful when dealing with social media corpora, which due to social media's nature, are often noisy.

While the normalized term frequency representation of the offensive expressions resulted in an accuracy well above the majority class baseline, its combination with n-gram features did not show an improvement on the accuracy. This can be due to the fact that the word and character n-grams already capture the information that the offensive expressions contain.

SVM has long proven to have a solid performance; however, with larger datasets, deep learning methods may be able to deliver outstanding results.

5.5 Comparison to Related Work

Our submitted model was the best-performing model in gender identification using textual tweets at the PAN 2018 author profiling shared task, with the accuracy of 0.809, 0.8221, and 0.82 on the Arabic, English, and Spanish datasets respectively.

Table 15 shows the ranking of the participating teams, and their accuracy results in gender identification using textual tweets. In each column, the top three results are made bold.

Table 15. Ranking of the PAN 2018 author profiling shared task participants in text classification

Team Name	Arabic	English	Spanish	Average ▼
daneshvar18	**0.8090**	**0.8221**	**0.8200**	**0.8170**
miranda18	**0.8170**	**0.8121**	0.8005	**0.8099**
vaneerden18	0.7830	**0.8116**	**0.8027**	**0.7991**
sierraloaiza18	**0.8120**	0.8011	0.7827	0.7986
laporte18	0.7910	0.8074	0.7959	0.7981
schuur18	0.7920	0.8074	0.7918	0.7971
takahashi18	0.7710	0.7968	0.7864	0.7847
snijders18	0.7490	0.7926	**0.8036**	0.7817
martinc18	0.7760	0.7900	0.7782	0.7814
lopezsantillan18	0.7760	0.7847	0.7677	0.7761
miller18	0.7590	0.7911	0.7650	0.7717
yigal18	0.7590	0.7911	0.7650	0.7717
pool18	0.7600	0.7853	0.7405	0.7619
gouravdas18	0.7430	0.7558	0.7586	0.7525
vondaeniken18	0.7320	0.7742	0.7464	0.7509
schaetti18	0.7390	0.7711	0.7359	0.7487
aragonsaenzpardo18	0.6480	0.7963	0.7686	0.7376
bayot18	0.6760	0.7716	0.6873	0.7116
gariboiorts18	0.6750	0.7363	0.7164	0.7092
tekir18	0.6920	0.7495	0.6655	0.7023
raiyani18	0.7220	0.7279	0.6436	0.6978
sandroni18	0.6870	0.6658	0.6782	0.6770
karlgren18	–	0.5521	–	–

Chapter 6

6 Gender and Age Classification on the ASI Dataset

6.1 Introduction

In this chapter, we describe our gender and age classification experiments on the ASI dataset. For the gender classification task, which is the main focus of this chapter, we used an SVM model as the classical machine learning baseline and performed several deep learning experiments using the Keras [Chollet 2015] library with TensorFlow [Abadi et al. 2015] backend. Section 6.2 describes the methodology for the gender classification experiments. In section 6.3.14, we discuss our experiments with age classification using SVM as the learning algorithm.

Part of our experiments were executed on Compute Canada's[24] advanced research computing (ARC) infrastructure.

6.2 Methodology

Our approach consists of text preprocessing, text vectorization, and deep neural network design. In this section, we will describe each of these steps, as well as the performance measure and evaluation protocol that was used.

6.2.1 Text Preprocessing

As described in section 4.2.5, we performed the following preprocessing steps in preparation of the ASI dataset:

- All retweets were removed.
- All URLs and *@username* mentions were removed.
- Tweets were lowercased.
- Character flooding was removed. Using NLTK's *TweetTokenizer*, we replaced repeated characters longer than 3 with sequences of length 3.

[24] https://www.computecanada.ca

All non-English tweets were removed. We used the *polyglot* library for language detection.

- The corpus was exported to XML files.

Moreover, we made sure that every user in the corpus has at least 100 words in all their tweets combined. For our experiments on the ASI dataset, after parsing the XML files of the prepared corpus, we merged all tweets of each user into a single string, adding a ' <TweetBorder> ' tag between every two tweets.

6.2.2 Text Vectorization

In order to train a deep learning model, our data should be formatted as tensors. The process of transforming text into numeric tensors is called *text vectorization*. This is done through two steps: tokenization and transformation. First, the text is segmented (tokenized) into words, characters, or a combination of word or character n-grams. Next, the tokens are associated with a vector, through two main techniques: *one-hot encoding* or *token (word/character) embedding*.

For our experiments, we used word embeddings to transform the textual data into numeric tensors. The word embeddings were learned from the dataset, and we experimented with different dimensions for the word embeddings, and different vocabulary sizes, as will be explained in section 6.3.

6.2.3 Deep Neural Network Design

Three key decisions need to be made to build a neural network model:

- **Activation of the last layer**

  The last-layer activation should be chosen in accordance with the problem at hand. For binary classification, a *sigmoid* activation is most suitable, while for multiclass, single-label classification, a *softmax* activation is regarded most appropriate.

  For all of our experiments we used a sigmoid activation for the last layer.

- **Loss function**

  As gender classification is a binary classification problem, we used *binary_crossentropy* as our loss function.

- **Optimization configuration**

  We used *rmsprop* as our optimizer, with its default learning rate in Keras's implementation.

The network architecture used for each experiment is described in detail in section 6.3.

6.2.4 Performance Measure and Evaluation Protocol

Since we approached gender classification as a binary classification problem, we used accuracy as the measure of success.

As for the evaluation protocol, we held out a stratified and randomly selected 20% subset of our dataset as the validation set and another 20% as the test set. In addition, we used the test of the PAN 2018 author profiling shared task (only the English dataset) as a second test set to examine the generalizability of our model.

For our age classification experiments, as will be explained in section 6.3.14, we approached the task as a multiclass, class-balanced classification problem. We have reported precision, recall, and F-measure as the measures of success. The evaluation protocol was similar to that of the gender classification task—maintaining a hold-out validation and test set.

6.3 Experiments and Results

6.3.1 Experimental Plan

As a classical machine learning baseline, we trained our best SVM model from the PAN 2018 author profiling task on the ASI dataset for the gender classification task.

As for the deep learning experiments, we started with a small deep neural network model that is capable of outperforming a simple baseline, such as the majority class baseline. In some literature this is referred to as having *statistical power*. We then tried to develop a model that overfits on the dataset, since a model that underfits is not powerful enough. This was done by increasing the size of the layers (adding more units), training for more epochs, or adding layers. By observing the loss and accuracy of the training and validation, we can find where the model overfits.

Next, we mitigated the overfitting issue by regularizing our model, to find the optimum point between overfitting and underfitting. We can overcome overfitting in a neural network, by adding dropout, performing L1- or L2-regularization, changing the number of units on the layers, and hyper-parameter tuning. Where required, we will add or remove layers and try different architectures.

As a general rule, we start with simpler deep learning architectures, and gradually add to the complexity of the networks, to make sure any further complexity delivers real benefits. If a computationally intensive model results in a lower performance than a cheaper model, the added computational cost would not be justifiable.

During all our experiments, we trained the model on a 60% randomly-selected, stratified subset of the dataset. 20% of the dataset (also randomly selected and stratified) was used for validation, and another 20% of the dataset was held out as the test set. The test set was not seen by the model at no point during the training and development of the model, and was kept for a final testing, to ensure that the model does not overfit on the validation set. Additionally, we used the test set of the PAN 2018 author profiling task as a second test set, to examine the generalizability of our model on a different dataset. Table 16 shows the size of each of the training, validation, and test sets for gender classification experiments. All of these datasets are stratified on gender labels.

Table 16. Size and source of the training, validation, and test sets for gender classifcation experiments

Dataset	Source	Number of users
Training set	60% of the ASI dataset	13,332
Validation set	20% of the ASI dataset	4,444
Test set 1	20% of the ASI dataset	4,444
Test set 2	PAN 2018 author profiling test set	1,900

6.3.2 SVM Baseline

We trained our best SVM model from the PAN 2018 author profiling task, as explained in Chapter 5, with the following features:

- Word unigrams, bigrams, and trigrams
- Character 3- to 5- grams
- Linear dimensionality reduction using latent semantic analysis (LSA) with 300 dimensions was performed on the n-grams.

The following hyper-parameters were used for both the word vectorizer and the character vectorizer:

- max_features: 10^4 and 10^5 (for both the word vectorizer and the character vectorizer)
- min_df: 2
- use_idf: True (default)
- sublinear_tf: True
- max_df: 1.0 (default)

The *max_features* hyper-parameter of each vectorizer controls the size of its vocabulary. Only the *max_features* features with the highest term frequency across the corpus would be considered and all other features would be ignored. We tried various values in the range of 10^3 and 10^7, and found the above two values to give the best results.

For example, when *max_features* is set to 10^4 for both vectorizers, only the top 10,000 word n-grams and the top 10,000 character n-grams would be considered (a total of 20,000 n-gram features). LSA then transforms the tf-idf matrix into a reduced-rank space with the dimensionality of 300.

Table 17. Accuracy scores for the SVM baseline with two configurations, trained on the ASI dataset

Number of n-gram features	Validation accuracy	ASI test set accuracy	PAN test set accuracy
10,000 × 2	83.37%	83.62%	81.58%
100,000 × 2	84.16%	84.77%	80.68%

Increasing the number of n-gram features slightly improved the accuracy score on the ASI validation and test sets and slightly reduced the accuracy score on the PAN test set. However, we did not find this difference to be statistically significant.

The accuracy score of 81.58% obtained on the PAN test set, as seen in Table 17, is very close to the 82.21% accuracy score of our best SVM model at the PAN 2018 author profiling shared task,

which was tested on the same test set, but trained on the PAN training set. This indicates that training our SVM system on the PAN and ASI training sets produces two models with similar predictive power, as evaluated on the PAN test set.

Moreover, in Table 17, the 83.37% and 83.62% accuracy scores on the ASI validation and test set are comparable to the 81.58% accuracy score on the PAN test set, which proves that the SVM model trained on the ASI dataset is generalizable to the PAN 2018 test set.

These similarities also indicate that the quality of the gender labels of our automatically annotated dataset is comparable to that of the PAN 2018 dataset, which was labeled in two steps: automatic annotation and manual verification. Chapter 4 describes the annotation process for both datasets in detail.

6.3.3 Basic Fully Connected Model

We started with a basic, fully connected model. The first layer is word embedding, which gets a 2D tensor as input and outputs a 3D tensor. We then flatten the time-series into a 2D tensor, which removes the notion of time from the input data, similar to what a bag-of-words approach does. We then follow that with two *densely connected* (also called fully connected) layers. Table 18 demonstrates the layers of the model and their output dimensions.

Table 18. Layers of the fully connected model and the shape of their outputs

Layer	**Output Shape**
Embedding	(*num_samples*, 2644, 8)
Flatten	(*num_samples*, 2644 × 8)
Dense 1	(*num_samples*, 32)
Dense 2	(*num_samples*, 1)

Word embeddings are meant to map natural language into a geometric space. As language varies among different cultures and contexts, it is reasonable to learn a new embedding space with every new task [Chollet 2017]. In our experiments, we used the default implementation of Keras for the

embedding layer. The weights of the embedding layer were randomly initialized and learned through backpropagation during the training.

We set the size of the vocabulary to 10,000. This means that only the 10,000 most frequent words in the dataset will be considered as features and all other tokens will be left out of the vocabulary. We tried different values for this parameter in the range of 1,000 and 100,000 and found 10,000 to be the most suitable.

Moreover, we set the embedding layer to only consider the first 2,644 words of each document (all tweets of each user concatenated). More precisely, for each user, only the first 2,644 words that are in the vocabulary (among the 10,000 most frequent words, as described above) are considered and the rest are cut off. This specific value was chosen because this was the maximum number of words in a document in the PAN 2018 dataset, which consisted of only 100 tweets per user. Although we experimented with other values for this variable, as large as 20,000, increasing it did not boost the performance of the model.

The dimensionality of the embeddings was set to 8. We experimented with larger values, but did not find a positive impact on the model.

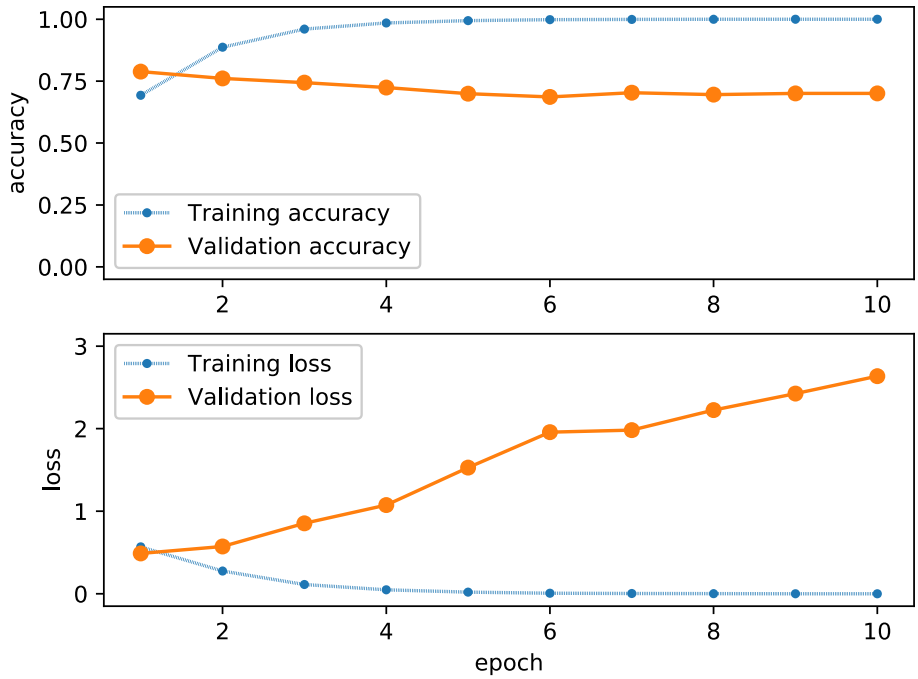

Figure 16. Performance of the fully connected model

As discussed in section 6.2.3, we used *binary_crossentropy* as our loss function and a *sigmoid* activation for the last layer, which are most suitable for binary classification. The model was trained for 10 epochs. Figure 16 displays the training and validation accuracy and loss for this model.

The highest validation accuracy is reached in epoch 1 at 78.83%; however, the model quickly overfits on the training set and the accuracy drops to 70.05% in epoch 10. The training loss converges to zero, while the validation loss keeps increasing. This means that the model is getting further away from the validation data, and hence the decline in the validation accuracy.

6.3.4 Adding Dropout to the Fully Connected Model

Dropout is considered one of the most effective and most common regularization techniques for neural networks. Dropout was developed by Geoff Hinton and his students at the University of Toronto [Srivastava et al. 2014].

Adding dropout to a layer, randomly drops out a number of output features of the layer during training, by setting them to zero. The extent of this is determined by the *dropout rate*, which is the fraction of the features that are set to zero. A common range for dropout is between 0.2 and 0.5.

Table 19. Layers of the fully connected model with dropout

Layer	**Output Shape**
Embedding	(*num_samples*, 2644, 8)
Flatten	(*num_samples*, 2644 × 8)
Dropout 1	(*num_samples*, 2644 × 8)
Dense 1	(*num_samples*, 32)
Dropout 2	(*num_samples*, 32)
Dense 2	(*num_samples*, 1)

Introducing this sort of noise in the output values of a layer by randomly removing a subset of neurons on each example, can break up some patterns in the network and prevent overfitting.

We added two dropout layers with a 0.5 dropout rate, to our initial model, as seen in Table 19.

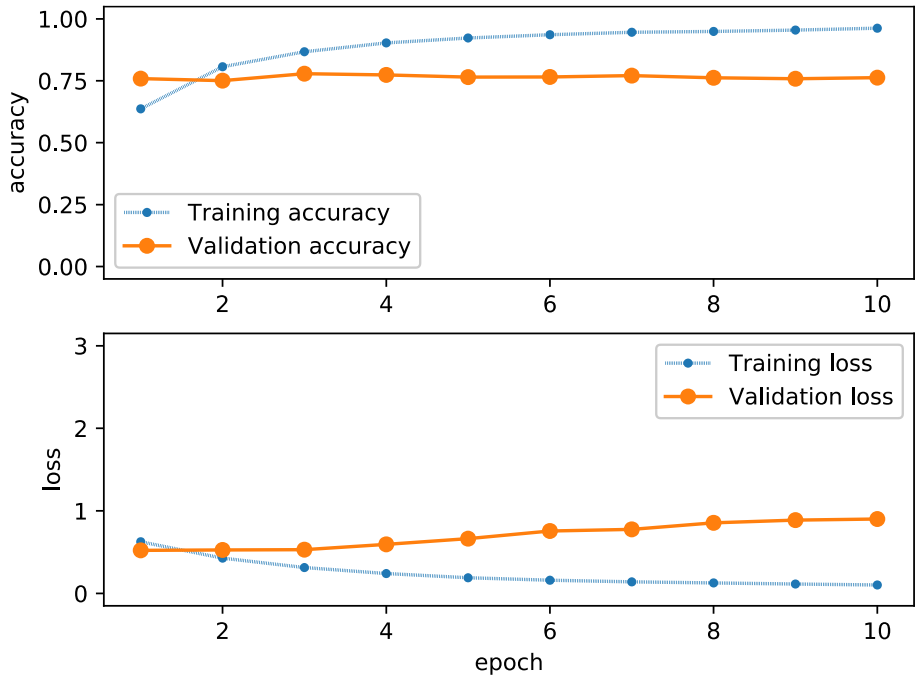

Figure 17. Performance of the fully connected model with dropout

As seen in Figure 17, adding dropout reduced the overfitting to some extent. The steep surge of validation loss is considerably reduced and the validation accuracy does not decrease as much as before. The validation accuracy peaks at 77.86% in epoch 3, and stays around the same amount until the end, with an average of 76.45% throughout all 10 epochs.

6.3.5 Adding Weight Decay to the Fully Connected Model

Another common technique to mitigate overfitting is weight regularization. By forcing the weights of a network to take only small values, we can reduce the complexity of the model, and thus minimize overfitting. This is done by adding a cost to the loss function of the network, for having large weights. There are two types of weight regularization:

- **L1 regularization**

  The added cost is proportional to the L1-norm of the weights (the absolute value of the weight coefficients):

$$\lambda \sum_{i=1}^{n} |\theta_i|$$

- **L2 regularization (also called *weight decay*)**

  The added cost is proportional to the L2-norm of the weights (the square of the weight coefficients):

$$\lambda \sum_{i=1}^{n} \theta_i^2$$

  Where λ is the regularization parameter, which determines how much to penalize the large weights.

We tried L1 and L2 regularization with λ values of 0.05, 0.01, 0.001, and 0.0001 and found L2 regularization with $\lambda = 0.01$ to be most effective. Weight regularization was applied to the first densely connected layer of the network (*Dense 1*). Figure 18 and Figure 19 display the results of two of these experiments. With $\lambda = 0.001$, the validation accuracy peaks at 78.35% in epoch 5, and with $\lambda = 0.01$, the validation accuracy peaks at 79.57% in epoch 4.

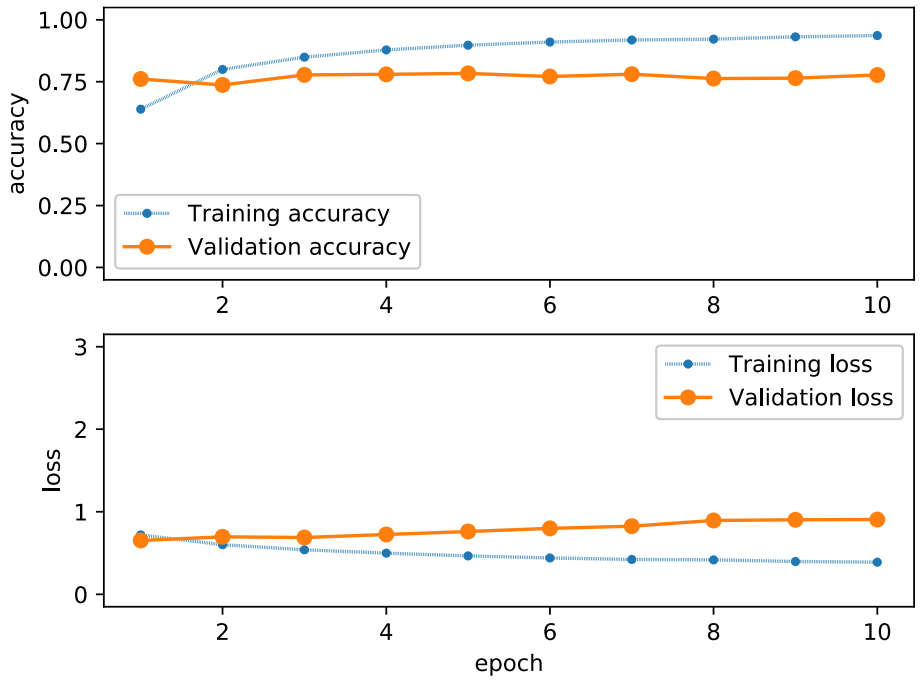

Figure 18. Performance of the fully connected model with dropout and L2 weight regularization = 0.001

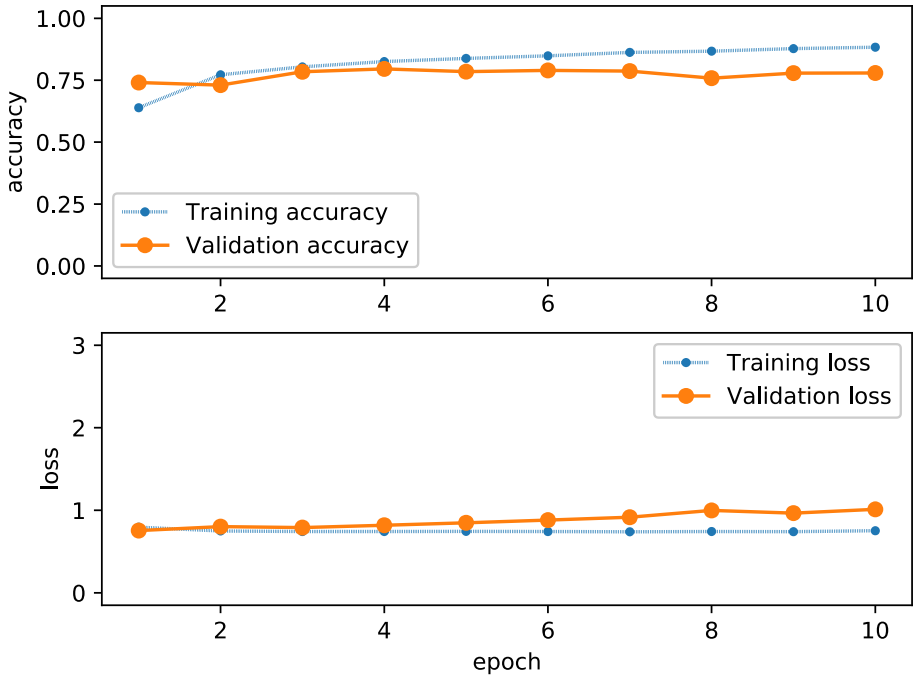

Figure 19. Performance of the fully connected model with dropout and L2 weight regularization = 0.01

6.3.6 Comparison of the Fully Connected Models

To better see the differences in the performance of the four fully connected models described in the previous sections, Figure 20 through Figure 23 display the training and validation loss and accuracy of the four models next to each other.

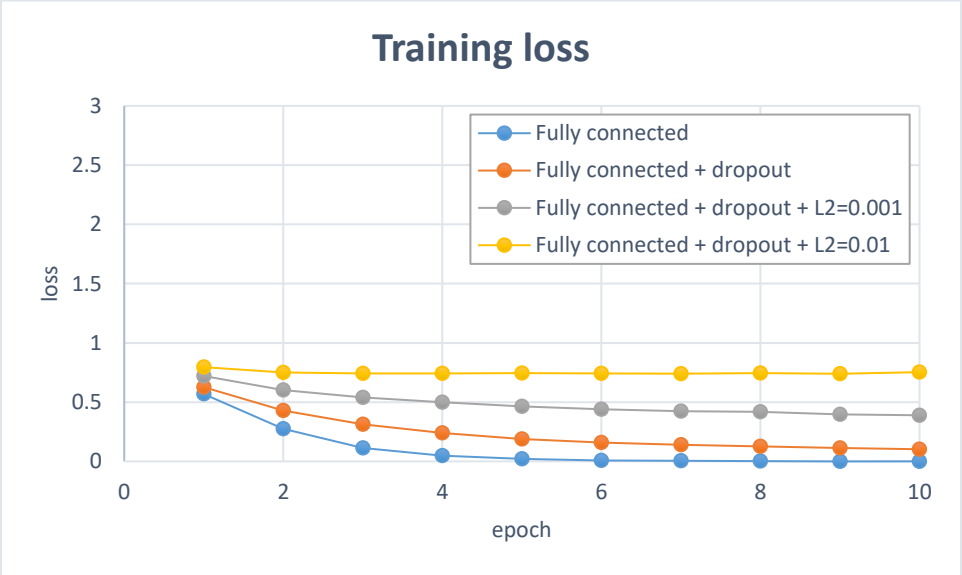

Figure 20. Training loss for the fully connected model, with and without dropout and L2 regularization

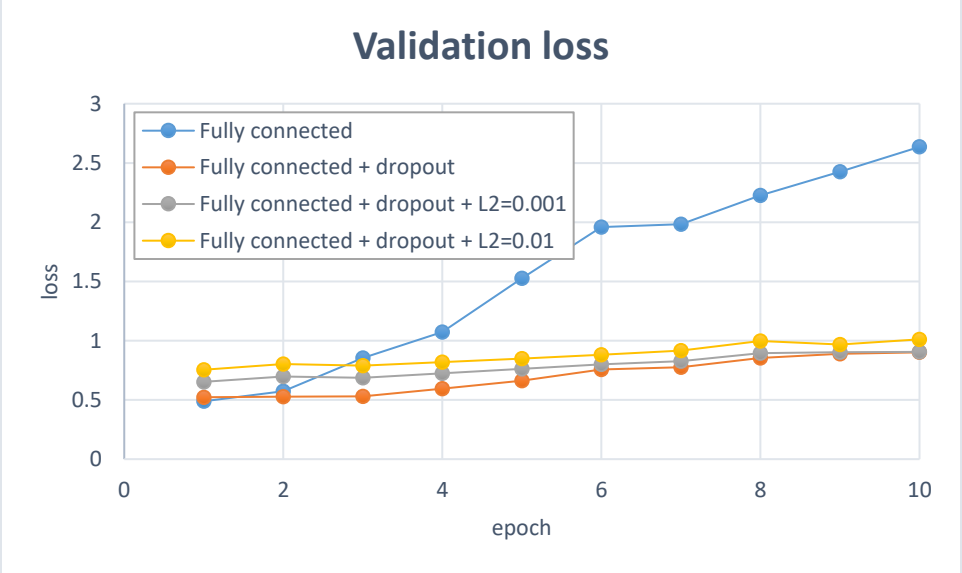

Figure 21. Validation loss for the fully connected model, with and without dropout and L2 regularization

Adding dropout and weight decay considerably reduced the upward trend of validation loss. The model with dropout has a slightly smaller validation loss compared to the models with dropout and weight decay.

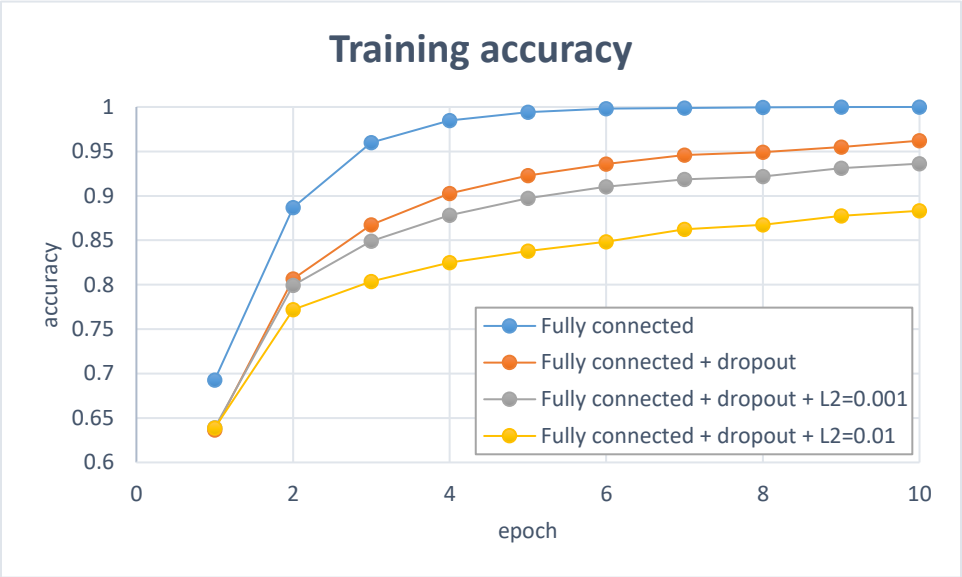

Figure 22. Training accuracy for the fully connected model, with and without dropout and L2 regularization

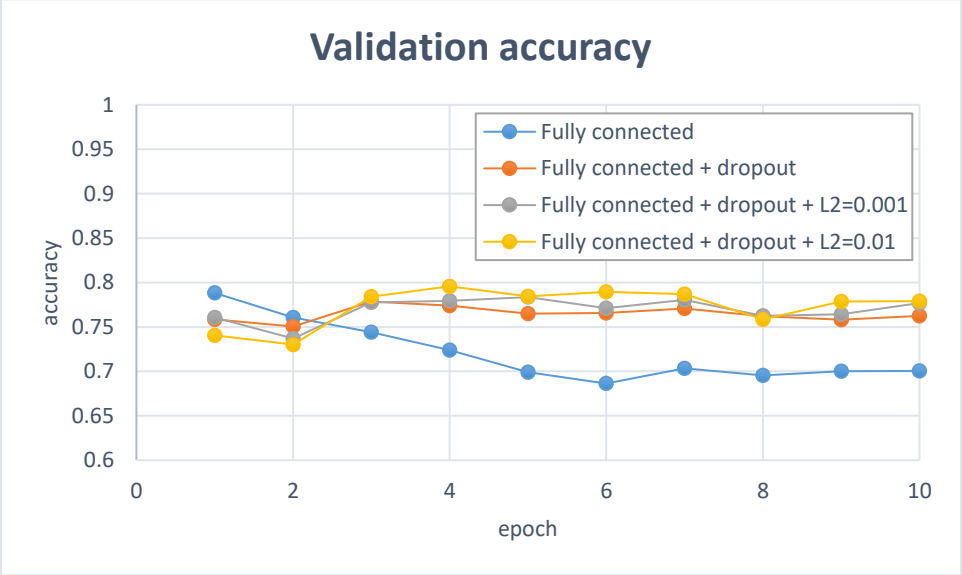

Figure 23. Validation accuracy for the fully connected model, with and without dropout and L2 regularization

As for validation accuracy, all three regularized models perform significantly better than the model without dropout or weight decay. After epoch 3, the model with dropout has a smaller validation

accuracy compared to the two models with L2 regularization. L2 regularization with $\lambda = 0.01$ results in a higher validation accuracy throughout epochs 3 to 10.

Table 20 shows the validation accuracy and validation loss of the four models at the epoch where their highest validation accuracy was observed.

Table 20. Performance of the fully connected models at their peak epoch

Model	Peak epoch	Validation accuracy	Validation loss
Fully connected	1	78.83%	0.4886
Fully connected + dropout	3	77.86%	0.5307
Fully connected + dropout + L2=0.001	5	78.35%	0.7613
Fully connected + dropout + L2=0.01	4	79.57%	0.8191

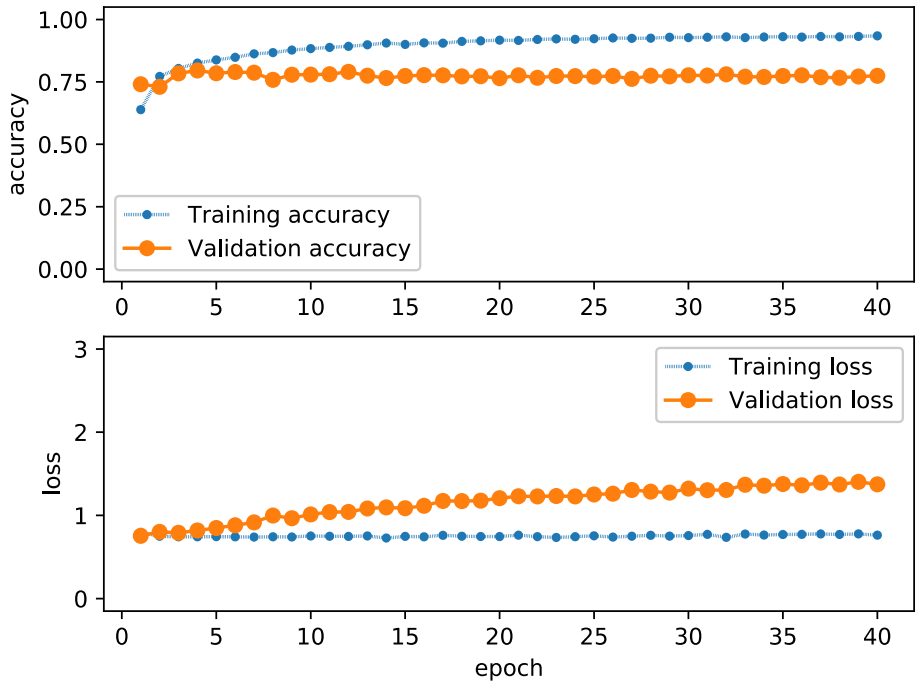

Figure 24. Performance of the fully connected model with dropout and L2 weight regularization = 0.01, trained for 40 epochs

With a validation accuracy higher than all other models—both at its peak and throughout the epochs—we found the model with dropout and L2 regularization of $\lambda = 0.01$ to have the best performance. Lastly, we trained this model for 40 epochs, as demonstrated in Figure 24. The peak of the validation accuracy still happens in epoch 4, as shown in Table 20.

6.3.7 Increasing the Capacity of the Best Fully-Connected Model

Now that we have mitigated the overfitting problem using dropout and weight regularization, it is time to increase the capacity of the network. An ideal model lies somewhere between overfitting and underfitting. One of the ways to get to a model that overfits is to train it for more epochs. At the end of the previous section, we tried to achieve this by training our best model for 40 epochs; however, this did not lead to any improvement. The other two ways are increasing the capacity of the network by increasing the size of the layers and by adding layers to the network.

We ran several experiments to increase the size of the layers in our fully connected model, none of which resulted in any improvement on the validation accuracy scores. We tried the combination of the following hyper-parameter values:

- Size of vocabulary: 10k, 50k, 100k
- Maximum sequence length: 5k, 10k, 20k, 25k
- Dimensionality of word embeddings: 8, 25, 50, 100, 200
- Number of units in the first densely connected layer (*Dense 1*): 32, 64

The next logical step is to investigate other network architectures that are capable of learning more complex patterns.

6.3.8 Recurrent Neural Networks

In order to increase the capacity of the network and capture the inter-word relationships and sentence structure, we opted to use recurrent layers, which have been used for the gender classification task in related work. As explained in Chapter 2, a recurrent neural network (RNN) is a neural network with an internal loop, which maintains a *state* containing information about the previous inputs that the network has processed.

Initial experiments on a simple implementation of RNN using the *SimpleRNN* module of Keras resulted in very poor results. This poor performance is caused by the *vanishing gradient problem*

[Bengio, Simard, and Frasconi 1994], which prevents the RNN from retaining the information about the inputs seen in many timesteps before.

The Long Short-Term Memory (LSTM) algorithm [Hochreiter and Schmidhuber 1997] tackles the vanishing gradient problem by allowing past information to be carried forward across timesteps. In addition, a bidirectional LSTM offers two different representations of the dataset, allowing it to capture patterns that a unidirectional LSTM would miss. A bidirectional LSTM layer consists of two LSTM layers which process the input sequence in different directions: chronologically and anti-chronologically. These two representations are then merged and form the output of the bidirectional LSTM.

In the following sections we discuss our experiments with bidirectional LSTM networks.

6.3.9 Bidirectional LSTM Model

We trained a bidirectional LSTM network, with layers shown in Table 21. The first layer is word embedding, with dimensionality of 32. The second layer is a bidirectional LSTM layer with 32 units (total of 64 units) and a *hyperbolic tangent (tanh)* activation. The third and last layer is a dense layer with a single unit and a *sigmoid* activation. The model was trained for 20 epochs using *binary_crossentropy* as the loss function. Figure 25 displays the training and validation accuracy and loss for the model.

Table 21. Layers of the bidirectional LSTM network

Layer	**Output Shape**
Embedding	(*num_samples*, 2644, 32)
Bidirectional LSTM	(*num_samples*, 64)
Dense	(*num_samples*, 1)

As seen in Figure 25, the model overfits the training data in epoch 5, after which validation loss begins to increase and validation accuracy drops. In epoch 5, validation loss reaches its minimum at 0.5472 and validation accuracy reaches its peak at 76.85%, which is less than the 79.57% accuracy of our best fully connected model.

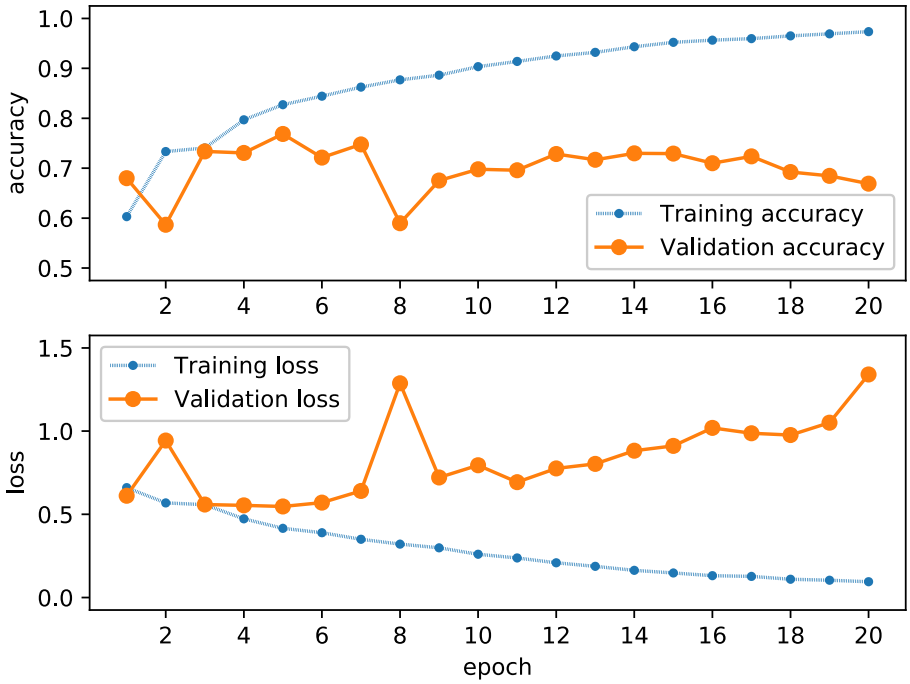

Figure 25. Performance of the bidirectional LSTM model

6.3.10 Adding Dropout to the Bidirectional LSTM Model

To mitigate overfitting in our LSTM model, we use dropout as our regularization technique; however, simply applying dropout before a recurrent layer would hamper its ability to learn, rather than improve it. As described in [Gal 2016], the appropriate way to use dropout with a recurrent network is to apply the same dropout mask at every timestep, rather than applying a temporally random mask at each timestep.

In a similar manner, the pattern of dropped units applied to the inner recurrent activations of the LSTM layer (referred to as a *recurrent* dropout mask) should also remain constant throughout all timesteps. Applying a temporally random dropout mask would undermine the network's ability to properly propagate its learning error through time.

In our Keras implementation, we used the value of 0.1 for both the dropout and the recurrent dropout. Figure 26 demonstrates the training and validation accuracy and loss for this model.

As seen in Figure 26, validation loss does not raise as much, compared to the model without dropout. Moreover, validation loss and validation accuracy do not fluctuate as much as they did in the previous model, specifically at epochs 2 and 8.

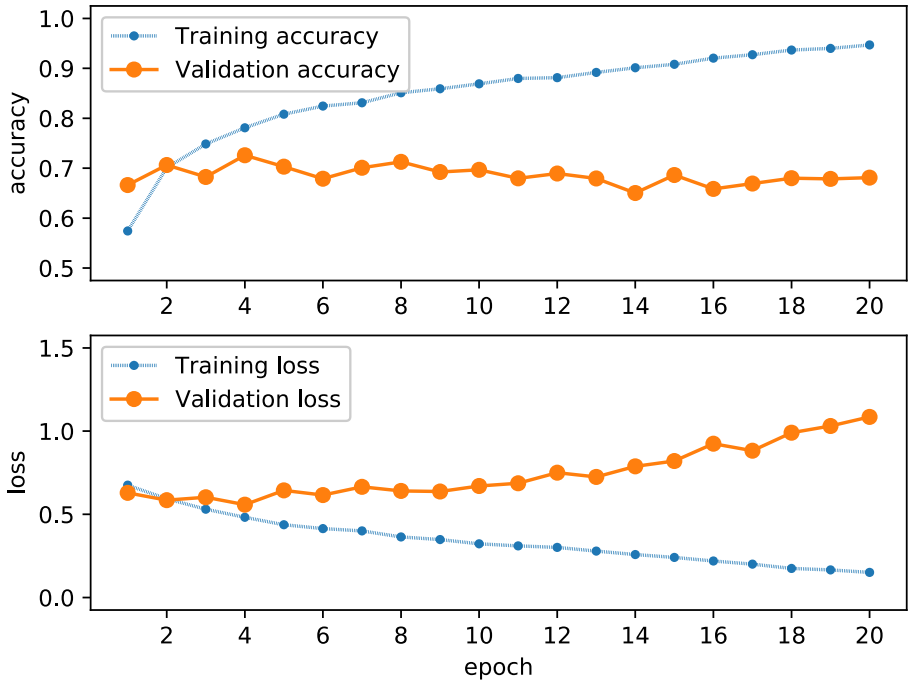

Figure 26. Performance of the bidirectional LSTM model with dropout

In epoch 4, validation loss reaches its minimum at 0.5581 and validation accuracy reaches its maximum at 72.59%. Although the LSTM model has shown a lower performance at this task, compared to the fully connected neural network and the SVM baseline, we will go one step further to add a second LSTM layer to the network, to investigate whether increasing the capacity of the recurrent network would improve the performance of the model.

6.3.11 Stacked Bidirectional LSTM Model

Adding layers to a neural network is one of the ways to increase its capacity. To stack recurrent layers on top of each other, all recurrent layers except the last one should return the full sequence of their outputs [Chollet 2017], which is a 3D tensor with shape *(batch_size, timesteps, units)*, instead of the output of the last timestep, which is a 2D tensor with shape *(batch_size, units)*. In Keras, this is done by setting the *return_sequences* argument of the intermediate recurrent layers to *True*. Table 22 presents the layers of the stacked bidirectional LSTM network and the shape of their outputs.

Same as before, the first layer is word embedding, with dimensionality of 32. The second and third layers are bidirectional LSTM layers with 32 units (total of 64 units) and 64 units (total of 128 units), respectively; both with a *hyperbolic tangent (tanh)* activation. As for regularization, dropout

and recurrent dropout with the value of 0.1 was applied on both LSTM layers. Finally, the fourth and last layer is a dense layer with a single unit and a *sigmoid* activation. The model was trained for 20 epochs using *binary_crossentropy* as the loss function.

Table 22. Layers of the stacked bidirectional LSTM network

Layer	**Output Shape**
Embedding	(*num_samples*, 2644, 32)
Bidirectional LSTM 1	(*num_samples*, 2644, 64)
Bidirectional LSTM 2	(*num_samples*, 128)
Dense	(*num_samples*, 1)

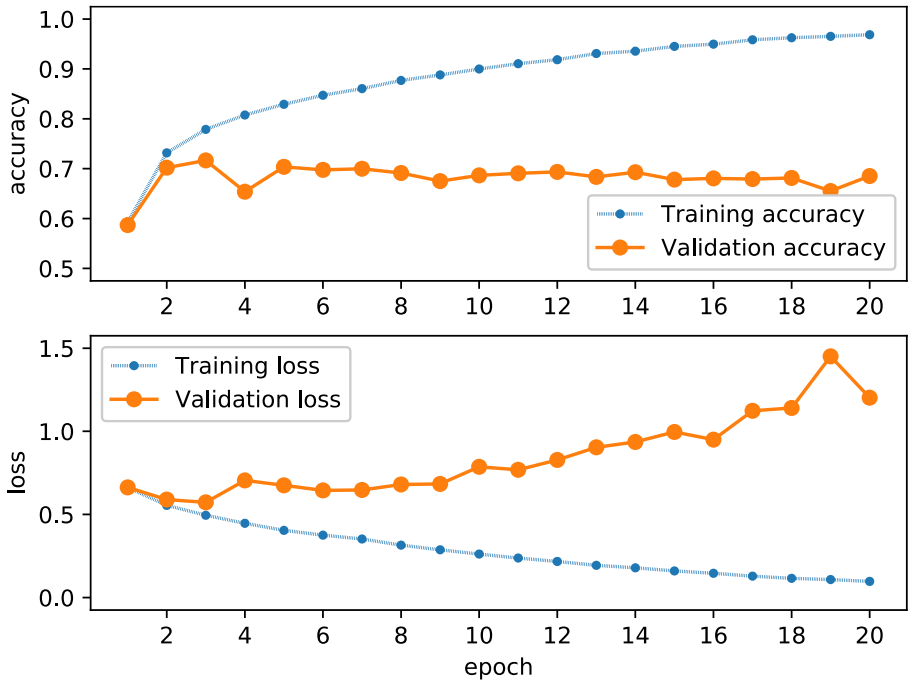

Figure 27. Performance of the stacked bidirectional LSTM model with dropout

As seen in Figure 27, adding a second recurrent layer was able to push the model towards overfitting. The training loss and training accuracy come closer to their extreme values (0 and 1,

respectively), more so than in the previous model. The model overfits in epoch 3, with validation loss reaching its minimum at 0.5729 and validation accuracy reaching its maximum at 71.65%.

Lastly, as a second regularization measure and to mitigate overfitting, we applied L2 weight regularization (weight decay) to the recurrent layers. On both LSTM layers, we applied L2 weight regularization with $\lambda = 0.001$ to the *kernel* weights matrix as well as the *recurrent_kernel* weights matrix. Figure 28 demonstrates the performance of this model.

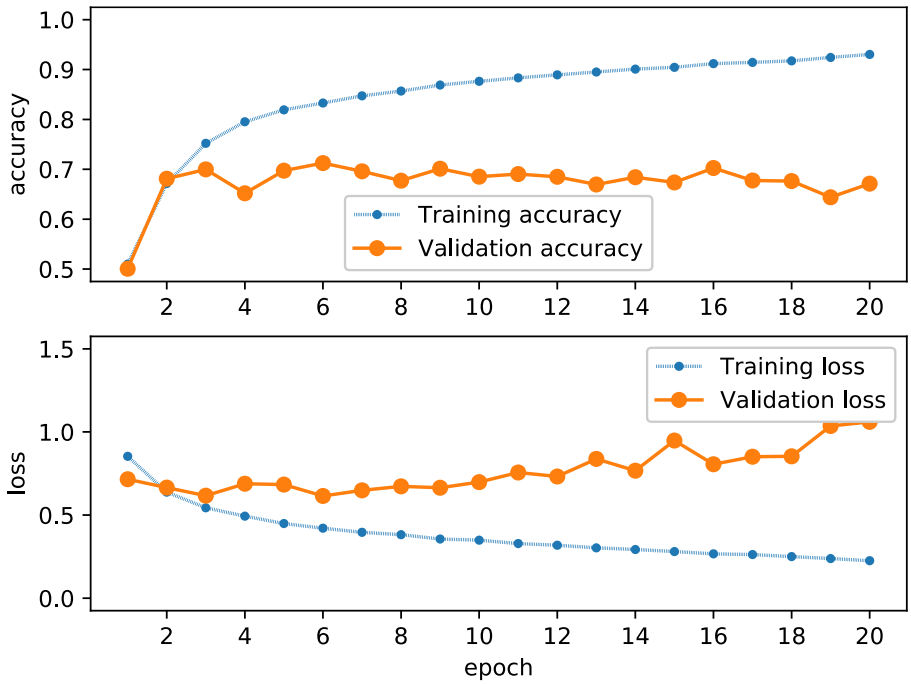

Figure 28. Performance of the stacked bidirectional LSTM model with dropout and L2 weight regularization

Applying weight decay to the network, adds a penalty proportional to the square value of the weight coefficients to the total loss of the network. Since this penalty is only added during training, it results in higher loss values at training time than at validation. You can see this effect in Figure 28. In addition, the validation loss does not increase as much as the model without weight regularization, at epochs 10 through 20. However, no significant improvement is seen in the validation accuracy. At epoch 6, validation loss reaches its minimum at 0.6149 and validation accuracy reaches its maximum at 71.24%.

In the next section, we compare the performance of the four bidirectional LSTM models with each other, and with the best performing fully connected model.

6.3.12 Comparison of the Bidirectional LSTM Models

Table 23 shows the validation accuracy and validation loss for the bidirectional LSTM models at their peak epoch, as well as the best performing fully connected model. To compare the performance of the LSTM models throughout all 20 epochs, refer to Figure 29 through Figure 32 in Appendix C.

Table 23. Performance of the bidirectional LSTM models at their peak epoch, compared to the best fully connected model

Model	Peak epoch	Validation accuracy ▼	Validation loss
Fully connected + dropout + L2=0.01	4	79.57%	0.8191
Bidirectional LSTM	5	76.85%	0.5472
Bidirectional LSTM + dropout	4	72.59%	0.5581
Stacked bidirectional LSTM + dropout	3	71.65%	0.5729
Stacked bidirectional LSTM + dropout + L2	6	71.24%	0.6149

As seen in Table 23, none of the recurrent networks were able to outperform the network with densely connected layers at this task. According to our experimental plan, we started with simpler network architectures, and step by step added to the complexity of our networks to make sure any further complexity delivers real benefits, and that the added computational costs are justifiable. However, the results show that a densely connected deep neural network, which has a considerably less computational cost, delivers a higher accuracy score than any of the RNN networks. On our system, the training and validation process took less than 12 minutes for the fully connected model, compared to a few hours for any of the bidirectional LSTM models.

6.3.13 Evaluating the Best Learning Model on the Test Sets

To ensure no overfitting has occurred on the validation set, we tested our best deep neural network model (fully connected, with dropout and L2 weight regularization) on our two test sets. Table 24 reviews the architecture of this model.

Table 24. Layers, their output shapes, and hyper-parameters for the fully connected model with dropout and L2 weight regularization

Layer	Output Shape	Hyper-parameters
Embedding	(*num_samples*, 2644, 8)	Dimensionality = 8 Size of vocabulary = 10,000 Maximum input length = 2644
Flatten	(*num_samples*, 2644 × 8)	
Dropout 1	(*num_samples*, 2644 × 8)	Dropout rate = 0.5
Dense 1	(*num_samples*, 32)	L2 weight regularization with $\lambda = 0.01$
Dropout 2	(*num_samples*, 32)	Dropout rate = 0.5
Dense 2	(*num_samples*, 1)	Activation: *sigmoid*

The model was trained for 4 epochs, and *binary_crossentropy* was used as the loss function. Table 25 shows the performance of this model as compared with the SVM baseline. Both models were trained on the ASI training set—a stratified 60% subset of the ASI dataset, as demonstrated in Table 16. The PAN test set is the official English test set of the PAN 2018 author profiling shared task.

Table 25. Performance of the best deep learning model as compared with the SVM baseline

Model	Validation accuracy ▼	ASI test set accuracy ▼	PAN test set accuracy ▼
SVM baseline	83.37%	83.62%	81.58%
Fully connected + dropout + L2=0.01, trained for 4 epochs	79.57%	80.31%	77.16%

The accuracy scores of the SVM baseline is higher than that of the fully connected deep neural network model by 3.80 percentage points (p.p.), 3.31 p.p., and 4.42 p.p. on the validation set, the ASI test set, and the PAN test set, respectively. It is worth mentioning that on our machine, the SVM model took about 104 minutes to train and 17 minutes to test (a total of 2 hours), as opposed to a 12-minute run time for the fully connected deep neural network model.

6.3.14 Age Classification Experiments

In this section we describe our age classification experiments on the ASI dataset. The main challenge for these experiments is the quality of the age labels. As previously discussed in section 4.2, the assigned probabilities for age groups were considerably lower than that of gender groups. While 52% of the users in the ASI dataset had an assigned gender label with a confidence level of more than 98%, only 3.4% of them had an assigned age group with the same confidence level. This is mainly because there are many gender-specific first names, while associating a first name with a specific age group is only possible for a very small proportion of first names.

For the age classification experiments, we randomly selected a subset of the ASI dataset which was stratified on age groups, consisting of 11,160 users. We then separated this dataset into three randomly selected, stratified sets: 60% for training, 20% for validation, and 20% for test.

We approached the task as a multiclass classification problem. For our learning algorithm, we used scikit-learn's implementation of linear SVM, which handles multiclass classification by transforming it to a binary classification problem using the one-vs-rest scheme.

Similar to our SVM baseline for gender classification, we trained an SVM classifier with a linear kernel, using the following as features:

- Word unigrams, bigrams, and trigrams. Vocabulary size = 10^6
- Character 3- to 5- grams. Vocabulary size = 10^6
- Linear dimensionality reduction using latent semantic analysis (LSA) with 300 dimensions performed on the n-grams

We tried different values for the size of the n-grams vocabulary and the rank of the LSA reduced space, and found the above values to give the highest results. Table 26 displays precision, recall, and F_1 score for each class.

While the obtained scores are well above the 0.167 majority class baseline, the overall poor performance is resulted by the limited quality of the age labels in the dataset. This is also reflected in the higher F_1 scores of the [0, 25], [25, 34], and [65, …] classes, which were the three age groups with the highest quality of age labels, as demonstrated in section 4.2.4.

Table 26. Performance measures for each age class

Class (age group)	Precision	Recall	F_1 Score	Support
[0, 25]	0.30	0.34	0.32	372
[25, 34]	0.28	0.32	0.30	372
[35, 44]	0.23	0.20	0.21	372
[45, 54]	0.24	0.15	0.18	372
[55, 64]	0.25	0.25	0.25	372
[65, …]	0.26	0.32	0.29	372

6.4 Discussion

A deep neural network consisting of a word embedding layer, flattening, and two densely connected layers performed better than RNN networks, including bidirectional LSTM, at gender classification.

An SVM classifier using word 1- to 3-gram and character 3- to 5-gram features and dimensionality reduction by means of LSA, achieved accuracy of 83.37% and 83.62% on the ASI validation and test set, respectively. The same model (trained on the ASI training set) obtained an accuracy of 81.58% when tested on the test set of the PAN 2018 author profiling shared task. This results is comparable to the 82.21% accuracy score of our SVM model trained on the PAN training set, which proves the generalizability of the model.

Meanwhile, the fully connected deep neural network model achieved lower accuracy scores than the SVM baseline by 3.80 percentage points (p.p.) and 3.31 p.p. on the ASI validation and test set, respectively. This is consistent with the findings of related work at the PAN shared tasks, where SVM classifiers have a history of being the front-runner.

While the deep learning models investigated in this work did not achieve higher accuracy scores than the SVM baseline, it is possible that other network architectures with more layers or more units on each layer can learn more complex patterns and outperform an SVM classifier at the same

task. Moreover, a larger dataset may improve the classification model and increase the statistical soundness of the results.

As for age classification, the limited quality of the age labels in the corpus, impacted the ability of the learning algorithm to train an accurate model. Training the same system on a dataset with higher-quality age labels would potentially deliver better results.

Chapter 7

7 Conclusion and Future Work

7.1 Discussion and Conclusion

In this work, we probed a variety of deep learning architectures and classical machine learning algorithms to construct a gender classification model trained on textual tweets.

We obtained state-of-the-art results in gender classification with a relatively simple system using a combination of word and character n-grams as features and SVM with a linear kernel as the learning algorithm. We found the combination of word 1- to 3-grams and character 3- to 5-grams features to deliver the best results as measured by accuracy. Moreover, tf-idf weighting with sublinear tf scaling and dimensionality reduction using LSA were found to be beneficial.

Regarding preprocessing, removing URLs and character flooding (repeated character sequences), and excluding out-of-vocabulary features, helped improve the accuracy and lower the number of features. These preprocessing measures are especially helpful when working with social media corpora, which are often noisy, due to the nature of social media.

A feedforward deep neural network consisting of a word embedding layer, flattening, and two densely connected layers outperformed our bidirectional LSTM networks. Nonetheless, other LSTM architectures are worth exploring, and larger models may be able to outperform our fully connected model.

Regularization techniques proved effective in mitigating overfitting. Dropout proved effective in all network architectures, when applied appropriately. As for weight regularization, L2 regularization (weight decay) proved more effective than L1 regularization. The regularization parameter, which determines how much to penalize the large weights, took very different optimum values for different network architectures. This underlines the importance of hyper-parameter optimization on the regularization parameters.

While the deep learning models investigated in this work did not achieve higher accuracy scores than the SVM model, it is probable that other network architectures with more layers or more units on each layer can learn more complex patterns and outperform an SVM classifier at the same task. Exploring a broader hypothesis space using a more extensive hyper-parameter tuning process may

also result in discovering better models. Moreover, a larger dataset may improve the classification model and increase the statistical soundness of the results.

Training our SVM system on the ASI and PAN training sets resulted in two models with similar predictive power, as evaluated on the PAN test set, with accuracy scores of 81.58% and 82.21%, respectively. Moreover, the SVM model trained on the ASI training set, achieved comparable results on the ASI and PAN datasets, which indicates the generalizability of the model.

7.2 Contributions

The contributions of this work to the author profiling body of research are as follows:

- A survey of the studies on gender and age identification from written documents, and the current state-of-the-art in social media author profiling.
- A gender classification system utilizing a combination of word and character n-grams as features, dimensionality reduction using Latent Semantic Analysis (LSA), and a linear Support Vector Machine (SVM) classifier.

  Our model achieved the highest performance in textual classification among 23 participating teams at the PAN 2018 author profiling shared task, with an 82.21%, 82.00%, and 80.90% accuracy score on the English, Spanish, and Arabic test set, respectively.

  Our approach was published in [Daneshvar and Inkpen 2018] as part of the CLEF 2018 Working Notes, with our source code publicly available at:
  https://github.com/SamanDaneshvar/pan18ap
- A dataset of more than 22,000 Twitter users in Canada and their public tweets posted in the span of 2012 to 2018, labeled with gender.
- Several deep learning gender classification systems, including:
  - A feedforward neural network consisting of a word embedding layer, flattening, and two densely-connected layers, achieving 79.57% validation accuracy.
  - A bidirectional LSTM neural network, achieving 76.85% validation accuracy.

  We have made the source code to our approach publicly available at:
  https://github.com/SamanDaneshvar/usermodeling

7.3 Future Work

7.3.1 Potential Improvements

Exploring other deep learning architectures, such as Convolutional Neural Networks (CNNs) and bidirectional LSTM with more layers or more units on each layer can lead to a model with a higher performance. Moreover, a more extensive hyper-parameter optimization process may improve the models.

Pre-trained word embeddings, such as ELMo, BERT, FastText, GloVe, and Word2Vec may provide superior representations for words, and result in better classification models.

In this work, we carried out some experiments on age classification. However, due to the limited quality of the age labels in our corpus, we did not obtain a high accuracy. The same system may achieve a much higher accuracy, if trained on a dataset with high-quality labels. Moreover, a different way of separating age groups may lead to more distinct age characteristics.

Lastly, larger datasets may improve the classification model and increase the statistical soundness of the results.

7.3.2 Extensions

In addition to gender and age, other traits of a social media user can be learned based on their activity on social media. In what follows, we name some of these traits and provide examples of the research that has been done in these areas:

Personality traits

Text of social media posts can be used to identify the personality traits of the author, as well as the emotions expressed in individual posts.

In early works, Oberlander and Nowson [2006] used blog texts to classify the level of four personality traits of bloggers: neuroticism, extraversion, agreeableness, and conscientiousness. Similarly, Celli [2012] developed an unsupervised system for personality detection based on the five personality traits in the *Big Five* model in psychology, also known as the *OCEAN* model: openness to experience, conscientiousness, extraversion, agreeableness, and neuroticism.

In a more recent work, Pereira Junior and Inkpen [2017] used IBM Watson's personality detection algorithms to obtain five scores for the Big Five personality traits, and trained a deep learning model on them.

Geolocation

A large body of work has been done on the detection of the physical location of users based on their activity on social media platforms. A great source of information about a user's location is the location of their *friends*—people in their social network. Rout et al. [2013] hypothesized that in many cases, a person's social network is sufficient to detect their location. They used an SVM classifier to identify the city of the users based on the known locations of their social ties.

The content of social media posts is another source of information about a user's location [Farzindar and Inkpen 2015]. One of the early works in this area was done by Roller et al. [2012], who adopted a variant of the K-Nearest Neighbors classifier to predict the location of authors. Utilizing training documents labeled with latitude/longitude coordinates, they partitioned the surface of the Earth into grids and constructed pseudo-documents by merging all the documents within each grid. Afterwards, the location of each test document was determined based on the location of the most similar pseudo-document.

Language variety

Language variety identification was addressed in the author profiling shared task at PAN 2017 [Rangel et al. 2017], with a dataset consisting of various dialects of the Arabic, English, Portuguese, and Spanish languages. The highest overall accuracy was obtained by a simple LDR method, which represented documents based on the probability distribution of occurrence of the words in each class. This indicates that language variety is closely tied to the usage of words.

Bibliography

Abadi, Martin, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2015. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems." http://tensorflow.org/.

Agrawal, Madhulika, and Teresa Gonçalves. 2016. "Age and Gender Identification Using Stacking for Classification: Working Notes of CLEF 2016." In *CEUR Workshop Proceedings*, 18:785–90. http://ceur-ws.org/Vol-1609/16090785.pdf.

Álvarez-Carmona, Miguel A, A Pastor López-Monroy, Manuel Montes-y-Gómez, Luis Villaseñor-Pineda, and Hugo Jair Escalante. 2015. "INAOE's Participation at PAN'15: Author Profiling Task: Notebook for PAN at CLEF 2015." In *CLEF 2015 Labs and Workshops, Notebook Papers. CEUR Workshop Proceedings.*, edited by Linda Cappellato, Nicola Ferro, Gareth Jones, and Eric San Juan. CEUR-WS.org.

Aragón, Mario Ezra, and Adrián Pastor López-Monroy. 2018. "A Straightforward Multimodal Approach for Author Profiling: Notebook for PAN at CLEF 2018." In *Proceedings of the Ninth International Conference of the CLEF Association (CLEF 2018).* http://ceur-ws.org/Vol-2125/paper_96.pdf.

Argamon, Shlomo, Moshe Koppel, Jonathan Fine, and Anat Rachel Shimoni. 2003. "Gender, Genre, and Writing Style in Formal Written Texts." *Text* 23 (3): 321–46. https://doi.org/10.1515/text.2003.014.

Arroju, Mounica, Aftab Hassan, and Golnoosh Farnadi. 2015. "Age, Gender and Personality Recognition Using Tweets in a Multilingual Setting: Notebook for PAN at CLEF 2015." In *CLEF 2015 Labs and Workshops, Notebook Papers. CEUR Workshop Proceedings.* Vol. 1391. http://ceur-ws.org/Vol-1391/57-CR.pdf.

Basile, Angelo, Gareth Dwyer, Maria Medvedeva, Josine Rawee, Hessel Haagsma, and Malvina Nissim. 2017. "N-GrAM: New Groningen Author-Profiling Model: Notebook for PAN at CLEF 2017." In *CLEF 2017 Evaluation Labs and Workshop -- Working Notes Papers, 11-14 September, Dublin, Ireland*, edited by Linda Cappellato, Nicola Ferro, Lorraine Goeuriot,

and Thomas Mandl. CEUR-WS.org. http://ceur-ws.org/Vol-1866/.

Bayot, Roy Khristopher, and Teresa Gonçalves. 2018. “Multilingual Author Profiling Using LSTMs: Notebook for PAN at CLEF 2018.” In *CEUR Workshop Proceedings*. Vol. 2125. http://ceur-ws.org/Vol-2125/paper_185.pdf.

Beales, Howard. 2009. “The Value of Behavioral Targeting.”

Bengio, Y, P Simard, and P Frasconi. 1994. “Learning Long-Term Dependencies with Gradient Descent Is Difficult.” *IEEE Transactions on Neural Networks* 5 (2): 157–66. https://doi.org/10.1109/72.279181.

Biber, Douglas. 1995. *Dimensions of Register Variation: A Cross-Linguistic Comparison*. Cambridge University Press.

Bilan, Ivan, and Desislava Zhekova. 2016. “CAPS: A Cross-Genre Author Profiling System: Working Notes of CLEF 2016.” In *CEUR Workshop Proceedings*, 824–35. http://ceur-ws.org/Vol-1609/16090824.pdf.

Bird, Steven, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O’Reilly Media Inc.

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. “Enriching Word Vectors with Subword Information.” *ArXiv Preprint ArXiv:1607.04606*.

Boser, Bernhard E, Isabelle M Guyon, and Vladimir N Vapnik. 1992. “A Training Algorithm for Optimal Margin Classifiers.” In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 144–52. COLT ’92. New York, NY, USA: ACM. https://doi.org/10.1145/130385.130401.

Bougiatiotis, Konstantinos, and Anastasia Krithara. 2016. “Author Profiling Using Complementary Second Order Attributes and Stylometric Features: Working Notes of CLEF 2016.” In *CEUR Workshop Proceedings*, 1609:836–45. http://ceur-ws.org/Vol-1609/16090836.pdf.

Burger, John D, John Henderson, George Kim, and Guido Zarrella. 2011. “Discriminating Gender

on Twitter." In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 1301–9. EMNLP '11. Stroudsburg, PA, USA: Association for Computational Linguistics. http://dl.acm.org/citation.cfm?id=2145432.2145568.

Celli, Fabio. 2012. "Unsupervised Personality Recognition for Social Network Sites." *ICDS 2012 The Sixth International Conference on Digital Society*, no. c: 59–62.

Chollet, François. 2015. "Keras."

———. 2017. *Deep Learning with Python*. 1st ed. Greenwich, CT, USA: Manning Publications Co.

Ciccone, Giovanni, Arthur Sultan, Léa Laporte, Elöd Egyed-Zsigmond, Alaa Alhamzeh, and Michael Granitzer. 2018. "Stacked Gender Prediction from Tweet Texts and Images: Notebook for PAN at CLEF 2018." In *Proceedings of the Ninth International Conference of the CLEF Association (CLEF 2018)*. http://ceur-ws.org/Vol-2125/paper_111.pdf.

Ciobanu, Alina Maria, Marcos Zampieri, Shervin Malmasi, and Liviu P Dinu. 2017. "Including Dialects and Language Varieties in Author Profiling: Working Notes of CLEF 2017." In *CEUR Workshop Proceedings*. http://ceur-ws.org/Vol-1866/paper_83.pdf.

"Company Info." 2019. Facebook Newsroom. 2019. https://newsroom.fb.com/company-info/.

Cortes, Corinna, and Vladimir Vapnik. 1995. "Support-Vector Networks." *Machine Learning* 20 (3): 273–97. https://doi.org/10.1007/BF00994018.

D'Agostino, Ralph B. 1971. "An Omnibus Test of Normality for Moderate and Large Size Samples." *Biometrika* 58 (2): 341–48. https://doi.org/10.1093/biomet/58.2.341.

D'Agostino, Ralph B, and E S Pearson. 1973. "Tests for Departure from Normality. Empirical Results for the Distributions of B2 and √b1." *Biometrika* 60 (3): 613–22. https://doi.org/10.1093/biomet/60.3.613.

Daneshvar, Saman, and Diana Inkpen. 2018. "Gender Identification in Twitter Using N-Grams and LSA: Notebook for PAN at CLEF 2018." In *CEUR Workshop Proceedings*, 2125:1–10. http://ceur-ws.org/Vol-2125/paper_213.pdf.

Däniken, Pius von, Ralf Grubenmann, and Mark Cieliebak. 2018. “Word Unigram Weighing for Author Profiling at PAN 2018: Notebook for PAN at CLEF 2018.” In *Proceedings of the Ninth International Conference of the CLEF Association (CLEF 2018).* http://ceur-ws.org/Vol-2125/paper_135.pdf.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. “BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding.” https://doi.org/arXiv:1811.03600v2.

Dhingra, Bhuwan, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, and William W. Cohen. 2016. “Tweet2Vec: Character-Based Distributed Representations for Social Media.” *ArXiv Preprint ArXiv:1605.03481*. http://arxiv.org/abs/1605.03481.

Dias, Rafael Felipe Sandroni, and Ivandré Paraboni. 2018. “Author Profiling Using Word Embeddings with Subword Information: Notebook for PAN at CLEF 2018.” In *Proceedings of the Ninth International Conference of the CLEF Association (CLEF 2018).*, 1–5. http://ceur-ws.org/Vol-2125/paper_97.pdf.

Dumais, Susan T. 2005. “Latent Semantic Analysis.” *Annual Review of Information Science and Technology* 38 (1): 188–230. https://doi.org/10.1002/aris.1440380105.

Dumais, Susan T., G W Furnas, Thomas K. Landauer, S Deerwester, and R Harshman. 1988. “Using Latent Semantic Analysis to Improve Access to Textual Information.” In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 281–85. CHI ’88. New York, NY, USA: ACM. https://doi.org/10.1145/57167.57214.

Farzindar, Atefeh, and Diana Inkpen. 2015. “Natural Language Processing for Social Media.” *Synthesis Lectures on Human Language Technologies* i (2): 1 PDF (xix, 146 pages). https://doi.org/10.2200/S00659ED1V01Y201508HLT030.

Franco-Salvador, Marc, Nataliia Plotnikova, Neha Pawar, and Yassine Benajiba. 2017. “Subword-Based Deep Averaging Networks for Author Profiling in Social Media: Working Notes of CLEF 2017.” In *CEUR Workshop Proceedings*. Vol. 1866. http://ceur-ws.org/Vol-1866/paper_192.pdf.

Gal, Yarin. 2016. “Uncertainty in Deep Learning.” University of Cambridge.

Garimella, Aparna, and Rada Mihalcea. 2016. “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 (PEOPLES)*, 1–10. Osaka, Japan: The COLING 2016 Organizing Committee. https://www.aclweb.org/anthology/W16-4301.

Gencheva, Pepa, Martin Boyanov, Elena Deneva, Preslav Nakov, Georgi Georgiev, Yasen Kiprov, and Ivan Koychev. 2016. “PANcakes Team: A Composite System of Domain-Agnostic Features For Author Profiling: Working Notes of CLEF 2016.” In *CEUR Workshop Proceedings*, 1609:874–80. http://ceur-ws.org/Vol-1609/16090874.pdf.

Guyon, Isabelle, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. 2002. “Gene Selection for Cancer Classification Using Support Vector Machines.” *Machine Learning* 46 (1): 389–422. https://doi.org/10.1023/A:1012487302797.

HaCohen-Kerner, Yaakov, Yair Yigal, Elyashiv Shayovitz, Daniel Miller, and Toby P Breckon. 2018. “Author Profiling: Gender Prediction from Tweets and Images: Notebook for PAN at CLEF 2018.” In *Proceedings of the Ninth International Conference of the CLEF Association (CLEF 2018).* Vol. 2. http://ceur-ws.org/Vol-2125/paper_74.pdf.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. “Long Short-Term Memory.” *Neural Comput.* 9 (8): 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2014. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

Jones, Eric, Travis Oliphant, and Pearu Peterson. n.d. “SciPy: Open Source Scientific Tools for Python.” http://www.scipy.org/.

Khan, Jamal Ahmad. 2017. “Author Profile Prediction Using Trend and Word Frequency Based Analysis in Text: Working Notes of CLEF 2017.” In *CEUR Workshop Proceedings*. Vol. 1866. http://ceur-ws.org/Vol-1866/paper_52.pdf.

Kheng, Guillaume, Léa Laporte, and Michael Granitzer. 2017. “INSA LYON and UNI PASSAU’s Participation at PAN@CLEF’17: Author Profiling Task: Working Notes of CLEF 2017.” In

CEUR Workshop Proceedings. Vol. 1866. http://ceur-ws.org/Vol-1866/paper_61.pdf.

Kocher, Mirco, and Jacques Savoy. 2017. "UniNE at CLEF 2017: Author Profiling Reasoning: Working Notes of CLEF 2017." In *CEUR Workshop Proceedings*. http://ceur-ws.org/Vol-1866/paper_56.pdf.

Kodiyan, Don, Florin Hardegger, Stephan Neuhaus, and Mark Cieliebak. 2017. "Author Profiling with Bidirectional RNNs Using Attention with GRUs: Working Notes of CLEF 2017." In *CEUR Workshop Proceedings*. Vol. 1866. http://ceur-ws.org/Vol-1866/paper_77.pdf.

Koppel, Moshe, Shlomo Argamon, and Anat Rachel Shimoni. 2002. "Automatically Categorizing Written Texts by Author Gender." *Literary and Linguistic Computing* 17 (4): 401–12. http://dx.doi.org/10.1093/llc/17.4.401.

Kosse, Rick, Youri Schuur, and Guido Cnossen. 2018. "Mixing Traditional Methods with Neural Networks for Gender Prediction: Notebook for PAN at CLEF 2018." In *Proceedings of the Ninth International Conference of the CLEF Association (CLEF 2018).* http://ceur-ws.org/Vol-2125/paper_189.pdf.

Li, Guichong. 2016. "Sampling Graphical Networks via Conditional Independence Coupling of Markov Chains." In *Advances in Artificial Intelligence*, edited by Richard Khoury and Christopher Drummond, 298–303. Cham: Springer International Publishing.

López-Monroy, Adrián Pastor, Manuel Montes-y-Gómez, Hugo Jair Escalante, Luis Villaseñor Pineda, and Thamar Solorio. 2017. "Social-Media Users Can Be Profiled by Their Similarity with Other Users: Working Notes of CLEF 2017." In *CEUR Workshop Proceedings*. Vol. 1866. http://ceur-ws.org/Vol-1866/paper_109.pdf.

López-Santillán, Roberto, Luis Carlos González-Gurrola, and Graciela Ramirez Alonso. 2018. "Custom Document Embeddings Via the Centroids Method: Gender Classification in an Author Profiling Task: Notebook for PAN at CLEF 2018." In *CEUR Workshop Proceedings*. Vol. 2125. http://ceur-ws.org/Vol-2125/paper_99.pdf.

Markov, Ilia, Helena Gómez-Adorno, Grigori Sidorov, and Alexander F Gelbukh. 2016. "Adapting Cross-Genre Author Profiling to Language and Corpus: Working Notes of CLEF

2016." In *CEUR Workshop Proceedings*, 947–55. http://ceur-ws.org/Vol-1609/16090947.pdf.

Martinc, Matej, Iza Škrjanec, Katja Zupan, and Senja Pollak. 2017. "PAN 2017: Author Profiling - Gender and Language Variety Prediction: Notebook for PAN at CLEF 2017." In *CLEF 2017 Evaluation Labs and Workshop -- Working Notes Papers, 11-14 September, Dublin, Ireland*, edited by Linda Cappellato, Nicola Ferro, Lorraine Goeuriot, and Thomas Mandl. CEUR-WS.org. http://ceur-ws.org/Vol-1866/.

Martinc, Matej, Blaz Skrlj, and Senja Pollak. 2018. "Multilingual Gender Classification with Multi-View Deep Learning: Notebook for PAN at CLEF 2018." In *Proceedings of the Ninth International Conference of the CLEF Association (CLEF 2018).* http://ceur-ws.org/Vol-2125/paper_156.pdf.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. "Efficient Estimation of Word Representations in Vector Space."

Miura, Yasuhide, Tomoki Taniguchi, Motoki Taniguchi, and Tomoko Ohkuma. 2017. "Author Profiling with Word+Character Neural Attention Network: Notebook for PAN at CLEF 2017." In *CLEF 2017 Evaluation Labs and Workshop -- Working Notes Papers, 11-14 September, Dublin, Ireland*, edited by Linda Cappellato, Nicola Ferro, Lorraine Goeuriot, and Thomas Mandl. CEUR-WS.org. http://ceur-ws.org/Vol-1866/.

Modaresi, Pashutan, Matthias Liebeck, and Stefan Conrad. 2016. "Exploring the Effects of Cross-Genre Machine Learning for Author Profiling in PAN 2016: Working Notes of CLEF 2016." In *CEUR Workshop Proceedings*, 1609:970–77. http://ceur-ws.org/Vol-1609/16090970.pdf.

Mulac, Anthony, Lisa B Studley, and Sheridan Blau. 1990. "The Gender-Linked Language Effect in Primary and Secondary Students' Impromptu Essays." *Sex Roles* 23 (9): 439–70. https://doi.org/10.1007/BF00289762.

Nieuwenhuis, Moniek, and Jeroen Wilkens. 2018. "Twitter Text and Image Gender Classification with a Logistic Regression N-Gram Model: Notebook for PAN at CLEF 2018." In *CEUR Workshop Proceedings*. Vol. 2125. http://ceur-ws.org/Vol-2125/paper_183.pdf.

Nowson, Scott, Julien Perez, Caroline Brun, Shachar Mirkin, and Claude Roux. 2015. “XRCE Personal Language Analytics Engine for Multilingual Author Profiling: Notebook for PAN at CLEF 2015.” In *CLEF 2015 Labs and Workshops, Notebook Papers. CEUR Workshop Proceedings.* Vol. 1391. http://ceur-ws.org/Vol-1391/100-CR.pdf.

Oberlander, Jon, and Scott Nowson. 2006. “Whose Thumb Is It Anyway?: Classifying Author Personality from Weblog Text.” In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, 627–34. COLING-ACL ’06. Stroudsburg, PA, USA: Association for Computational Linguistics. http://dl.acm.org/citation.cfm?id=1273073.1273154.

Orts, Òscar Garibo i. 2018. “A Big Data Approach to Gender Classification in Twitter: Notebook for PAN at CLEF 2018.” In *Proceedings of the Ninth International Conference of the CLEF Association (CLEF 2018).* http://ceur-ws.org/Vol-2125/paper_204.pdf.

Patra, Braja Gopal, Kumar Gourav Das, and Dipankar Das. 2018. “Multimodal Author Profiling for Twitter: Notebook for PAN at CLEF 2018.” In *Conference and Labs of the Evaluation Forum*, 1–10. http://ceur-ws.org/Vol-2125/paper_115.pdf.

Pedregosa, F, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, et al. 2011. “Scikit-Learn: Machine Learning in Python.” *Journal of Machine Learning Research* 12: 2825–30.

Pennebaker, James W. 2013. *The Secret Life of Pronouns: What Our Words Say About Us*. Bloomsbury Press.

Pennebaker, James W., Matthias R Mehl, and Kate G Niederhoffer. 2003. “Psychological Aspects of Natural Language Use: Our Words, Our Selves.” *Annual Review of Psychology* 54 (1): 547–77. https://doi.org/10.1146/annurev.psych.54.101601.145041.

Pennington, Jeffrey, Richard Socher, and Christopher D Manning. 2014. “GloVe: Global Vectors for Word Representation.” In *Empirical Methods in Natural Language Processing (EMNLP)*, 1532–43. http://www.aclweb.org/anthology/D14-1162.

Pereira Junior, Romualdo Alves, and Diana Inkpen. 2017. “Using Cognitive Computing to Get Insights on Personality Traits from Twitter Messages BT - Advances in Artificial

Intelligence: 30th Canadian Conference on Artificial Intelligence, Canadian AI 2017, Edmonton, AB, Canada, May 16-19, 2017, Proceedings." Edited by Malek Mouhoub and Philippe Langlais 10233: 278–83. https://doi.org/10.1007/978-3-319-57351-9_32.

Peters, Matthew, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. "Deep Contextualized Word Representations." *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2227–37. https://doi.org/10.18653/v1/N18-1202.

Potthast, Martin, Tim Gollub, Francisco Rangel, Paolo Rosso, Efstathios Stamatatos, and Benno Stein. 2014. "Improving the Reproducibility of PAN's Shared Tasks: Plagiarism Detection, Author Identification, and Author Profiling." In *Information Access Evaluation Meets Multilinguality, Multimodality, and Visualization. 5th International Conference of the CLEF Initiative (CLEF 14)*, edited by Evangelos Kanoulas, Mihai Lupu, Paul Clough, Mark Sanderson, Mark Hall, Allan Hanbury, and Elaine Toms, 268–99. Berlin Heidelberg New York: Springer. https://doi.org/10.1007/978-3-319-11382-1_22.

Poulston, Adam, Zeerak Waseem, and Mark Stevenson. 2017. "Using TF-IDF n-Gram and Word Embedding Cluster Ensembles for Author Profiling: Working Notes of CLEF 2017." In *CEUR Workshop Proceedings*. Vol. 1866. http://ceur-ws.org/Vol-1866/paper_72.pdf.

Raiyani, Kashyap, Teresa Gonçalves, Paulo Quaresma, and Vítor Beires Nogueira. 2018. "Multi-Language Neural Network Model with Advance Preprocessor for Gender Classification over Social Media: Notebook for PAN at CLEF 2018." In *CEUR Workshop Proceedings*. Vol. 2125. http://ceur-ws.org/Vol-2125/paper_105.pdf.

Rangel, Francisco, Fabio Celli, Paolo Rosso, Martin Potthast, Benno Stein, and Walter Daelemans. 2015. "Overview of the 3rd Author Profiling Task at PAN 2015." In *CLEF 2015 Evaluation Labs and Workshop*, edited by Linda Cappellato, Nicola Ferro, Gareth Jones, and Eric San Juan. CEUR-WS.org.

Rangel, Francisco, Marc Franco-Salvador, and Paolo Rosso. 2017. "A Low Dimensionality Representation for Language Variety Identification." http://arxiv.org/abs/1705.10754.

Rangel, Francisco, Paolo Rosso, Manuel Montes-y-Gómez, Martin Potthast, and Benno Stein. 2018. “Overview of the 6th Author Profiling Task at PAN 2018: Multimodal Gender Identification in Twitter.” In *Working Notes Papers of the CLEF 2018 Evaluation Labs*, edited by Linda Cappellato, Nicola Ferro, Jian-Yun Nie, and Laure Soulier. CEUR Workshop Proceedings. CLEF and CEUR-WS.org.

Rangel, Francisco, Paolo Rosso, Martin Potthast, and Benno Stein. 2017. “Overview of the 5th Author Profiling Task at PAN 2017: Gender and Language Variety Identification in Twitter.” In *Working Notes Papers of the CLEF 2017 Evaluation Labs*, edited by Linda Cappellato, Nicola Ferro, Lorraine Goeuriot, and Thomad Mandl. Vol. 1866. CEUR Workshop Proceedings. CLEF and CEUR-WS.org. http://ceur-ws.org/Vol-1866/.

Razavi, Amir H, Diana Inkpen, Sasha Uritsky, and Stan Matwin. 2010. “Offensive Language Detection Using Multi-Level Classification.” In *Proceedings of the 23rd Canadian Conference on Advances in Artificial Intelligence*, 16–27. AI’10. Berlin, Heidelberg: Springer-Verlag. https://doi.org/10.1007/978-3-642-13059-5_5.

Richards, Christina, Walter Pierre Bouman, Leighton Seal, Meg John Barker, Timo O Nieder, and Guy T’Sjoen. 2016. “Non-Binary or Genderqueer Genders.” *International Review of Psychiatry* 28 (1): 95–102. https://doi.org/10.3109/09540261.2015.1106446.

Roller, Stephen, Michael Speriosu, Sarat Rallapalli, Benjamin Wing, and Jason Baldridge. 2012. “Supervised Text-Based Geolocation Using Language Models on an Adaptive Grid.” In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 1500–1510. EMNLP-CoNLL ’12. Stroudsburg, PA, USA: Association for Computational Linguistics. http://www.aclweb.org/anthology/D12-1137.

Rosen, Aliza. 2017. “Tweeting Made Easier.” The Twitter Blog. 2017. https://blog.twitter.com/official/en_us/topics/product/2017/tweetingmadeeasier.html.

Rout, Dominic, Kalina Bontcheva, Daniel Preo\ctiuc-Pietro, and Trevor Cohn. 2013. “Where’s @Wally?: A Classification Approach to Geolocating Users Based on Their Social Ties.” In *Proceedings of the 24th ACM Conference on Hypertext and Social Media*, 11–20. HT ’13. New York, NY, USA: ACM. https://doi.org/10.1145/2481492.2481494.

Schaetti, Nils. 2017. “UniNE at CLEF 2017: TF-IDF and Deep-Learning for Author Profiling: Working Notes of CLEF 2017.” In *CEUR Workshop Proceedings*. Vol. 1866. http://ceur-ws.org/Vol-1866/paper_80.pdf.

———. 2018. “Character-Based Convolutional Neural Network and ResNet18 for Twitter Author Profiling: Notebook for PAN at CLEF 2018.” In *Proceedings of the Ninth International Conference of the CLEF Association (CLEF 2018).* http://ceur-ws.org/Vol-2125/paper_100.pdf.

Schler, Jonathan, Moshe Koppel, Shlomo Argamon, and James W. Pennebaker. 2006. “Effects of Age and Gender on Blogging.” *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs* 6: 199–205.

Sezerer, Erhan, Ozan Polatbilek, Özge Sevgili, and Selma Tekir. 2018. “Gender Prediction From Tweets With Convolutional Neural Networks: Notebook for PAN at CLEF 2018.” In *Proceedings of the Ninth International Conference of the CLEF Association (CLEF 2018).* http://ceur-ws.org/Vol-2125/paper_116.pdf.

Shaban, Hamza. 2019. “Twitter Reveals Its Daily Active User Numbers for the First Time.” *The Washington Post*, February 7, 2019. https://www.washingtonpost.com/technology/2019/02/07/twitter-reveals-its-daily-active-user-numbers-first-time/.

Sierra, Sebastián, Manuel Montes-y-Gómez, Thamar Solorio, and Fabio A. González. 2017. “Convolutional Neural Networks for Author Profiling in PAN 2017: Working Notes of CLEF 2017.” In *CEUR Workshop Proceedings*. Vol. 1866. http://ceur-ws.org/Vol-1866/paper_93.pdf.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting.” *J. Mach. Learn. Res.* 15 (1): 1929–58. http://dl.acm.org/citation.cfm?id=2627435.2670313.

Stamatatos, Efstathios, Francisco Rangel, Michael Tschuggnall, Mike Kestemont, Paolo Rosso, Benno Stein, and Martin Potthast. 2018. “Overview of PAN-2018: Author Identification, Author Profiling, and Author Obfuscation.” In *Experimental IR Meets Multilinguality,*

Multimodality, and Interaction. 9th International Conference of the CLEF Initiative (CLEF 18), edited by Patrice Bellot, Chiraz Trabelsi, Josiane Mothe, Fionn Murtagh, Jian Yun Nie, Laure Soulier, Eric Sanjuan, Linda Cappellato, and Nicola Ferro. Berlin Heidelberg New York: Springer.

Statistics Canada. 2017. "Census Profile. 2016 Census." Ottawa. https://www12.statcan.gc.ca/census-recensement/2016/dp-pd/prof/details/page.cfm?Lang=E&Geo1=PR&Code1=01&Geo2=&Code2=&Data=Count&SearchText=Canada&SearchType=Begins&SearchPR=01&B1=All&TABID=1.

Stout, Luka, Robert Musters, and Chris Pool. 2018. "Author Profiling Based on Text and Images: Notebook for PAN at CLEF 2018." In *Proceedings of the Ninth International Conference of the CLEF Association (CLEF 2018).* http://ceur-ws.org/Vol-2125/paper_79.pdf.

Takahashi, Takumi, Takuji Tahara, Koki Nagatani, Yasuhide Miura, Tomoki Taniguchi, and Tomoko Ohkuma. 2018. "Text and Image Synergy with Feature Cross Technique for Gender Identification." In *Working Notes Papers of the CLEF*.

Tellez, Eric Sadit, Sabino Miranda-Jiménez, Mario Graff, and Daniela Moctezuma. 2017. "Gender and Language-Variety Identification with MicroTC: Working Notes of CLEF 2017." In *CEUR Workshop Proceedings*. Vol. 1866. http://ceur-ws.org/Vol-1866/paper_104.pdf.

Tellez, Eric Sadit, Sabino Miranda-Jiménez, Daniela Moctezuma, Mario Graff, Vladimir Salgado, and José Ortiz-Bejar. 2018. "Gender Identification through Multi-Modal Tweet Analysis Using MicroTC and Bag of Visual Words: Notebook for PAN at CLEF 2018." In *Working Notes of {CLEF} 2018 - Conference and Labs of the Evaluation Forum, Avignon, France, September 10-14, 2018.* http://ceur-ws.org/Vol-2125/paper_164.pdf.

Ucelay, Maria José Garciarena, Maria Paula Villegas, Dario G Funez, Leticia C Cagnina, Marcelo Luis Errecalde, Gabriela Ramírez-de-la-Rosa, and Esaú Villatoro-Tello. 2016. "Profile-Based Approach for Age and Gender Identification: Working Notes of CLEF 2016." In *CEUR Workshop Proceedings*, 1609:864–73. http://ceur-ws.org/Vol-1609/16090864.pdf.

Veenhoven, Robert, Stan Snijders, Daniël van der Hall, and Rik van Noord. 2018. "Using Translated Data to Improve Deep Learning Author Profiling Models: Notebook for PAN at

CLEF 2018." In *Proceedings of the Ninth International Conference of the CLEF Association (CLEF 2018).* http://ceur-ws.org/Vol-2125/paper_178.pdf.

Weren, Edson Roberto Duarte. 2015. "Information Retrieval Features for Personality Traits: Notebook for PAN at CLEF 2015." In *CLEF 2015 Labs and Workshops, Notebook Papers. CEUR Workshop Proceedings.* Vol. 1391. http://ceur-ws.org/Vol-1391/169-CR.pdf.

Werlen, Lesly Miculicich. 2015. "Statistical Learning Methods for Profiling Analysis: Notebook for PAN at CLEF 2015." In *CLEF 2015 Labs and Workshops, Notebook Papers. CEUR Workshop Proceedings.* http://ceur-ws.org/Vol-1391/16-CR.pdf.

Wiemer-Hastings, Peter. 2004. "Latent Semantic Analysis." In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, 1–14.

Appendix A. Glossary of Twitter Terms[25]

- **Tweet (noun):** A tweet may consist of text, photos, GIFs, and videos.
- **Tweet (verb):** The act of sending a tweet.
- **Follower:** A follower is another Twitter account that has followed a Twitter user to receive the user's tweets in their Home timeline.
- **Home timeline:** The Home timeline of a Twitter user displays a stream of tweets from the accounts which the user follows.
- **Retweet (noun):** A tweet that a user forwards to their followers, to share an interesting post.
- **Retweet (verb):** The act of sharing another account's tweet with one's followers.
- **Protected tweets:** By default, all tweets are public. However, a user can choose to *protect* their tweets. Doing so would make the tweets of the said users visible only to their followers.
- **Profile:** The profile of a Twitter user shows information such as bio and profile photo, as well as any tweets that the user has posted.
- **Profile photo:** A personal image picked by a user that appears on their profile and next to each of their tweets.
- **Bio:** A short personal description, with a maximum length of 160 characters, which appears in a Twitter user's profile.
- **Mention:** The act of mentioning another account in a tweet, by including *@username* in the tweet, where *username* is the identifier of the said account.
- **Hashtag:** Any word or phrase immediately preceded by the # symbol. Hashtags are used to categorize tweets, and enable browsing the tweets containing the same keyword or topic.
- **Direct Message (DM):** A Private, one-on-one message sent from one Twitter account to another. These messages are not shared with anyone else except the two accounts involved.

[25] https://help.twitter.com/en/glossary

Appendix B. Additional Details of the ASI Dataset

Table 27. Distribution of users based on municipality (complete list)

Municipality	Users	% Users
Toronto	35,506	17.6%
Vancouver	26,466	13.1%
Montréal	18,981	9.4%
Ottawa & Gatineau (Ontario part)	14,863	7.4%
Calgary	11,816	5.9%
Edmonton	10,788	5.3%
Halifax	6,693	3.3%
Hamilton	5,398	2.7%
Winnipeg	4,933	2.4%
Kitchener, Cambridge & Waterloo	4,841	2.4%
Victoria	4,036	2.0%
London	3,509	1.7%
Québec	2,659	1.3%
Saskatoon	2,478	1.2%
Regina	2,235	1.1%
St. Catharines & Niagara	2,191	1.1%
New Glasgow	2,114	1.0%
Kelowna	2,057	1.0%
Guelph	1,867	0.9%
Kingston	1,852	0.9%
Windsor	1,837	0.9%
St. John's	1,761	0.9%
Barrie	1,603	0.8%
Oshawa	1,527	0.8%
Peterborough	1,402	0.7%
Fredericton	1,182	0.6%
Saint John	1,113	0.6%
Lethbridge	1,059	0.5%
Red Deer	992	0.5%
Moncton	976	0.5%
Charlottetown	862	0.4%
Abbotsford & Mission	773	0.4%
Ottawa & Gatineau (Quebec part)	768	0.4%
Greater Sudbury	763	0.4%

Cape Breton	757	0.4%
Brantford	736	0.4%
Nanaimo	722	0.4%
Medicine Hat	712	0.4%
Thunder Bay	653	0.3%
Kamloops	618	0.3%
Belleville	565	0.3%
Sarnia	511	0.3%
Brandon	491	0.2%
Chatham-Kent	480	0.2%
Wood Buffalo	468	0.2%
Chilliwack	418	0.2%
Grande Prairie	397	0.2%
Vernon	361	0.2%
North Bay	361	0.2%
Campbell River	350	0.2%
Penticton	345	0.2%
Trois-Rivières	313	0.2%
Norfolk	297	0.1%
Yellowknife	292	0.1%
Sherbrooke	283	0.1%
Truro	278	0.1%
Orillia	278	0.1%
Whitehorse	276	0.1%
Courtenay	263	0.1%
Saguenay	260	0.1%
Prince George	255	0.1%
Squamish	240	0.1%
Sault Ste. Marie	237	0.1%
Kawartha Lakes	229	0.1%
Brockville	224	0.1%
Moose Jaw	216	0.1%
Canmore	209	0.1%
Corner Brook	208	0.1%
Cornwall	207	0.1%
Woodstock	198	0.1%
Owen Sound	196	0.1%
Okotoks	181	0.1%
Stratford	180	0.1%

Prince Albert	167	0.1%
Rimouski	165	0.1%
Cobourg	162	0.1%
Collingwood	158	0.1%
Timmins	154	0.1%
Leamington	141	0.1%
Summerside	141	0.1%
Centre Wellington	136	0.1%
Midland	134	0.1%
Yorkton	132	0.1%
Fort St. John	127	0.1%
Duncan	123	0.1%
Swift Current	119	0.1%
Lloydminster (Alberta part)	117	0.1%
Parksville	111	0.1%
Kentville	110	0.1%
Miramichi	109	0.1%
Camrose	107	0.1%
Drummondville	104	0.1%
Estevan	100	0.0%
Kenora	98	0.0%
Saint-Jean-sur-Richelieu	94	0.0%
Port Hope	90	0.0%
Pembroke	90	0.0%
Sylvan Lake	88	0.0%
Cranbrook	85	0.0%
Lacombe	79	0.0%
North Battleford	76	0.0%
Brooks	75	0.0%
Saint-Hyacinthe	75	0.0%
Powell River	73	0.0%
Port Alberni	72	0.0%
Tillsonburg	72	0.0%
Victoriaville	72	0.0%
Salmon Arm	72	0.0%
Steinbach	69	0.0%
Cold Lake	68	0.0%
Sorel-Tracy	68	0.0%
High River	67	0.0%

Granby	66	0.0%
Thompson	65	0.0%
Terrace	64	0.0%
Rouyn-Noranda	63	0.0%
Quesnel	62	0.0%
Strathmore	60	0.0%
Bathurst	58	0.0%
Ingersoll	57	0.0%
Shawinigan	55	0.0%
Joliette	55	0.0%
Petawawa	54	0.0%
Wetaskiwin	54	0.0%
Dawson Creek	53	0.0%
Edmundston	51	0.0%
Portage la Prairie	50	0.0%
Grand Falls-Windsor	48	0.0%
Baie-Comeau	45	0.0%
Williams Lake	42	0.0%
Temiskaming Shores	41	0.0%
Rivière-du-Loup	39	0.0%
Amos	38	0.0%
Prince Rupert	37	0.0%
Val-d'Or	36	0.0%
Thetford Mines	36	0.0%
Salaberry-de-Valleyfield	35	0.0%
Matane	34	0.0%
Lloydminster (Saskatchewan part)	33	0.0%
Bay Roberts	30	0.0%
Sept-Iles	27	0.0%
Campbellton (New Brunswick part)	24	0.0%
Alma	22	0.0%
Saint-Georges	18	0.0%
Lachute	14	0.0%
Elliot Lake	13	0.0%
Dolbeau-Mistassini	10	0.0%
Cowansville	9	0.0%
Hawkesbury (Ontario part)	6	0.0%
Campbellton (Quebec part)	4	0.0%
Grand Total	**201,774**	**100.0%**

Probability Distribution of Age Groups

Table 28. Probability distribution of age group [0, 25]

Probability Range	Users	% Users
0–0.05	31,302	12.57%
0.05–0.1	21,142	8.49%
0.1–0.15	15,868	6.37%
0.15–0.2	11,163	4.48%
0.2–0.25	9,284	3.73%
0.25–0.3	7,923	3.18%
0.3–0.35	97,615	39.21%
0.35–0.4	5,933	2.38%
0.4–0.45	5,674	2.28%
0.45–0.5	3,921	1.58%
0.5–0.55	4,555	1.83%
0.55–0.6	5,566	2.24%
0.6–0.65	4,221	1.70%
0.65–0.7	4,335	1.74%
0.7–0.75	3,680	1.48%
0.75–0.8	2,967	1.19%
0.8–0.85	3,033	1.22%
0.85–0.9	3,100	1.25%
0.9–0.95	2,415	0.97%
0.95–1	5,235	2.10%
Grand Total	**248,932**	**100.00%**

Table 29. Probability distribution of age group [25, 34]

Probability Range	Users	% Users
0–0.05	33,679	13.53%
0.05–0.1	28,463	11.43%
0.1–0.15	111,459	44.77%
0.15–0.2	21,754	8.74%
0.2–0.25	16,250	6.53%
0.25–0.3	12,094	4.86%
0.3–0.35	9,408	3.78%
0.35–0.4	5,457	2.19%
0.4–0.45	2,927	1.18%
0.45–0.5	2,409	0.97%
0.5–0.55	950	0.38%
0.55–0.6	505	0.20%
0.6–0.65	274	0.11%
0.65–0.7	156	0.06%
0.7–0.75	33	0.01%
0.75–0.8	73	0.03%
0.8–0.85	21	0.01%
0.85–0.9	12	0.00%
0.9–0.95	2	0.00%
0.95–1	3,006	1.21%
Grand Total	**248,932**	**100.00%**

Table 30. Probability distribution of age group [35, 44]

Probability Range	Users	% Users
0–0.05	38,672	15.54%
0.05–0.1	31,680	12.73%
0.1–0.15	122,297	49.13%
0.15–0.2	19,930	8.01%
0.2–0.25	15,792	6.34%
0.25–0.3	9,182	3.69%
0.3–0.35	3,331	1.34%
0.35–0.4	3,448	1.39%
0.4–0.45	2,353	0.95%
0.45–0.5	1,287	0.52%
0.5–0.55	267	0.11%
0.55–0.6	112	0.04%
0.6–0.65	57	0.02%
0.65–0.7	162	0.07%
0.7–0.75	59	0.02%
0.75–0.8	34	0.01%
0.8–0.85	15	0.01%
0.85–0.9	2	0.00%
0.9–0.95	11	0.00%
0.95–1	241	0.10%
Grand Total	**248,932**	**100.00%**

Table 31. Probability distribution of age group [45, 54]

Probability Range	Users	% Users
0–0.05	53,614	21.54%
0.05–0.1	24,527	9.85%
0.1–0.15	115,472	46.39%
0.15–0.2	24,162	9.71%
0.2–0.25	12,179	4.89%
0.25–0.3	7,837	3.15%
0.3–0.35	5,317	2.14%
0.35–0.4	2,894	1.16%
0.4–0.45	1,382	0.56%
0.45–0.5	982	0.39%
0.5–0.55	322	0.13%
0.55–0.6	83	0.03%
0.6–0.65	22	0.01%
0.65–0.7	17	0.01%
0.7–0.75	22	0.01%
0.75–0.8	10	0.00%
0.8–0.85	3	0.00%
0.85–0.9	2	0.00%
0.9–0.95	0	0.00%
0.95–1	85	0.03%
Grand Total	**248,932**	**100.00%**

Table 32. Probability distribution of age group [55, 64]

Probability Range	Users	% Users
0–0.05	69,312	27.84%
0.05–0.1	15,780	6.34%
0.1–0.15	102,774	41.29%
0.15–0.2	11,485	4.61%
0.2–0.25	18,985	7.63%
0.25–0.3	9,476	3.81%
0.3–0.35	5,990	2.41%
0.35–0.4	6,619	2.66%
0.4–0.45	3,674	1.48%
0.45–0.5	2,102	0.84%
0.5–0.55	1,329	0.53%
0.55–0.6	575	0.23%
0.6–0.65	136	0.05%
0.65–0.7	186	0.07%
0.7–0.75	184	0.07%
0.75–0.8	64	0.03%
0.8–0.85	56	0.02%
0.85–0.9	46	0.02%
0.9–0.95	24	0.01%
0.95–1	135	0.05%
Grand Total	**248,932**	**100.00%**

Table 33. Probability distribution of age group [65, …]

Probability Range	Users	% Users
0–0.05	80,638	32.39%
0.05–0.1	19,122	7.68%
0.1–0.15	100,925	40.54%
0.15–0.2	6,752	2.71%
0.2–0.25	7,661	3.08%
0.25–0.3	5,993	2.41%
0.3–0.35	7,043	2.83%
0.35–0.4	4,997	2.01%
0.4–0.45	3,798	1.53%
0.45–0.5	2,604	1.05%
0.5–0.55	2,962	1.19%
0.55–0.6	1,835	0.74%
0.6–0.65	1,082	0.43%
0.65–0.7	817	0.33%
0.7–0.75	680	0.27%
0.75–0.8	500	0.20%
0.8–0.85	259	0.10%
0.85–0.9	44	0.02%
0.9–0.95	33	0.01%
0.95–1	1,187	0.48%
Grand Total	**248,932**	**100.00%**

Appendix C. Comparison of the Bidirectional LSTM Models

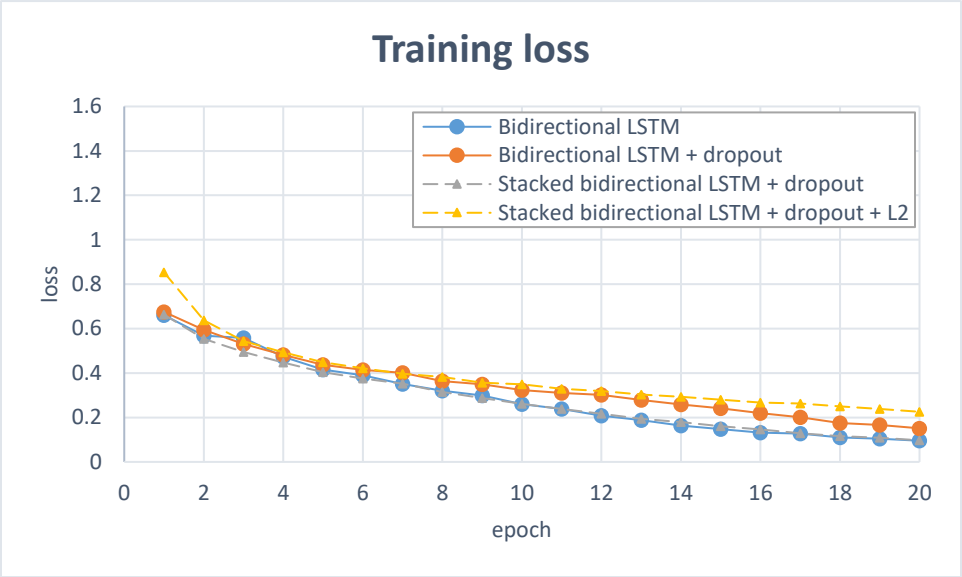

Figure 29. Training loss for the bidirectional LSTM models

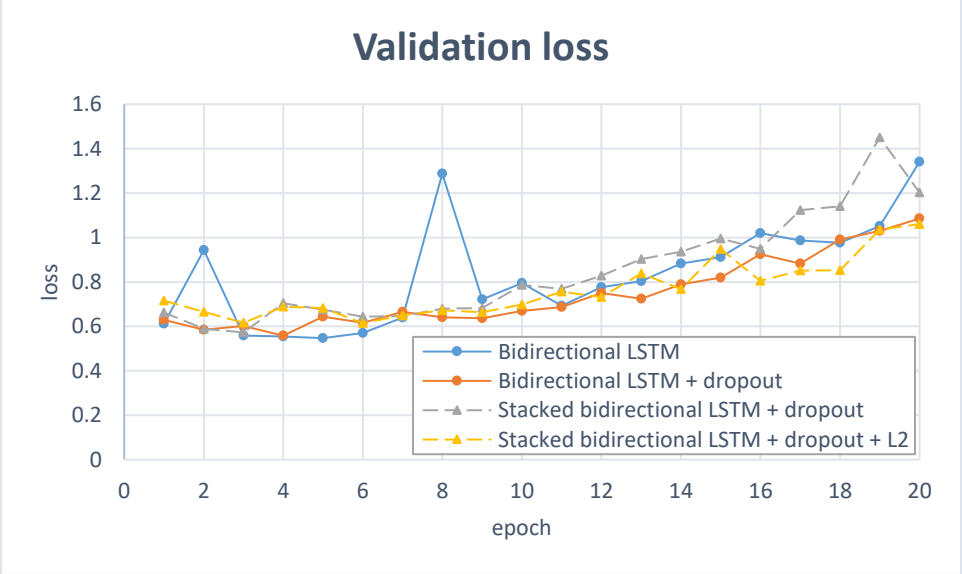

Figure 30. Validation loss for the bidirectional LSTM models

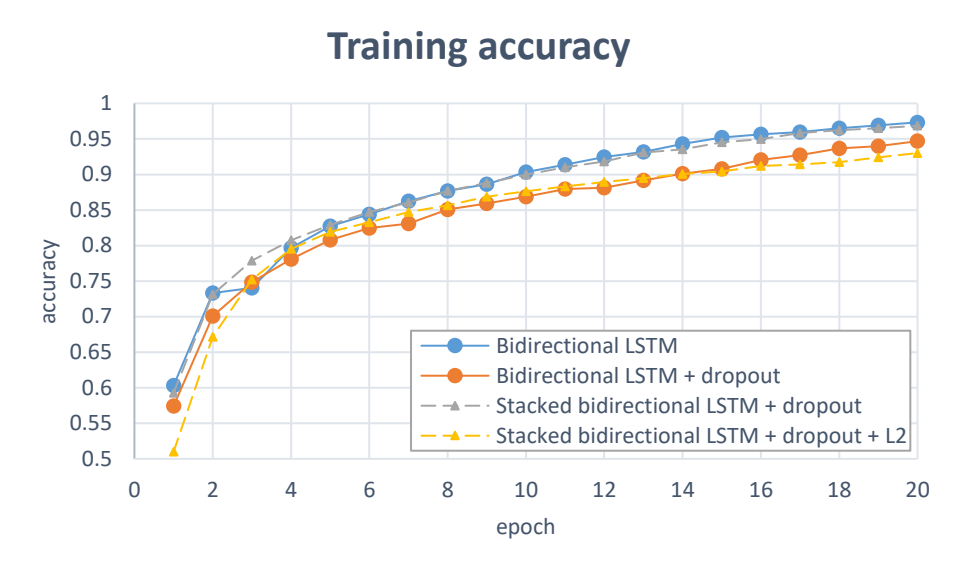

Figure 31. Training accuracy for the bidirectional LSTM models

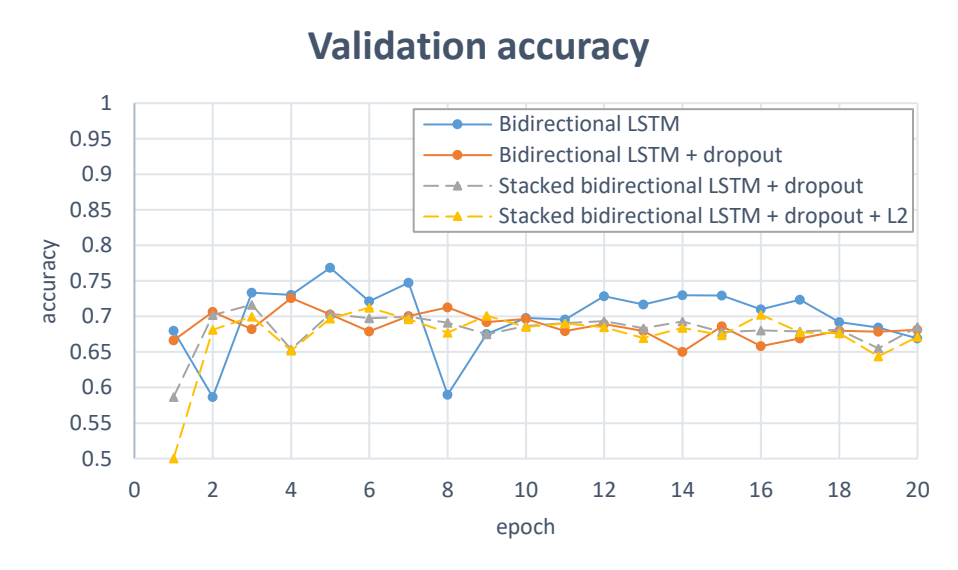

Figure 32. Validation accuracy for the bidirectional LSTM models