LIN3011/LIN3012: Data-Driven NLP Final Project

Deborah Vella
University of Malta
deborah.vella.17@um.edu.mt

1. ABSTRACT

This paper deals with the task of gender classification from blog posts. To cater for this, The Blog Authorship Corpus is used, by decreasing it to a smaller dataset of 1500 examples. The paper will discuss various algorithms and feature sets which can be used to address this NLP problem. It will proceed by describing the methodology of the implementation and last but not least, evaluating the results of five different classifiers trained and tested on four different feature sets.

2. INTRODUCTION

The technological advancements has brought with it various tasks which a lot of researchers try to solve, in different domains. In NLP one of the tasks that has been researched about in previous years is the classification of gender from text. This interest has risen due to the success of social media such as twitter, blogs, and text available online. Different researchers have dealt with this problem through different methods and classifiers. Usually, some features are decided upon to eventually be extracted from the corpora available, and then one or more classifiers are used to train on these extracted feature sets. The main features which were used in various previous studies are part-of-speech, word-level, syntactic, and character-level features. The algorithms used can vary a lot, having some using deep learning approaches, others decision tree based approaches (such as random forest) and more. This study will discuss some of the algorithms used in different research papers, which are then implemented and evaluated. The methodology will also be discussed, explaining each decision made and what inspired it. The corpus is an annotated blog corpus, which is very useful as there are no topic restrictions for the bloggers, hence, it is very rich in variety.

3. BACKGROUND RESEARCH

3.1 Naive Bayes Classifier

A Naive Bayes classifier, predicts the most likely class of the provided input based on its feature vector. The algorithm has given good results in various domains and applications such as text classification and medical diagnosis. This classifier makes the assumption that given a class, the features are independent of each other, meaning that no feature is affected by any other feature. This is illustrated by the following equation:

$$P(X|C) = \prod_{i=1}^{n} P(X_i|C),$$

[1] where $X = (X_1, ..., X_n)$ and C is a class. This classifier also makes use of a discriminant function $f_i^*(x) = P(C = i|X = x)$, which after applying Bayes theorem becomes $P(C = i|X = x) = \frac{P(X=x|C=i)P(C=i)}{P(X=x)}$. Having the knowledge that $P(X = x)$ is constant, therefore, the same for all classes, it can be ignored, which results in $f_i^*(x) = P(X = x|C = i)$, having $P(X = x|C = i)$ being the *class-conditional probability distribution.* Finally, the Bayes classifier finds the class which is the highest out of all classes, given an example x.

$$h^*(x) = argmax_i P(X = x|C = i)P(C = i)$$

By the assumption of independent features, the final Naive Bayes formula is described by

$$f_i^N B(x) = argmax_i P(C = i) \prod_{j=1}^{n} P(X_j = x_j|C = i)$$

The aforementioned independence assumption can in reality be a rather unrealistic assumption, hence, the Naive Bayes classifier can result in inaccurate probabilities, however, it still manages to classify most of the examples as they should. It was concluded that Naive Bayes works best when it is applied on totally independent features or on functionally dependent features [1].

3.2 Logistic Regression

Logistic regression can be used for classification problems as well. Its aim is to learn directly a scoring function f(x) and maximising the likelihood $P(y|x)$. It gets the posterior probability $\mathbb{P}(y|x)$ by avoiding to learn the conditional distributions $p(x|y)$ and the prior $\mathbb{P}(y)$.

$$score(x) = \log\left(\frac{\mathbb{P}(y = 1|x)}{1 - \mathbb{P}(y = 0|x)}\right) = f(x)$$

The model is defined by

$$f(x) = \mathbf{w} \ . \ \mathbf{x} + b$$

meaning that, it tries to compute the probability that example x is part of a class, given weights $\mathbf{w}$, and a bias b. This is followed by a decision rule which classifies example $\mathbf{x}$ by $\hat{y}$ which can be either 0 or 1.

$$assign\ \mathbf{x}\ to\ \hat{y} = \begin{cases} 1 \ \ if\ f(\mathbf{x}) \geq 0 \\ 0 \ \ if\ f(\mathbf{x}) < 0 \end{cases}$$

Since the output is required to be a probability, an increasing monotone function $\mathbb{R} \rightarrow [0, 1]$: sigmoid is used. The

sigmoid function transforms any input number into a probability between zero and one.

$$\mathbb{P}(y = 1|\mathbf{x}) = \frac{1}{1 + exp^{f(x)}}$$

Therefore, provided that logistic regression is being used as a classification technique, a threshold has to be applied, having anything above it classified as one class and anything below it is classified as another class. Therefore, the decision changes as follows:

$$assign\ \mathbf{x}\ to\ \hat{y} = \begin{cases} 1 & if\ \mathbb{P}(y = 1|\mathbf{x}) \geq 0.5 \\ 0 & if\ \mathbb{P}(y = 1|\mathbf{x}) < 0.5 \end{cases}$$

Furthermore, logistic regression makes use of the binary cross entropy loss function, $l(y, p)$, to give an indication of how far off the result is from the target output, to eventually be able to optimize the classifier accordingly.

$$l(y_i, p_i) = -y_i \log p_i - (1 - y_i) \log(1 - p_i)$$

having $p_i = \mathbb{P}(y_i = 1|\mathbf{x}_i)$. For optimization, a gradient descent algorithm is usually used, denoted by $h = -\bigtriangledown J(\theta)$, where θ represents the two parameters, weights and bias [2].

3.3 Random Forest

Random forest is usually a suitable algorithm to use for text classification, given that it performs well on high dimensional data, while maintaining a simplistic algorithm. It can also be used to deal with regression problems. Random forest is a collection of decision trees, each used to perform classifications. Every ensemble is created from a bootstrap sample obtained from the training data. The nodes are split based on the best split of a random subset of the features rather than, split based on the best split of all features. The bias usually increase as a result of the randomness of the algorithm's nature. This is compensated by decreasing the variances with averaging [3]. The basic algorithm of random forest is as follows [4]:

1. Get n_{tree} bootstrap samples from the training data.
2. For each bootstrap sample, create an unpruned classification tree, splitting each node by choosing the best split among the variables of the random sample m_{try}.
3. Perform the predictions on the new examples by aggregating the classifications outputted by the n_{tree} trees. Hence, choosing the most popular predicted class from the trees.

3.4 SVM

Support Vector Machine (SVM) was firstly presented in COLT-92 by Boser, Guyon and Vapnik [5]. The idea of SVM is to maximise the predictive accuracy, but avoiding overfitting at the same time, the result being either a classification technique or a regression technique. The goal is to search for the optimal hyperplane which separates the data into different groups, each conforming with a particular class. The optimal hyperplane, has the maximum margin, meaning that the hyperplane has the maximum distance between itself and the data points of the classes [6]. The maximum margin equation is illustrated below [5]:

$$margin = argmind_{x \epsilon D} = argmin_{x \epsilon D} \frac{|\mathbf{x}.\mathbf{w} + b|}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

where $\mathbf{x}.\mathbf{w} + b = 0$ is the definition of the hyperplane. The search for optimal hyperplane is highly dependent on the support vectors, which are the nearest data points to the hyperplane. Mathematically this means that

- $If\ Y_i = +1;\ wxi + b \geq 1$
- $If\ Y_i = -1;\ wxi + b \leq 1$
- $For\ all\ i;\ y_i(w_i + b) \geq 1$ [5]

having x being an input vector, w is a weight vector, b is the bias and y_i is the shortest distance to the nearest support vector. Figure 1 visually illustrates these formulas.

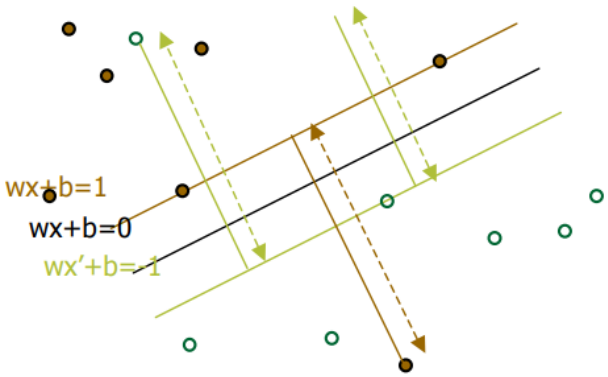

Figure 1: This is a figure of the hyperplane, and its margin, together with the closest planes which lie on the closest support vectors [5]

Furthermore, SVMs cater for non-linearity, which means that the data points are not linearly separable, hence, no straight line can separate the data points into their own classes. To deal with non-linearity, SVMs make use of the **kernel trick** which essentially is the mapping of data points from one dimension to a higher dimensional space. In addition to this, feature spaces are to be used as they easily show similarity and hence, pattern recognition can be done. An example of the transition from linearly inseparable to linearly separable is shown in Figure 2.

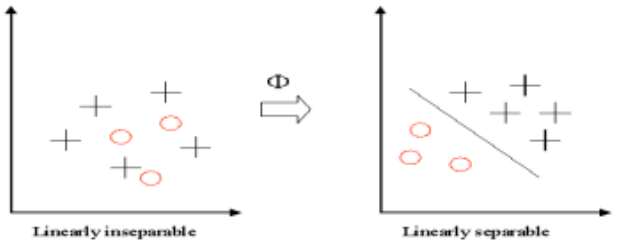

Figure 2: The left-hand side show linearly inseparable data, while the right-hand side shows the same data which is now linearly separable after applying the kernel trick [5]

Some examples of kernels are as follows:

- Polynomial kernel: $K(x, x') = \langle x, x' \rangle^d$ or $K(x, x') = (\langle x, x' \rangle + 1)^d$

- Gaussian Radial Basis Function:
  $K(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$
- Exponential Radial Basis Function:
  $K(x, x') = \exp\left(-\frac{\|x-x'\|}{2\sigma^2}\right)$ [5]

A support vector machine may sometimes perform even better than Neural Networks, but may be harder to implement and use. As can be seen, SVMs make use of supervised learning to eventually be able to classify new instances to their correct class.

3.5 Neural Networks

Neural Networks are learning algorithms which are inspired by the biological functioning of the brain. Neural Networks are usually made up of five main components:

1. A directed graph which is the network, connected with links.
2. A state variable associated with each node.
3. A real-valued weight for each link.
4. A real-valued bias associated with each node.
5. An activation function, to be able to deal with linearly inseparable data, example ReLu, sigmoid and tanh.

The inputs are the nodes which do not have any links going into them and the output node is the one at the end which has no links going out of it. The rest of the nodes are the hidden nodes which make up the hidden layers. The values propagate through the network starting from the input and ending at the output. A feed-forward network calculates the outputs given a set of inputs as just discussed. Furthermore, a layered feed-forward NN, is one which traverses the same number of links for any path chosen from input to output. Neural networks also use a loss function and some type of optimizer such as gradient descent, Adam optimizer and backpropagation. Layered feed-forward neural networks became popular throughout the years, one main reason being that they can generalise well and are able to classify new unseen data providing competitive results [7].

4. LITERATURE REVIEW

In 2006, Jonathan Schler et al. [8] created their own corpus from scratch by reviewing all accessible blogs from blogger.com, and downloaded all the blogs from which the author gender could be deduced. In all 71,000 blogs were collected each having at least 200 appearances of common English words. They proceeded by extracting feature vectors from the corpus, in this case they used stylistic features (function words, parts-of-speech, blog words and hyperlinks). The learning algorithm used is Multi-Class Real Window (MCRW). From the results it was concluded that each gender has more common topics they write about for example females tend to write more about their personal lives while male bloggers tend to write more about politics.

Later on in 2010, Cathy Zhang and Pengyu Zhang [9] tackled the gender identification problem from text, by using various features and algorithms. The features used are words (either a binary representation or a frequency representation), average word or sentence length, part of speech tags and word factor analysis. The classifiers used are Naive Bayes, SVM and linear discriminant analysis (LDA). These were trained on a blog corpus.

The following year, in 2011, Na Cheng et al. [10] made use of three different algorithms to address the gender classification problem. These algorithms are Bayesian logistic regression, AdaBoost decision tree and support vector machine. The algorithms were trained and evaluated on large corpora including Reuters Corpus Volume 1 data set a and Enron e-mail data set. Various feature types were used which are character-based features, word-based features, syntactic features, structural features and function words. The results obtained from their experiments show that the features that best distinguish between male and female are function words, word-based features and structural features. The Reuters data was pre-processed by discarding unnecessary information for example date and time, and only kept those sentences whose length lies between 200 and 1000 words. The Enron data was pre-processed by removing headers, any signatures and reply texts. For this corpus, only the emails whose word length lie between 50 words and 1000 words were kept.

In the same year, Vasiliki Simaki et al. [11] also approached the gender identification problem, this time researching the question of whether men and women use different unigrams. The corpus used is the 'Blog author gender classification data set' which is a gender-annotated corpus. Fifty seven different unigrams were examined, which are all the alphabetic letters, both upper case and lower case together with special characters and punctuation. From this study, it was concluded that twenty-five unigrams are statistically significant, which implies that gender classification can also be done on character-level features.

In 2015, Aric Bartle and Jim Zheng [12] took a deep learning approach towards the gender classification task. The corpora used were blogs and books written in the 19th/20th century. A bag of words was used alongside an SVM to generate a feature vector. Furthermore, average embedding was used, which essentially is the average of the word vectors in the document. A paragraph2vec model was also used and an SVM was implemented to classify the resultant documents. Last but not least POS features were used on the blog corpus. In this paper, they refined the RCNN deep learning approach, and eventually created a new model called Windowed Recurrent Convolutional Neural Network (WRCNN).

In 2017, Tolkachev Alexey [13], performed gender prediction on the Blog Authorship Corpus [8]. Six different classifiers were used to train and test the features with, which are: K-Nearest Neighbor (K-NN), Multinomial Bayes, Logistic Regression, Svm, Random Forest and last but not least, Gradient Boosting Classifier. The feature set used was TF-IDF, having four-grams on each post. The classifiers were trained and tested on character-based features (example number of characters, number of upper-cases, number of white spaces etc.), syntactic features (number of single quotes, commas, periods, colons etc.) and word-based features (example number of words, vocabulary richness, num-

ber of short words etc.).

Vijay Prakash Dwivedi et al. [14] presented two different ways in approaching gender classification. The first approach is manual features extraction including two feature classes (character length sequence patterns and 13 new word classes). The second approach makes use of Bidirectional Long Short Term Memory Networks (BLSTMs). Two corpora containing blog data were used, one which has already been well used by other researchers, and another one which filtered from the Blog Authorship data set and is twenty times larger than the first. The BLSTM performed better on the larger corpus than on the smaller one.

5. METHODOLOGY

5.1 Corpus

The Blog Authorship Corpus is used for this study, but was downloaded as a csv file from kaggle[1]. The blog contains around 681,288 posts collected from 19,320 bloggers, obtained from blogger.com. In all it contains over 140 million words. The csv file contains a column for each different attribute: id, gender, age, topic, sign, dateand text (the blogs). The only data which is needed for this project are the gender column and the text column. It should be noted that only 1500 examples were used from the whole data set.

5.2 Data Preprocessing

Each blog was altered by keeping words which have frequency between 3 and 6000. This way words which may be deemed as not very useful for the classifier are removed. These are usually words which rarely occur, hence have a frequency of 1 or 2, or else they are words that occur way too frequent such as the word "I". Furthermore, numerical words are also removed as they usually do not help the classifiers. The TFIDF and Count vectorizers used from Python's sklearn library remove any stop words and punctuation and convert all words to lower case, as the same word with different cases will be assumed to be a different word.

5.3 Algorithms used

Five different classification algorithms are implemented, which are all trained and tested on the same data set. The train and test labels are changed into binary labels beforehand applying 0 for male and 1 for female. Their results will be discussed later in the paper.

- Naive Bayes: A Multinomial Naive Bayes classifier is implemented by using sklearn python library.
- Logistic Regression: The Logistic regression is also implemented by using sklearn.
- Random Forest: Random Forest is also implemented through sklearn.
- SVM: This was implemented by using sklearn. The kernels used are polynomial, linear and rbf, and different gamma and c values are used.
- Feed-Forward Neural Network: This was implemented with keras. Dropout is applied after each layer to prevent overfitting to the training set. The Relu activation function is used on all the layers except the output layer, which uses the sigmoid function. The loss is calculated with the binary cross-entropy function, and the network is optimized through the Adam optimizer.

[1]https://www.kaggle.com/rtatman/blog-authorship-corpus

5.4 Feature sets

Four different methods are used to vectorize the data. Two of them are implemented from scratch and two of them are implemented through the sklearn libraries. All the feature sets can be extracted from either the original data or the preprocessed data. The classifiers might perform better when using the original data, because the character-level and word-level features need to analyse every character or word in the original blog.

5.4.1 Word-level features

The word-level features are extracted from scratch from the data set and are converted into a matrix of vectors. The decision of which features are extracted was made based upon [13] [10] choice of word features. The selected ones are:

- Total number of words, N
- Average length per word
- Vocabulary richness (total number of unique words/N)
- Number of long words (longer than 6)/N
- Number of short words (shorter than 3)/N
- Yule's K measure

5.4.2 Character-level features

The character-level features are also extracted from scratch from the data and are converted into a matrix of vectors. The features which are extracted were also decided upon [13] [10] choice of character features. The selected ones are:

- Total number of characters, C
- Total number of alphabetic characters(letters)/C
- Total number of upper cases/C
- Total number of digits/C
- Total number of white spaces/C
- Total number of special characters/C

5.4.3 Count Vectorizer

The count vectorizer might be the simplest of all as its job is to tokenize text and create a vocabulary of words from that text. It eventually uses the vocabulary to compute the frequency of every word having vectors representing this information. The sklearn library was used to implement this vector, removing any stop words and punctuation (by default).

5.4.4 TF-IDF Vectorizer

TF-IDF stands for Term Frequency - Inverse Document Frequency, which determine the final score of each word. The Term Frequency is the number of times a particular word is found in the document, while Inverse Document Frequency, is used to downscale the words that are very frequent in all documents. The sklearn library was also used to implement this vector, removing any stop words and punctuation (by default).

6. EVALUATION OF RESULTS

For evaluation, all five algorithms were trained on all the different features vectors, by splitting the data into a training set and a test set. The results are illustrated in tables. Regarding the SVM, for each kernel only the best results given from the different parameters are displayed here. The best parameters for the polynomial kernel are gamma = 0.01 and c value = 0.01, for the linear gamma=0.1 and c value = 0.1 and for rbf gamma = 0.01 and c value = 10. When running the classifiers it was noted that the SVM was taking too long to classify on the character and word feature sets, hence, the SVM results are only shown for the other two feature sets. It should also be noted that the results of the feature sets illustrated in this paper are obtained from the original data and not the preprocessed data. This is because, when the classifiers were trained and tested on the preprocessed data, the results show that they performed much worse.

When evaluating Table 1, it can be seen that the best algorithms for both genders are Naive Bayes and Neural Network, while the worst performing classifier is the SVM using an rbf kernel. For each classifier it can be seen that the majority of results do not vary much from one gender to another. When considering the SVM the best performing kernel is the linear kernel. Logistic regression and random forest although they are not the best, they still provide competitive results. Overall, the f1-score of the male gender are higher than the female gender's scores, which show that the blogs of the male gender were classified better.

When analysing Table 2 it is could be noticed that the best performing algorithms are once again Naive Bayes and NN, having Logistic Regression tightly following up. This time all the SVM kernels performed rather badly on the TF-IDF Vector having each accuracy being 50.40. Moreover, the female gender was the worst classified in SVMs having all precision, recall and f-measure amount to 0.00. On the other hand, when comparing the genders for each other classifier, it can be seem that the results are very similar to each other, hence both genders are classified rather well.

When proceeding to analyse Table 3 it is highly noticeable that the classifiers did not perform as good on the character feature set. In this case, the best classifier is the random forest reaching an f1-score of 0.69 and an accuracy of 79.97. The worst is Naive Bayes as opposed to the results in the tables discussed above. In addition to this, for each classifier (except for random forest), there is a significant difference between the results of the male gender and the results of the female gender. These show that the classifiers performed much better on the male gender, and therefore, the female gender classifications diminished the overall accuracy.

Finally when analysing Table 4 it can be seen that just like the character-level feature set, the word-level feature set did not perform very well. The best classifier is once more the random forest classifier, and the worst is Naive Bayes. The female gender obtained worse results than the male gender. It is clear that the classifiers performed worst on the word-level feature set, and the best on the Counter Vectorizer and TF-IDF Vectorizer. This shows that the features extracted are not as robust as they are needed to be. Furthermore, in most cases, the male gender obtained better results, which may imply that the female features might not be as strong as required. Another reason might be that they need more data to classify the test examples better, as Figure 3 shows that there is slightly less data annotated with female gender.

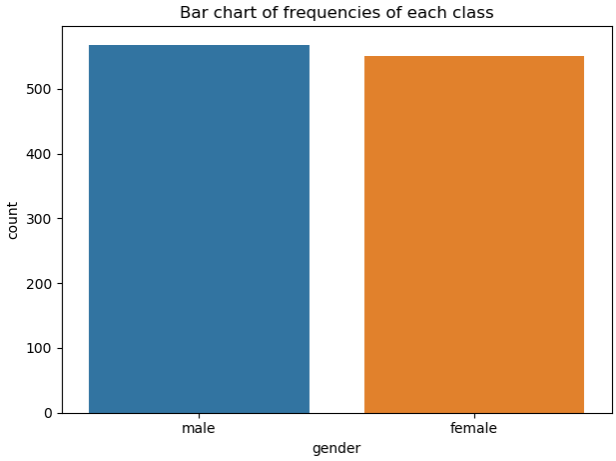

Figure 3: This is a bar plot of the data distribution on the genders.

7. CONCLUSIONS AND FUTURE WORK

This paper is a research of different classifiers and feature sets which can be used for NLP text classification tasks. This study evolves around the gender classification problem from text, specifically blogs. The Blog Authorship Corpus was used to train and test five different classifiers, and extract four different types of feature sets. The algorithms explored are Naive Bayes, Logistic Regression, Random Forest, SVM with different kernels and different hyperparameters, and a feed-forward Neural Network. The feature sets used are the Count vector, TF-IDF vector, character-level feature set and a word-level feature set. From the results it was concluded that the best results out of all experiments are given by Naive Bayes and the Neural Network on the Count vector and the TF-IDF vector. The results also show that in most cases, the male gender was classified better than the female gender. Future work would be done in the aim of obtaining better results with the other two feature vectors. This can be done by extracting more than just six features for each feature set. Furthermore, syntactic features could also be explored and compared with the other features. Regarding the classifiers, the SVM results could be made better by further experimentation with kernels and different gamma and c values, while the Neural Network could be made better with the experimentation of its hyperparameters, as the ones used here might not be the best. Last but not least, the full data set may be used for training and testing, which may result in a significant increase in all the classifiers' performance.

8. REFERENCES

[1] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, pp. 41–46, 2001.

Counter Vectorizer								
		NB	LR	RF	SVM(poly)	SVM(linear)	SVM(rbf)	NN
	precision	0.86	0.76	0.74	0.51	0.68	0.50	0.83
Male	recall	0.80	0.86	0.84	0.98	0.90	1.00	0.86
	f1-score	0.83	0.81	0.79	0.67	0.77	0.67	0.85
	precision	0.81	0.83	0.81	0.73	0.85	0.00	0.85
Female	recall	0.86	0.72	0.70	0.06	0.56	0.00	0.82
	f1-score	0.84	0.77	0.75	0.11	0.68	0.00	0.84
	Accuracy	83.38	79.09	76.94	52.28	73.46	50.40	84.18

Table 1: A table of the results obtained by the Counter Vectorizer

TF-IDF Vectorizer								
	NB	LR	RF	SVM(poly)	SVM(linear)	SVM(rbf)	NN	
	precision	0.83	0.79	0.72	0.50	0.50	0.50	0.83
Male	recall	0.85	0.89	0.84	1.00	1.00	1.00	0.86
	f1-score	0.84	0.84	0.78	0.67	0.67	0.67	0.85
	precision	0.85	0.87	0.80	0.00	0.00	0.00	0.85
Female	recall	0.83	0.75	0.68	0.00	0.00	0.00	0.82
	f1-score	0.84	0.81	0.73	0.00	0.00	0.00	0.84
	Accuracy	83.91	82.31	75.60	50.40	50.40	50.40	84.18

Table 2: A table of the results obtained by the TF-IDF Vectorizer

Character-level Features					
		NB	LR	RF	NN
	precision	0.51	0.53	0.70	0.54
Male	recall	0.84	0.87	0.72	0.79
	f1-score	0.63	0.66	0.71	0.64
	precision	0.52	0.61	0.70	0.62
Female	recall	0.18	0.20	0.68	0.34
	f1-score	0.27	0.30	0.69	0.43
	Accuracy	51.21	53.89	79.97	56.84

Table 3: A table of the results obtained by the character-level feature vectorizer

Word-level Features					
		NB	LR	RF	NN
	precision	0.51	0.52	0.61	0.54
Male	recall	0.47	0.73	0.53	0.90
	f1-score	0.49	0.61	0.57	0.67
	precision	0.49	0.50	0.57	0.52
Female	recall	0.53	0.29	0.64	0.13
	f1-score	0.51	0.36	0.60	0.20
	Accuracy	49.87	51.47	58.71	53.62

Table 4: A table of the results obtained by the word-level feature vectorizer

[2] G. Gasso, "Logistic regression," 2019.

[3] A. Palomino-Garibay, A. T. Camacho-Gonzalez, R. A. Fierro-Villaneda, I. Hernandez-Farias, D. Buscaldi, I. V. Meza-Ruiz, *et al.*, "A random forest approach for authorship profiling," in *Proceedings of CLEF*, 2015.

[4] A. Liaw, M. Wiener, *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[5] V. Jakkula, "Tutorial on support vector machine (svm)," *School of EECS, Washington State University*, vol. 37, 2006.

[6] L. Wang, *Support vector machines: theory and applications*, vol. 177. Springer Science & Business Media, 2005.

[7] D. J. Montana and L. Davis, "Training feedforward neural networks using genetic algorithms.," in *IJCAI*, vol. 89, pp. 762–767, 1989.

[8] J. Schler, M. Koppel, S. Argamon, and J. W. Pennebaker, "Effects of age and gender on blogging.," in *AAAI spring symposium: Computational approaches to analyzing weblogs*, vol. 6, pp. 199–205, 2006.

[9] C. Zhang and P. Zhang, "Predicting gender from blog posts," *University of Massachussetts Amherst, USA*, 2010.

[10] N. Cheng, R. Chandramouli, and K. Subbalakshmi, "Author gender identification from text," *Digital Investigation*, vol. 8, no. 1, pp. 78–88, 2011.

[11] V. Simaki, A. Koumpouri, I. Mporas, and V. Megalooikonomou, "Gender identification of blog authors: Do men and women prefer different character unigrams?," 2011.

[12] A. Bartle and J. Zheng, "Gender classification with deep learning," in *Technical report*, The Stanford NLP Group., 2015.

[13] T. Alexey, "Author gender prediction," 2017.

[14] V. P. Dwivedi, D. K. Singh, S. Jha, *et al.*, "Gender classification of blog authors: With feature engineering and deep learning using lstm networks," in *2017 Ninth International Conference on Advanced Computing (ICoAC)*, pp. 142–148, IEEE, 2017.