Université de Montréal

Personal information prediction from written texts

par

Khalil Bibi

Département d'informatique et recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures et postdoctorales
en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en informatique

mars 2020

Université de Montréal

Faculté des études supérieures et postdoctorales

Ce mémoire intitulé

Personal information prediction from written texts

présenté par

Khalil Bibi

a été évalué par un jury composé des personnes suivantes :

Yoshua Bengio

(président-rapporteur)

Esma Aïmeur

(directeur de recherche)

Philippe Langlais

(co-directeur)

Jian-Yun Nie

(membre du jury)

Mémoire accepté le :

3 Mars 2020

Sommaire

La détection de la paternité textuelle est un domaine de recherche qui existe depuis les années 1960. Il consiste à prédire l'auteur d'un texte en se basant sur d'autres textes dont les auteurs sont connus. Pour faire cela, plusieurs traits sur le style d'écriture et le contenu sont extraits. Pour ce mémoire, deux sous-problèmes de détection de la paternité textuelle ont été traités : la prédiction du genre et de l'âge de l'auteur. Des données collectées de blogs en ligne ont été utilisées pour faire cela. Dans ce travail, plusieurs traits (*features*) textuels ont été comparé en utilisant des méthodes d'apprentissage automatique. De même, des méthodes d'apprentissage profond ont été appliqués. Pour la tâche de classification du genre, les meilleurs résultats ont été obtenus en appliquant un système de vote majoritaire sur la prédiction d'autres modèles. Pour la classification d'âge, les meilleurs résultats ont été obtenu en utilisant un classificateur entrainé sur TF-IDF.

Mots-clés: Détection de la paternité textuelle, traitement automatique des langues naturelles, apprentissage machine, apprentissage profond, vie privée

Summary

Authorship Attribution (AA) is a field of research that exists since the 60s. It consists of identifying the author of a certain text based on texts with known authors. This is done by extracting features about the writing style and the content of the text. In this master thesis, two sub problems of AA were treated: gender and age classification using a corpus collected from online blogs. In this work, several features were compared using several feature-based algorithms. As well as deep learning methods. For the gender classification task, the best results are the ones obtained by a majority vote system over the outputs of several classifiers. For the age classification task, the best result was obtained using classifier trained over TF-IDF.

Keywords: Authorship attribution, natural language processing, machine learning, deep learning, privacy

Contents

Sommaire v

Summary vii

List of tables xiii

List of figures xvii

List of abbreviations xix

Acknowledgements xxi

Introduction 1

Chapter 1. Related works 3

1.1. Style and content features 3

1.2. Topic modeling features 8

1.3. Document embeddings 8

1.4. Readability features 11

Chapter 2. Learning methods 13

2.1. Feature-based models 13

2.1.1. Naive Bayes (NB) 13

2.1.1.1. Gaussian Naive Bayes (GNB) 14

2.1.1.2. Multinomial Naive Bayes (MNB) 14

2.1.1.3. Bernoulli Naive Bayes (BNB) 14

2.1.2. Decision Tree 15

2.1.3. Random Forest (RF) ... 15
2.1.4. Perceptron ... 15
2.1.5. Support Vector Machine (SVM) ... 16
2.1.6. Logistic Regression (LR) ... 17

2.2. Deep learning methods ... 17
2.2.1. Feed-Forward Neural Network (FFNN) ... 17
2.2.2. Recurrent Neural Network (RNN) ... 19
2.2.3. Convolutional Neural Network (CNN) ... 21
2.2.4. Recurrent Convolutional Neural Network (RCNN) ... 21
2.2.5. Early stopping method ... 23

Chapter 3. Text vectorization techniques ... 25

3.1. Letter percentage (LP) ... 25

3.2. Digit percentage (DP) ... 25

3.3. Linguistic Inquiry and Word Count (LIWC) ... 25

3.4. General inquirer (GI) ... 26

3.5. Part Of Speech (POS) ... 26

3.6. Stopwords (SW) ... 26

3.7. Function words (FW) ... 27

3.8. Document embedding (EMB) ... 27

3.9. Vocabulary Richness (VR) ... 28

3.10. Topic modeling ... 28

3.11. Stylometric Features (SF) ... 29

3.12. n-gram features ... 29

3.13. Discriminative n-gram Indexing ... 29

Chapter 4. Methodology 33

4.1. Corpus description 33

4.2. Preprocessing and filtering 35

Chapter 5. Results 37

5.1. Gender classification results 38
5.1.1. Style and content features 39
5.1.2. n-gram features 40
5.1.3. Topic modelling 45
5.1.4. Deep learning methods 46
5.1.4.1. Feed Forward Neural Network (FFNN) 46
5.1.4.2. Recurrent Neural Network (RNN) 47
5.1.4.3. Recurrent Convolutional Neural Network (RCNN) 48

5.2. Age classification results 49
5.2.1. Style and content features 49
5.2.2. n-gram features 50
5.2.3. Topic modelling 52

Chapter 6. Analysis 55

Conclusion 61

Bibliography 63

Appendix A. List of stopwords .... A-i

Appendix B. List of function words .... B-i

Appendix C. List of POS tags .... C-i

Appendix D. Features description from Cheng et al. [2011] .... D-i

Appendix E. Results for n-gram features for gender classification .... E-i

Appendix F. Deep learning results for gender classification . . . . F-i

Appendix G. Results for n-gram features for age classification . . . . G-i

List of tables

1.1 Accuracies (%) reported in Argamon et al. [2009] on the Blogger corpus from Schler et al. [2006] 4

1.2 Comparison between accuracies (%) of Markov et al. [2017] and the winner of PAN AP 2015 over the dataset of PAN AP 2015 9

1.3 PAN author profiling 2015 results reported by Gomez Adorno et al. [2016] 11

1.4 PAN author profiling 2016 results reported by Gomez Adorno et al. [2016] 11

4.1 Statistics from the corpus 35

4.2 Statistics about the splits made over the corpus 36

5.1 Accuracy (%) on the validation corpus for each individual feature using a LR classifier with d being the dimension of the feature vector considered 39

5.2 Greedy search for the best combination using LR with d being the dimension of the features vector considered 40

5.3 Accuracies on the validation corpus of LR classifier over character level 9-grams while varying S 41

5.4 Comparison of DnI and TF-IDF accuracies (%) on the validation corpus for gender classification using LR over some examples of features with d being the dimension of the features vector considered 44

5.5 Comparison of LR results using two representation ways topic modeling features for gender classification 45

5.6 Accuracies (%) on validation and test corpora of the best performing models 49

5.7 Accuracies (%) on the validation corpus of LR classifier over character level 8-grams while varying S 51

5.8 Comparison of DnI and TF-IDF accuracies (%) on the validation corpus for age classification using LR over some examples of features with d being the dimension of the features vector considered 52

5.9 Comparison of LR results on the validation corpus using two representation ways topic modeling features for age classification 53

5.10 Accuracies (%) on validation and test corpora of the best performing models 53

D.1 Detailed features from Cheng et al. [2011] . . . . D-i

E.1 Accuracy (%) for each features individually with each algorithm for gender classification with d being the dimension of the feature vector considered . . . . E-i

E.2 Accuracies (%) for different discriminative n-grams features with n being the length of the sequence considered and S the parameter of the discriminative indexing method . . . . E-ii

E.3 Greedy search to find the best combination using majority vote system over outputs of classifiers trained on Discriminative n-gram Indexing with $S = 5$ . . . . E-v

E.4 TF-IDF results for gender classification with n being the length of the sequence considered while considering all n-grams for each feature . . . . E-vi

E.5 Greedy search for the best combination using majority vote system over outputs of LR classifiers trained on TF-IDF . . . . E-vii

E.6 Greedy search for the best combination using majority vote system over outputs of LR classifiers trained on TF-IDF and others trained on Discriminative n-gram Indexing (DnI) . . . . E-viii

E.7 Accuracies (%) using vector of probability for each topic topic . . . . E-ix

E.8 Accuracies (%) using vector of binary representation for each topic . . . . E-ix

F.1 RNN unidirectional performances with LR=0.01 . . . . F-ii

F.2 RNN bidirectional performances with LR=0.01 . . . . F-ii
F.3 RNN unidirectional performances with LR=0.001 . . . . F-iii
F.4 RNN bidirectional performances with LR=0.001 . . . . F-iii
F.5 Performances of RCNN with LR=0.01 . . . . F-iii
F.6 Performances of RCNN with LR=0.001 . . . . F-iii

G.1 Accuracy (%) for each features individually with each algorithm for age classification with d being the dimension of the feature vector considered . . . . G-i
G.2 Greedy search for the best combination using LR with d being the dimension of the features vector considered . . . . G-ii
G.3 Accuracies (%) for different Discriminative n-grams Indexing (DnI) with n being the length of the sequence considered and S the parameter of DnI . . . . G-ii
G.4 Greedy search to find the best combination using majority vote system over outputs of classifiers trained on Discriminative n-gram Indexing with $S = 5$ . . . . . G-vi
G.5 TF-IDF results for gender classification with n being the length of the sequence considered while considering all n-grams for each feature . . . . G-vii
G.6 Greedy search for the best combination using majority vote system over outputs of LR classifiers trained on TF-IDF . . . . G-vii
G.7 Accuracies (%) using vector of probability for each topic topic . . . . G-viii
G.8 Accuracies (%) using vector of binary representation for each topic . . . . G-viii

List of figures

1.1 3grams assigned for each subgroup for the sentence "The actors wanted to see if the pact seemed like an old-fashioned one." 7

2.1 Illustration of the kernel trick (from Towardsdatascience [c]) 16

2.2 Feed-forward neural network architecture (from Towardsdatascience [b]) 18

2.3 One to One 20

2.4 One to Many 20

2.5 Many to One 20

2.6 Many to Many model 1 20

2.7 Many to Many model 2 20

2.8 Structure of a bidirectionnal RNN (from Towardsdatascience [a]) 20

2.9 Example of applying a kernel over an image for CNN 21

2.10 Structure of RCNN from Lai et al. [2015] for a partial sentence from "A sunset stroll along the South Bank affords an array of stunning vantage points" 22

2.11 Explanation of early stopping method (from Fouryears) 24

3.1 Steps for n-gram indexing with $S = 200$ 31

4.1 Example of two blog posts from the corpus 33

4.2 Number of bloggers male and female as function of their age 34

4.3 Percentage of blog posts for each range of number of words split by gender and age of authors 34

5.1 Comparison of different learning algorithms over DnI features with $S = 5$ while varying n for character-based n-grams 42

5.2 Accuracy (%) of LR classifier on validation corpus with varying the percentage of 6-grams character based 43

6.1 Number of misclassified blog posts from the validation corpus split by the majority vote system using classifiers trained on discriminative n-gram indexing 56

6.2 Number of misclassified blog posts from the test corpus split by the majority vote system using classifiers trained on discriminative n-gram indexing 56

6.3 Percentage of misclassified posts from test corpus for each range of percentages of unseen 8-grams during training 57

6.4 Number of misclassified blog posts from the dev corpus by the best performing system for gender classification regrouped by ranges of number of words 58

6.5 Number of misclassified blog posts from the test corpus by the best performing system for gender classification regrouped by ranges of number of words 58

List of abbreviations

- AA: Authorship Attribution
- AP: Authorship Profiling
- NLP: Natural Language Processing
- LIWC: Linguistic Inquiry and Word Count
- GI: General Inquirer
- POS: Part-Of-Speech
- SW: StopWords
- FW: Function Words
- VR: Vocab Richness
- SF: Stylometric Features
- TF-IDF: Term Frequency–Inverse Document Frequency
- DnI: Discriminative n-gram Indexing
- KNN: K-Nearest Neighbor
- GNB: Gaussian Naive Bayes
- MNB: Multinomial Naive Bayes
- BNB: Bernoulli Naive Bayes
- RF: Random Forest
- SVM: Support Vector Machine
- LR: Logistic Regression or Learning Rate (for deep models)
- FFNN: Feed-Forward Neural Network
- RNN: Recurrent Neural Network
- RCNN: Recurrent Convolutional Neural Network

Acknowledgements

These last two years are the most formative ones during all my academic career and my personal life since it is the first time I start a new experience in a foreign country. I met amazing people here at the University of Montreal who made this experience easier for me. When I came to Canada, I was excited about this whole new life, but I was unsure about what I wanted to do, and after finishing my courses, I was pretty sure about which kind of research project I want to work on.

I want to thank my co-directors Esma Aimeur and Philippe Langlais for the time they considered for my supervision and for the effort they made. Also, I want to thank my workgroup from machine learning and data sciences courses. Moreover, I want to thank all people from the RALI lab who were always there when I needed support. I shared a fantastic time with all of these people, hours of fun through the social events we made, and the board games we played.

Certainly, I want to thank my family especially my mum because none of this could be happening without her support. A special 'thanks' goes to my aunt who is living in Ottawa which made my life easier in Canada. Also, I want to thank my grandmother, my uncles, and my cousins.

Moreover, I want to thank all my Tunisian and Moroccan friends for all the fun moments we shared and for their support.

Introduction

Authorship Attribution (AA) is the task of predicting or identifying the author of a given text based on a collection of documents written by this author and others. This is typically done by leveraging the writing style or the content of such documents. Research in authorship attribution started in the early 1960s with two American statisticians (Mosteller and Wallace [1963]). Authorship Attribution can be used in different applications.

In academia, it helps to detect plagiarism (i.e. university essays) since it considers some linguistic information which is useful for this task (Uzuner et al. [2005]). People can benefit from AA by identifying the authors of messages sent anonymously, an ever-increasing problem given the rising amount of anonymous textual information shared online (poems, blog posts, programming codes, historic text from uncertain authorship, false claims of authorship, criminal threats, bullying messages, sexual harassment text, etc.).

Another context in which authorship attribution can be used is to detect whether a text has been produced by a given author (binary prediction). Another example within the same context is having 2 co-authors who collaborated on writing a book or an article. AA helps us figure out who wrote which part of this collaborative work based on the authors' previous works [1].

AA may also be employed to detect changes in style for a given author like the work of Hirst et al. [2011]. The authors used this type of information in order to infer that the writer (Agatha Christie) was affected by Alzheimer's disease at some point in her carrier. It is also useful to distinguish between single-author and multi-author texts. For instance in PAN 2019 workshop[2], participants were asked to predict the number of authors of a text based on the change of the writing style.

[1]`http://www.aicbt.com/authorship-attribution/`

[2]`https://pan.webis.de/clef19/pan19-web/style-change-detection.html`

We can not assume that an author will always write about the same genre or topic or domain. So, we can not assume that the genre and the topic and the domain are good indicators to predict the author. The trending sub-problems in AA concerning this issue are the *cross-genre*, *cross-domain*, and *cross-topic* problems. For instance, Overdorf and Greenstadt [2016] used 2 cross-domain datasets. The first dataset is taken from blogs and the bloggers' twitter feeds. The second dataset is retrieved from Reddit comments and the same Reddit users' twitter feeds. For each one of the cross-domain datasets, Overdorf and Greenstadt [2016] tried the cross-domain AA task. They trained each time on one of the domains (i.e. blog posts or tweets) and tested on the other one.

In this thesis, instead of predicting a specific author, we will be focusing on a class of authors. We consider a 3-class classification as a way of representing the age of the author ([10-19], [20,29] or >=30), as well as a binary gender classification to predict the author's gender (male or female). These personal information won't affect the author's privacy. However, they could be useful in many cases. Several features and machine learning algorithms are tested in order to classify the texts and to compare which ones are more useful for each task. Also, some deep learning models are tested.

The remainder of this work is structured as follows. Chapter 1 explores some AA's datasets, features, techniques and their results. Chapter 2 explains the algorithms and techniques behind prediction algorithms and deep learning methods which are used for classification. Chapter 3 presents techniques used to vectorize texts in order to use them as input to the classifiers. Chapter 4 describes the corpus used in this thesis and the preprocessing methods. Chapter 5 reports the results of the gender and the age classification task, respectively. Chapter 6 concludes by some analysis made about the results found, and by discussing avenues useful to pursue.

Chapter 1

Related works

Authorship Attribution (AA) is used in several domains, *e.g.* authorship verification and plagiarism detection. Researchers have studied many different types of datasets (long texts, short texts, programming codes) and extracted varied features before using it as input to their models. In this chapter we present some of AA approaches based on the types of features, and will also describe some datasets.

1.1. Style and content features

Stylometric features refer to the extraction of information about an author's writing style, such as prevalent usage of certain parts of speech. Content-based features refer to the analysis of the content of a text, such as the presence of a specific word or a certain sequence of characters. This type of features was used over multiple datasets which were constructed from short texts such as blog posts, tweets and SMS.

Koppel et al. [2008] used style and content features on the Blogger dataset. Schler et al. [2006] built this corpus by collecting more than 600,000 blog posts written by 19,320 bloggers from `Blogger.com`. The average number of words per author was 7,250. The texts were labeled by the age and gender of the authors. We use this dataset in this thesis considering that the number of examples it contains is sufficient to train and test the models within a reasonable time. For stylistic features, Koppel et al. [2008] for instance capture the presence of personal pronouns such as **I, me, him** that they discerned to be more female centric, while male use more often articles such as **the**. As for content features, the authors gave the example of some words which were used more by male authors, and other words which were used more by female authors for the same task of classification. Although the work of Argamon et al. [2009] is not reproducible because of a lack of details, they show that content features outperformed the stylometric features with around 75% of accuracy for both classification tasks. They also studied the impact of combining both of the features which led to better performances. All of the results are shown in table 1.1.

Tab. 1.1. Accuracies (%) reported in Argamon et al. [2009] on the Blogger corpus from Schler et al. [2006]

Task	Baseline	Style	Content	Both
Gender	50.00	72.00	75.10	76.10
Age	42.70	66.90	75.50	77.70

Cheng et al. [2011] used two datasets (Reuters and Enron) to study the impact of 545 linguistic features for AA for a gender classification task. These features were grouped into 5 sub-categories:

- character-based: contains 29 stylometric features such as number of white space characters. These features scored 73.48%.
- word-based: contains 101 features such as "vocabulary richness" and "psycho-linguistic" features. These features scored 59.08%.
- syntactic: composed of 10 stylometric features which provide information about the authors' writing style at the sentence level such as the preference of usage of certain punctuation marks. This type of feature scored individually 65.37%.
- structural-based: regroups 132 features which provide information about an author's habits for line and paragraph structure. This includes features such as the average number of sentences and words per paragraph. These features scored 61.62%.
- function words: composed of 392 content features which shows the percentage of each word, from a certain list, in the text. They scored 74.81%.

All of these sub-categories are described in detail in table D.1, Appendix D. The first dataset consists of 6,769 articles written by Reuters journalists, with an average of 520 words per article. The second dataset is a subset from the Enron[1] email dataset, and consists of 8,970 emails with an average of 116 words per email. Support Vector Machines (SVM), Adaboost Decision Tree and Logistic Regression (LR) classifiers were used with those features. All of the accuracies mentioned above are those of an SVM classifier. The authors varied the number of messages used during training in order to gauge the impact of the number of examples used. They concluded that increasing the number of training examples and using the features explored above with the mentioned classifiers increase the accuracy in each case. SVM was the best performing classifier with an accuracy of 76.75% for the Reuters dataset and 82.23% for the emails dataset by combining all features. The authors also trained classifiers on texts with an upper limit on the length (number of tokens) and tested their performance on texts with the same maximum length. The results showed that SVM perform better with short emails (number of tokens less than 100), with an accuracy of 85.13% using all of the features.

N-gram features are explored in section 3.12 and are another example of content features, as they focus on a sequence of characters/words from the text. Several studies have been conducted to date to understand the importance of n-grams in the context of AA. Rocha et al. [2017] studied the impact of these features on data collected from social media. They constructed a dataset composed of 10 million tweets from 2014 written by 10,000 authors, with each tweet restricted to a maximum of 140 characters including the hashtags. In their experiments, they used character level 4-grams along with word-level and Part Of Speech (POS) n-grams $\forall\ n \in [1,5]$. They restricted the experiments to 50 authors, and studied the impact of changing the number of training documents. For the task of identifying the tweet's author, they reported an accuracy of almost 70% by combining all of the features mentioned above using an SVM classifier and they concluded that increasing the length of the corpus leads to better performances.

Other studies examined the potential of character level n-gram features. For instance, Jankowska et al. [2013] used character 4-grams to compute similarity between different

[1]`https://www.cs.cmu.edu/~enron/`

n-gram profiles, explored in subsection **??**, using equation **??**. For the gender classification task, they report accuracies of 58% on a Spanish dataset and 53% on an English dataset.

Character n-grams tend to give good AA predictions. Sapkota et al. [2015] grouped character-level 3-grams into the following 10 subgroups to capture more information about other n-grams:

(1) prefix: n-gram characters, covering the first n characters of a token
(2) affix: n-gram characters covering the last 3 characters of a token
(3) space-prefix: n-gram characters starting with a space
(4) space-suffix: n-gram characters ending with a space
(5) whole-word: n-gram characters covering a whole token of size n
(6) mid-word: n-gram characters covering the middle parts of a token
(7) multi-word: n-gram characters covering the token suffix, a space and the next token prefix
(8) beg-punct: n-gram characters starting with a punctuation
(9) mid-punct: n-gram characters containing a punctuation in the middle
(10) end-punct: n-gram characters ending with a punctuation

Figure 1.1, taken from Sapkota et al. [2015], shows the 3-grams assigned for each of the subgroups mentioned above for the sentence: "The actors wanted to see if the pact seemed like an old-fashioned one.".

From Figure 1.1 we observe that the character 3gram "one" can be found both as a whole word and as a part within the middle of a word. This method helps to capture the preferences of each author class and was applied to two types of datasets:

- two single-domain datasets were used. The first one was written by 10 authors and the second one by 50 authors.
- two multi-domain datasets which were written by 13 authors.

They showed that, for the single domain datasets, the categories prefix, suffix, space-prefix, and mid-word gave the best performances for identifying the true author of a text. For the multi-domain datasets, the categories prefix, space-prefix, beg-punct and, mid-punct provided the best results for the same task.

Fig. 1.1. 3grams assigned for each subgroup for the sentence "The actors wanted to see if the pact seemed like an old-fashioned one."

Subgroup	3grams assigned to
prefix	act, wan, pac, see, lik, fas
suffix	ors, ted, act med, ike, ned
space-prefix	␣ac, ␣wa, ␣to, ␣se, ␣if, ␣th, ␣pa, ␣li, ␣an, ␣ol, ␣on
space-suffix	he␣, re␣, ed␣, to ␣, ee␣, if ␣, ct␣, ke␣, an␣
whole-word	the, see, old, one
mid-word	cto, tor, ant, nte, eem, eme, ash, shi, hio, ion, one
multi-word	e␣a, s␣w, d␣t, o␣s, e␣i, f␣t, e␣p, t␣s, d␣l, n␣o, d␣o
beg-punct	-fa
mid-punct	d-f
end-punct	ld-, ne.

Term Frequency-Inverse Document Frequency (TF-IDF), used to show how important is a word in a given text, is also part of the content features. This technique has been successfully used in Information Retrieval (IR) and Natural Language Processing (NLP) to classify texts. The two components are given by:

- TF$(e,T) = \dfrac{f_{e,T}}{\sum_{e' \in T} f_{e',T}}$ where $f_{e,T}$ is the number of occurrences of the element e in a text T
- IDF$(e,D) = \log \dfrac{|D|}{|t \in D \text{ with } e \in t|}$
  where:
  - $|D|$ is the number of texts in the data set
  - $|t \in D$ with $e \in t|$ is the number of texts in which the element e appears

TF-IDF(e,T,D) is defined to be TF(e,T) $\times$ IDF(e,D). Several studies have used TF-IDF in AA to give weight to elements from the data set. A text is represented as a vector with each component associated to a specific item in the vocabulary. Grivas et al. [2015] combined TF-IDF for trigrams, and applied it on the PAN AP 2015 dataset. They reported an accuracy of more than 80% in the gender classification task over all languages, using an SVM with a linear kernel, and an accuracy of over 60% using an SVM with a Radial Basis Function (RBF) kernel.

1.2. Topic modeling features

When writing a text in a personal space like a blog, each author tends to write about several topics he is interested in. Using topic information could help us define whether a specific author wrote a text or not. Several studies in AA used topic modeling features to classify a text or to predict the author of a given text. Anwer et al. [2019] used the Improved Sqrt-Cosine similarity (ISC) with Latent Dirichlet Allocation (LDA) on a corpus of 6,000 documents of Urdu newspaper articles written by 15 authors. Texts were vectorized into many dimensions based on the number of topics chosen and the ISC measure was used to give a similarity between two texts which can be mathematically written as

$$ISC(x,y) = \frac{\sum_{i=1}^{k} \sqrt{x_i y_i}}{\sqrt{\sum_{i=1}^{k} x_i}\sqrt{\sum_{i=1}^{k} y_i}} \tag{1.2.1}$$

where k is the dimension of x and y, which is also the number of the topics extracted. They tried two approaches. Instance-based where each document was treated individually. Profile-based where all documents of a specific author were concatenated and treated as a single document. The number of topics is a parameter that affects the performance of the algorithm. Thus, the authors tried several values for this parameter and found that instance-based with 55 topics gives the best accuracy of 92.89%.

1.3. Document embeddings

Document embedding is the process of representing a text or a document into a vector based on the vector representation of its tokens which is done using several methods such as word2vec (Mikolov et al. [2013]). A popular technique used for the document representation is doc2vec (Le and Mikolov [2014]). This technique is explained in more details in section 3.8. Several researches in AA used this technique. Others trained a feed-forward neural network with the aim of learning the language model to represent a text into a vector of embeddings like the work of Markov et al. [2017].

The training was done by learning each token's representation based on the context. After learning that, the text representation is the average or the concatenation of all the vectors of its tokens. The datasets used in their work are:

- PAN AP 2014: it is composed of English and Spanish blog posts and social media, and English reviews. A subset of the training corpus was used to test the model trained on the PAN AP 2016 corpus since, at the time the experiences were done, the test dataset of PAN AP 2016 was not publicly available.
- PAN AP 2015: it is composed of tweets in English, Dutch, Italian and Spanish. The dataset is well balanced in terms of number of texts for each gender (2 classes) and for each class of age (4 classes).
- PAN AP 2016: it is composed of tweets in English, Dutch and Spanish. The dataset is well balanced in terms of number of texts for each gender (2 classes) and for each class of age (5 classes).

For PAN AP 2015, the authors applied the doc2vec method and used a logistic regression classifier on this type of representation. They reported an average accuracy of more than 60% for the gender classification task and almost 65% for the age classification task over all of the languages. The accuracy results reported are good but they did not outperform those of the winner of PAN AP 2015 as shown in table 1.2. The winner of PAN AP 2015 (Carmona et al. [2015]) combined style and content features such as the frequency of stopwords and punctuation marks. They also used representation based on TF-IDF.

Tab. 1.2. Comparison between accuracies (%) of Markov et al. [2017] and the winner of PAN AP 2015 over the dataset of PAN AP 2015

Work	English		Spanish		Dutch	Italian
	Gender	Age	Gender	Age	Gender	Gender
Markov et al. [2017]	65.00	69.08	56.00	62.00	56.67	70.00
Winner of PAN AP 2015	78.28	79.60	91.00	82.00	86.84	91.17

However, for PAN AP 2016, Markov et al. [2017] reported the best accuracies, i.e. 76.98% for gender classification on tweets in English and 77.20% in Spanish and 75.54% in Dutch. They also showed the results for age classification: 46.01% for tweets in English and 50.44% for those in Spanish.

All of the results cited above are single-genre results; the classifier was trained and tested on the same genre of data, but there are other methods available, like the cross-genre. In this method the classifier is trained on some type of dataset and tested on another type. For instance, a Logistic Regression classifier was trained using the training corpus of PAN AP 2016 and then tested on a subset of the training corpus of PAN AP 2014. The accuracies reported in Markov et al. [2017] were not promising since they got, for gender classification tasks on texts written in English, 54.42% of accuracy on blog posts, 49.85% on social media texts and 51.75% on reviews texts.

Gomez Adorno et al. [2016] used the same feature on the corpus from PAN AP 2015 and 2016. However, instead of using a pretrained model to extract the vector representation, they trained a NN for this task. They started by training a neural network to learn the representation of a word into a vector, based on the context of words next to it. After that, they trained another NN to represent the whole text into a vector. They tried 9 representation techniques as input for the NN. Given a corpus $D = [T_1,T_2,\ldots,T_i]$ they reversed the order of texts in order to train the NN to generalize better its output. They reversed as follows:

- inverse order of texts so the input would be: $D' = [T_i,T_{i-1},\ldots,T_1]$
- separation of texts with odd indexes from texts with even indexes, with starting the representation by the odd subset, so the input would be: $D' = [T_1,T_3,\ldots,T_2,T_4]$
- separation of texts with odd indexes from texts with even indexes, with starting the representation by the even subset, so the input would be: $D' = [T_2,T_4,\ldots,T_1,T_3]$
- exchange each of the texts with odd indexes with the text with the next index so the input would be: $D' = [T_2,T_1,T_4,T_3,\ldots]$
- shift two elements to the left, so the input would be: $D' = [T_3,T_4,\ldots,T_i,T_1,T_2]$
- exchange each text with index i with text with index $i+3$, so the input would be: $D' = [T_4,T_5,T_6,\ldots,T_1,T_2,T_3]$
- exchange documents with an index multiple of 3 with the next text, so the input would be: $D' = [T_1,T_2,T_4,T_3,T_5,T_7,\ldots]$
- exchange documents with an index multiple of 4 with the next text, so the input would be: $D' = [T_1,T_2,T_3,T_5,T_4,T_6,\ldots]$
- exchange documents with an index i multiple of 3 with the text with index $i+2$, so the input would be: $D' = [T_1,T_2,T_5,T_3,T_4,T_8,\ldots]$

After training the NN, they compared the impact of preprocessing the data on the performance of the classifiers. For the PAN AP 2015 corpus, two classifiers (SVM and LR) were trained using the resulting representation as features. They obtained almost the same performances each time with accuracies higher than 70% most of the times, for both of the gender and age classification tasks. Thus, sometimes applying a preprocessing method or not lead to the same performances. For PAN AP 2016, LR gave the best performances most of the time with an average accuracy of almost 77% for the gender classification (GC) and more than 50% for the age classification (AC) . Tables 1.3 and 1.4 report the best accuracies obtained by Gomez Adorno et al. [2016] on the PAN 2015 and PAN 2016 corpuses respectively. Each result was obtained after training the classifiers, LR and SVM, using 10-fold cross-validation technique.

Tab. 1.3. PAN author profiling 2015 results reported by Gomez Adorno et al. [2016]

Task	Language	Classifier	Preprocessing	Accuracy (%)
AC	English	LR	Yes	74.34
GC	English	SVM	No	71.71
AC	Spanish	SVM	Yes	69.00
GC	Spanish	SVM	No	75.00
GC	Dutch	SVM or LR	-	76.47
GC	Italian	SVM or LR	-	84.21

Tab. 1.4. PAN author profiling 2016 results reported by Gomez Adorno et al. [2016]

Task	Language	Classifier	Preprocessing	Accuracy (%)
AC	English	LR	Yes	46.71
GC	English	LR	Yes	78.64
AC	Spanish	SVM	Yes	57.20
GC	Spanish	LR	Yes	75.60
GC	Dutch	LR	Yes	77.60

1.4. Readability features

Readability features are indices that gives a score of understandability of a text. These features are commonly used in AA since they provide information about the complexity of the author's writing style. Several indices for measuring this type of information exist such as:

- Automated Readability Index (ARI) (from Smith and Senter [1967]):

$$4.71\Big(\frac{\text{number of letters}}{\text{number of words}}\Big) + 0.5\Big(\frac{\text{number of words}}{\text{number of sentences}}\Big) - 21.43 \tag{1.4.1}$$

- Gunning Fog Index (GFI) (from Gunning [1952]):

$$0.4\Big[\Big(\frac{\text{number of words}}{\text{number of sentences}}\Big) + 100\Big(\frac{\text{number of complex words}}{\text{number of words}}\Big)\Big] \tag{1.4.2}$$

  where complex words are word of 3 syllables or more.

- SMOG (from Laughlin [1969]):

$$1.0430\sqrt{\text{number of polysyllables} \times \frac{30}{\text{number of sentences}}} + 3.1291 \tag{1.4.3}$$

  where a polysyllable is a word having more than two syllables.

Gressel et al. [2014] used the above mentioned readability indices and other ones and combined them with other features. They used the PAN 2014 corpus which is composed of hotel reviews, social media data, tweets and blog posts. For each type of those data, the corpus was almost well balanced for the gender classification task, but not for age classification. An average accuracy of 50% was reported for the gender classification task over all types of data which would be equivalent to a random classifier, so, these features are not used in this thesis. They also reported an average of 25% for the age classification task over all of types of data over 5 age groups.

In this literature review, we have described a number of studies reporting many features, machine learning algorithms, corpora and tasks. This landscape lacks however, a systematic comparison of approaches proposed so far, which is the main motivation of our work. Instead of gathering yet another corpus, we decided to study thoroughly one benchmark involving 2 tasks: gender and age classification, mainly for practical considerations (not too big, not too small).

Chapter 2

Learning methods

The second chapter of the thesis explores the learning methods used in this work for the classification task, such as deep learning and other feature-based methods.

2.1. Feature-based models

This section presents feature-based learning algorithm. They are used for classification and sometimes for regression tasks. Each element from the dataset can be seen as a tuple (x_i,y_i) where x_i is the input and y_i is the class or the value to predict.

2.1.1. Naive Bayes (NB)

Naive Bayes (NB) classifier is based on Bayes rule which can be written as

$$P(y \mid x) = \frac{P(x \mid y)P(y)}{P(x)} \tag{2.1.1}$$

where $x = x_1,x_2,x_3,\ldots,x_n$ is an input sequence and y is the class. Assuming x_i are independent of each other given the class, we have:

$$P(y \mid x_1,x_2,\ldots,x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i \mid y)}{P(x)} \tag{2.1.2}$$

We are interested in $\hat{y} = argmax_y P(y \mid x)$ which is

$$\hat{y} = argmax_y P(y) \prod_{i=1}^{n} P(x_i \mid y) \tag{2.1.3}$$

since the denominator is fixed for the maximisation.

2.1.1.1. *Gaussian Naive Bayes (GNB)*

One way to extend NB to real-valued attributes is to suppose that the data is following a Gaussian distribution.

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \tag{2.1.4}$$

where μ_y (the mean) and σ_y^2 (the standard deviation) are the parameters being learned.

2.1.1.2. *Multinomial Naive Bayes (MNB)*

Multinomial Naive Bayes is often used for text classification where a document is represented as a vector of the count of its words. A vector $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$ is parameterizing this distribution with θ_{yi} being the probability $P(x_i \mid y)$ of a feature i to appear given a class y. The parameters of θ_y are estimated using a smoothed maximum likelihood (e.g. relative frequency counting):

$$\theta_y = \frac{N_{yi} + \alpha}{N_y + \alpha n} \tag{2.1.5}$$

where:

- $N_{yi} = \sum_{x \in T} x_i$ represents the number of times a feature i appears in an element of class y in the training
- $N_y = \sum_{i=1}^{n} N_{yi}$
- $\alpha \geq 0$ is a smoothing parameter

2.1.1.3. *Bernoulli Naive Bayes (BNB)*

Bernoulli Naive Bayes is similar to Multinomial Naive Bayes. However, instead of using the word count vectors or TF-IDF scores it uses boolean variables. For instance, if a word appears in a certain text it will take 1 in the vector. Otherwise the value will be 0. For a classification task, the likelihood of a text given a class C_j can be mathematically written as

$$p(x \mid C_j) = \prod_{i=1}^{n} p_{ji}^{x_i} (1 - p_{ji})^{(1-x_i)} \tag{2.1.6}$$

where p_{ji} is the probability of the feature x_i to be generated by class C_j.

2.1.2. Decision Tree

A decision tree is a tree used for classification or regression tasks. It can be seen as a flowchart or a graph where each node represents a test to be made on a specific attribute or feature. Each branch from the node represents the outcome of the test. The path followed from root to leaf is a sequence of classification tests. While building a decision tree, one has to consider when to stop because if one stops too late it will overfit the training set while if one stops too early, it leads to under-fitting.

2.1.3. Random Forest (RF)

Random forest is the grouping of many decision trees as a solution to over-fitting. The bagging algorithm (Breiman [1994]) is usually used to train the decision trees. A bagging algorithm consists of training several classifiers on different sub-sets from the training set. Then, a majority vote is applied over the predictions of these classifiers.

2.1.4. Perceptron

Perceptron is a binary classification algorithm which learns a linear function in order to classify an input:

$$f(x) = \langle \boldsymbol{w}, x \rangle + \boldsymbol{b} \tag{2.1.7}$$

where:

- w is the weight vector of dimension d
- b is the bias

If the data is linearly separable, the perceptron algorithm will always find the solution (Rosenblatt [1962]). The perceptron algorithm works as follows:

(1) Initialize the parameters w and b with 0

(2) While there are misclassified inputs, we pick a random input from the training data and do:

   (a) If $f(x_i) \times y_i > 0$ where $y_i \in \{-1,1\}$ then the input x_i was correctly classified and no further action is needed

   (b) Otherwise x_i is misclassified and we update as follows:

   - $w \leftarrow w + (y_i \times x_i)$
   - $b \leftarrow b + y_i$

If the data is not linearly separable, the algorithm does not terminate and we have to choose a stopping criteria such as limiting the number of iterations over the training data.

2.1.5. Support Vector Machine (SVM)

SVM is an algorithm that learns a discriminant function $f(x)$ to separate two classes by using a hyperplane, which can mathematically be written as

$$f(x) = \langle \boldsymbol{w}, x \rangle + \boldsymbol{b} \tag{2.1.8}$$

where $\boldsymbol{w}$ is the weight and $\boldsymbol{b}$ is the bias. The algorithm chooses a hyperplane to maximize the margin. The margin is the distance from the hyperplane to the nearest training examples from different classes. Two types of margins are used by an SVM, soft margin that tolerates some errors and hard margin that does not. In most cases, the data is not linearly separable. Thus, a kernel trick is used to obtain a nonlinear SVM. The principle behind this is to project the data in a new feature space where the data could be linearly separable as illustrated in Figure 2.1.

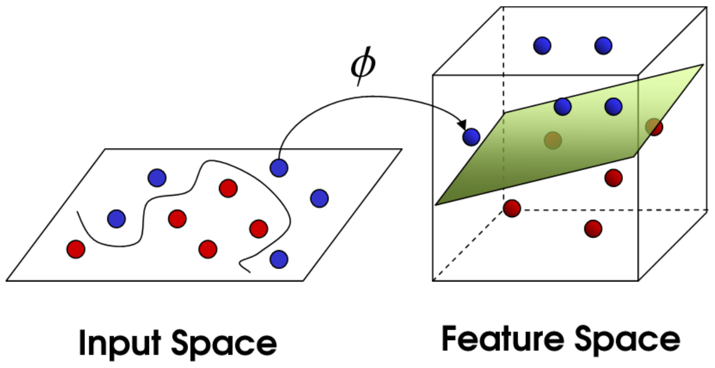

Fig. 2.1. Illustration of the kernel trick (from Towardsdatascience [c])

Examples of kernel functions typically used are:

- Polynomial (homogeneous) : $k(\vec{x_i},\vec{x_j}) = (\vec{x_i} \cdot \vec{x_j})^d$
- Polynomial (inhomogeneous) : $k(\vec{x_i},\vec{x_j}) = (\vec{x_i} \cdot \vec{x_j} + 1)^d$
- Gaussian radial basis function (RBF) : $k(\vec{x_i},\vec{x_j}) = \exp(-\gamma \|\vec{x_i} - \vec{x_j}\|^2)$ for $\gamma > 0$ and sometimes fixed as $\gamma = \dfrac{1}{2\sigma^2}$
- Hyperbolic tangent : $\tanh(\kappa \vec{x_i} \cdot \vec{x_j} + c)$ for some $\kappa > 0$ and $c < 0$

In this thesis, an RBF kernel is used for the experiments with the SVM.

2.1.6. Logistic Regression (LR)

Logistic regression is used for classification and regression tasks:

$$f_{\mathbf{w},\mathbf{b}}(x_i) = \text{sigmoid}(\langle \boldsymbol{w}, x_i \rangle + \boldsymbol{b}) \tag{2.1.9}$$

where $\text{sigmoid}(x) = \dfrac{1}{1+\exp(-x)}$ is the activation function, $\boldsymbol{w}$ is the weight matrix, and $\boldsymbol{b}$ is the bias vector. Each of the parameters $\boldsymbol{w}$ and $\boldsymbol{b}$ is initialized randomly and is updated following the gradient in order to get closer to the optimal parameters. The *sigmoid* function returns a probability and therefore a threshold is used. For example if $\text{sigmoid}(\langle \boldsymbol{w}, x_i \rangle + \boldsymbol{b}) > 0.5$, the class is 0 otherwise the class is 1. This is a limitation of the logistic regression because if the data is not linearly separable many examples might be misclassified.

2.2. Deep learning methods

Deep learning methods offer a powerful system to represent complex functions (Goodfellow et al. [2016]). Deep learning models are used for classification and regression tasks. This section explores some deep learning methods used for classification and regression problems.

2.2.1. Feed-Forward Neural Network (FFNN)

Neural networks are inspired from the human brain. The feedforward neural network, also known as Multi-Layer Perceptron (MLP), is the quintessential model for deep learning. It learns a function $f(x)$ that maps an input data x to a class y by learning a set of parameters $\boldsymbol{\theta}$. The final output would be $y = f(x; \boldsymbol{\theta})$ with $\boldsymbol{\theta}$ as the parameters which is composed of a set of weight matrices and bias vectors. A FFNN can stack layers, so the final output is the one of applying an activation function of the output of previous layers. Each layer has its own parameters and its activation function.

The simplest architecture of a FFNN is composed of two layers which are the input layer and the output layer. It is similar to a logistic regression model. Each layer between the input and output layers is called a hidden layer and it is composed of hidden units (neurons). A hidden unit takes input data and operates on it through its activation function after multiplying it by its weight matrix and adding its bias. One of the most used activation functions for the hidden units is the *RELU* function and for the output layer the so-called *softmax*.

As parameters, a weight matrix and a bias vector are assigned to each hidden layer based on how many hidden units it has. Two adjacent layers are fully connected so each hidden unit is fully connected with all hidden units in the adjacent layer, as shown in figure 2.2.

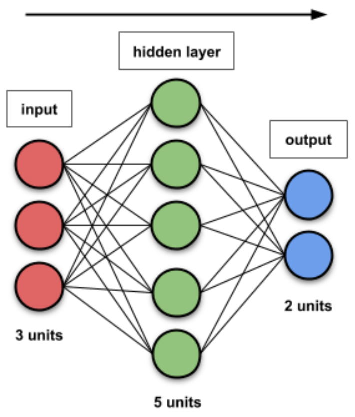

Fig. 2.2. Feed-forward neural network architecture (from Towardsdatascience [b])

During training, a back-propagation method is used to compute the gradient of the parameters by computing the derivatives of the error function with regard to the weights (Werbos [1990]). And an optimizer is used to update the parameters using this gradient. It is also used to optimize the time demanding training of neural networks Goodfellow et al. [2016]. Many algorithms could be used for optimization such as Stochastic Gradient Descent, Adam, Adadelta, Adagrad, RMSProp, Adamax and Nadam. A cost function is used to penalize the model for the bad prediction it is making. A cost function can be written as

$$C(f_{\boldsymbol{\theta}},D_n) = \frac{1}{n}\sum_{i=1}^{n} L(y_i,f_\theta(x_i)) \tag{2.2.1}$$

where:

- $f_{\boldsymbol{\theta}}$ as the function with parameter $\boldsymbol{\theta}$
- D_n as a data set with n elements
- $L(y_i,f_\theta(x_i))$ as the loss function which calculates the mismatch between the prediction of the model and real target. A loss function for example could be the log-likelihood of the categorical softmax prediction (e.g. cross-entropy).

Another parameter involved in the training phase, is the step size also known as the learning rate, which determines how fast the NN has to go to the optimum.

2.2.2. Recurrent Neural Network (RNN)

A Recurrent neural network (RNN) is a type of NN that allows the usage of previous inputs and outputs to form the current hidden state. The RNN is also used for language modeling: $p(w_1, \dots, w_T) = \prod_{t=1}^{T} p(w_t \mid w_{t-n-1}, \dots, w_{t-1})$ where n is the number of previous words to consider. The main difference between the RNN and the FFNN is the ability to use sequences of variable length as inputs instead of inputs with a fixed length or dimension. This type of NN is used with text data into many architectures such as:

- **One to One:** this is similar to the basic model of NN where there is a single input and a single output as shown in figure 2.3.
- **One to Many:** this architecture is used, for instance, in some models for music generation, the input is a sequence of music notes and the model generates the rest like shown in figure 2.4.
- **Many to One:** this architecture is used for sentiment classification, for example when dealing with a comment about a product whether is it a positive or a negative comment.
- **Many to Many:** there are two types for this architecture. The first type is illustrated in figure 2.6 and is used for POS-tagging where the input is a sentence and the output is the tag of each token from the sentence. The second type illustrated in figure 2.7 is used in machine translation for example when the input is a sentence in a certain language and the output is its translation in another language, possibly of a different length.

For this thesis, the architecture of many to one is used because the main concern is a classification task of texts.

The output of a hidden state at time t is mathematically written as

$$h_t = \tanh(\mathbf{b} + U h_{t-1} + \mathbf{W} C(w_t)) \tag{2.2.2}$$

where:

- C is the lookup table
- $\mathbf{W}$ is the weight
- $\mathbf{b}$ is the bias

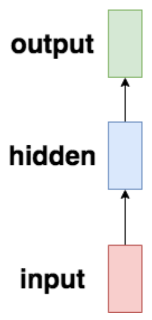

Fig. 2.3. One to One

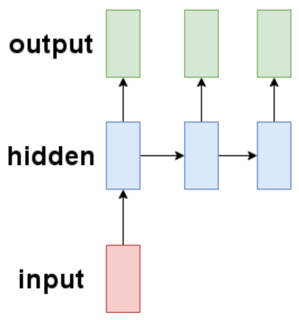

Fig. 2.4. One to Many

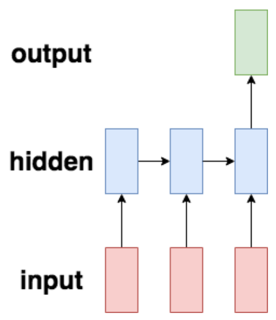

Fig. 2.5. Many to One

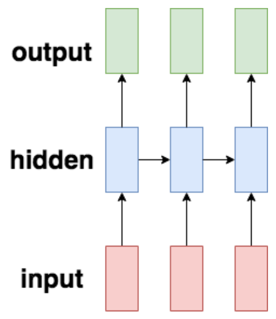

Fig. 2.6. Many to Many model 1

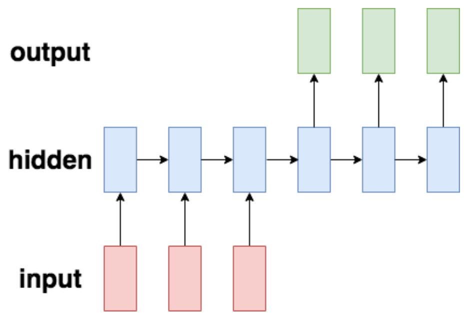

Fig. 2.7. Many to Many model 2

- U are weights that the current hidden state received from the previous hidden state at time $t-1$

A particular variant of RNN is the bidirectional RNN. It consists in putting two independent RNNs together, and the input sequence is fed in order to the first RNN and in reverse order in the second one. The output of the above mentioned RNNs is combined (*e.g.* concatenated or summed) at each time step. Figure 2.8 shows the structure of a bidirectional RNN.

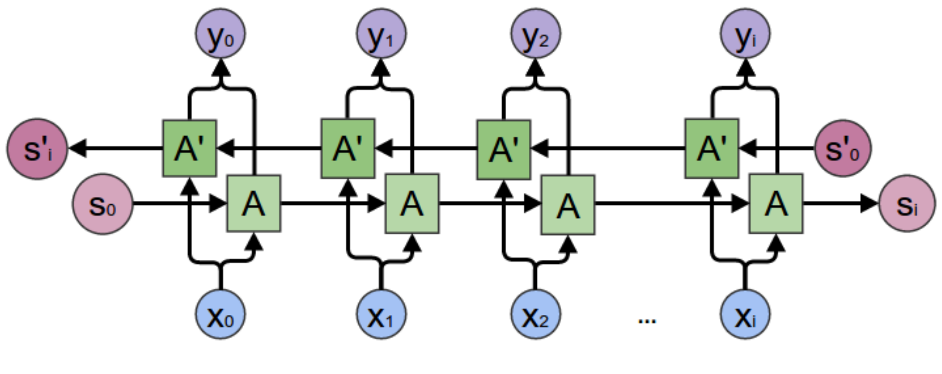

Fig. 2.8. Structure of a bidirectionnal RNN (from Towardsdatascience [a])

2.2.3. Convolutional Neural Network (CNN)

The Convolutional Neural Network (CNN) (LeCun et al. [1989]) is a type of NN that is often used in image processing to solve problems like object recognition. Its name indicates that it applies a convolution operation instead of an ordinary matrix multiplication operation in at least one of its layers (Goodfellow et al. [2016]). In the convolutional layer, a kernel is used in order to extract features. The output is known as feature map. Figure 2.9 shows an example of the output of applying a kernel over an image.

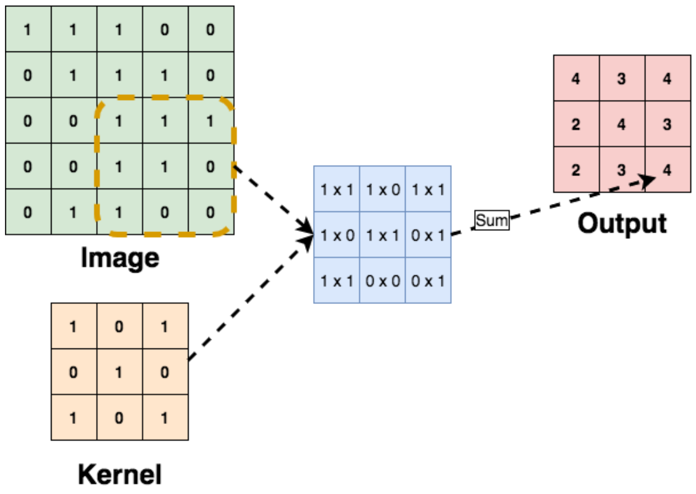

Fig. 2.9. Example of applying a kernel over an image for CNN

Another layer is used for this type of neural network whose main goal is to reduce the convolved feature. This layer is known as pooling layer. There are two main variants for this layer: max pooling and average pooling. The first variant returns the maximum value of the part of the image covered by the kernel. The second one returns the average. The CNN is also used for text classification (Kim [2014]). However, for this task, a 1d convolutional layer should be used to extract features from the text's words representations.

2.2.4. Recurrent Convolutional Neural Network (RCNN)

Recurrent Convolutional Neural Network (RCNN) (Lai et al. [2015]) is a type of NN which is the combination of CNN and RNN. The model of RCNN is illustrated in figure 2.10.

The input of Figure 2.10 is a text T which is composed of a sequence of words w_1,w_2, . . . ,w_n

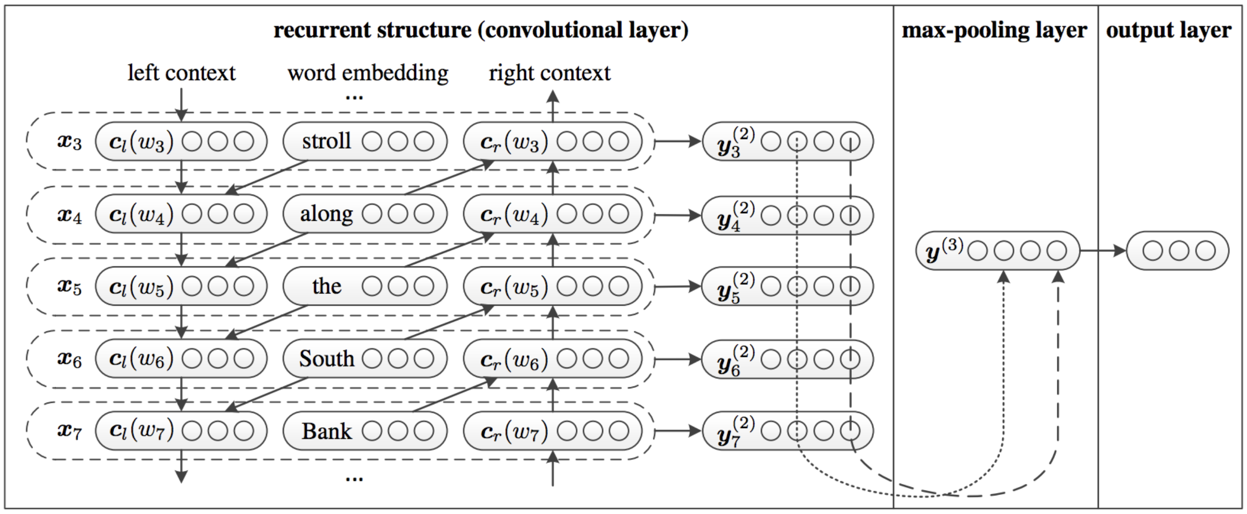

Fig. 2.10. Structure of RCNN from Lai et al. [2015] for a partial sentence from "A sunset stroll along the South Bank affords an array of stunning vantage points"

and the output contains class elements that can be written as $p(c|T;\theta)$ which is the probability of text T to belong to class c and θ being the parameters of the model.

For this model the first step is learning the word representation. In order to obtain more information about a word meaning for this model, a word is represented by combining it with its context. To do so, a bidirectional RNN, explored in the previous subsection, is used. $c_l(w_i)$ and $c_r(w_i)$ are defined as the left and right contexts of the word w_i respectively following the equations 2.2.3 and 2.2.4

$$c_l = f(W^{(l)}c_l(w_{i-1}) + W^{(sl)}e(w_{i-1})) \tag{2.2.3}$$

$$c_r = f(W^{(r)}c_r(w_{i+1}) + W^{(sr)}e(w_{i+1})) \tag{2.2.4}$$

where:

- $e(w)$ as the word embedding of the word w
- $W^{(l)}$ and $W^{(r)}$ the matrices that transform the current hidden layer into the next hidden layer
- $W^{(sl)}$ and $W^{(sr)}$ the matrices used for combination of the semantic of the current word with the next word's left and right contexts respectively
- $f(x)$ is a non linear activation function

Following the example from figure 2.10, $c_l(w_5)$ encodes the semantics of "…stroll along" and $c_r(w_5)$ encode the semantics of "South Bank…". In order to represent a word w_i, we concatenate $c_l(w_i)$, $e(w_i)$ and $c_r(w_i)$ illustrated as follows

$$x_i = [c_l(w_i); e(w_i); c_r(w_i)] \tag{2.2.5}$$

After obtaining the representation x_i of word w_i explained in equation 2.2.5, a linear transformation is applied to x_i and the results are sent to the next layer.

$$y_i^{(2)} = \tanh(W^{(2)} x_i + b^{(2)}) \tag{2.2.6}$$

After computing the word representations, the second step is to compute the text representation by applying a max-pooling layer over all the text's words.

$$y_k^{(3)} = \max_{i=1}^{\mathrm{n}} y_{ik}^{(2)} \tag{2.2.7}$$

The k-th element of $y^{(3)}$ is therefore the maximum of the k-th elements of $y_i^{(2)}$.

The last layer in this model is an output layer that follows the equation

$$y^{(4)} = \mathbf{W^{(4)}} y^{(3)} + \mathbf{b^{(4)}} \tag{2.2.8}$$

and the softmax function is applied to convert it to probabilities which can be written as

$$p_c = \frac{\exp(y_c^{(4)})}{\sum_{j=1}^{n} \exp(y_j^{(4)})} \tag{2.2.9}$$

where c is a class and p_c is its probability.

2.2.5. Early stopping method

One of the common methods shared by all of the deep learning techniques explored above is the early stopping method which allows us to stop the training at some epoch and obtain the best performing parameters on the validation dataset. Figure 2.11 illustrates how this method works. This method was applied with all deep learning experiments.

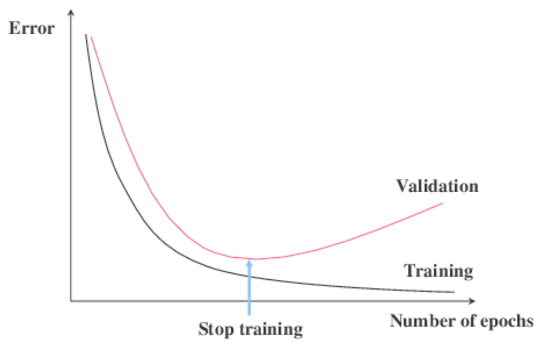

Fig. 2.11. Explanation of early stopping method (from Fouryears)

Chapter 3

Text vectorization techniques

In order to use the feature-based algorithms explored in the previous chapter, the texts must be represented by vectors. Several methods of representation are explored in this third chapter which gave the most interesting results. Some other vectorization techniques were eliminated due to their bad performances.

3.1. Letter percentage (LP)

A 26-dimensional vector captures the frequency in the text of each letter in the alphabet.

3.2. Digit percentage (DP)

A 10-dimensional vector captures the frequency of each digit [0-9] in the text.

3.3. Linguistic Inquiry and Word Count (LIWC)

LIWC is a textual analysis program developed by researchers in psychology (Pennebaker et al. [2015]). It is used to represent the proportion of some categories of words expressing emotions or the way authors think. The last version of LIWC, which is used in this project, was released in 2015. Essentially, LIWC takes a text as input and encodes it in a 76 dimensional vector where each element represents the proportion of a certain category. The categories of LIWC are regrouped in 4 major classes: standard linguistic dimensions, psychological processes, personal concerns, and spoken categories. As an example of categories from LIWC, we have "Total function words" which regroups the sub-categories "Total pronouns", "Articles", "Prepositions", "Auxiliary verbs", "Common Adverbs", "Conjunctions" and "Negations". As an emotion category, we can mention the "Negative emotion" which belongs to the "Affective processes" category, which is part of the "Psychological processes".

3.4. General inquirer (GI)

General inquirer[1](Stone et al. [2007]) is a system for content analysis; it contains 184 word categories that a word in a text could belong to. As an output, general inquirer gives a vector of dimension 184, which represents the number of words for each category or the proportions for each category. The main reason we are using such ensemble of features is that it contains sentiment categories that provide sentiment analysis information. Positive, negative, religion, politics, place, social, and region are examples of word categories provided by the GI.

The difference between LIWC and GI is that LIWC gives an idea about the type of words used such as verbs and adverbs and regroups some words into psychological class and personal concerns class. GI has a list of specific words related to each domain and counts the occurrences of those words without giving an idea about its type.

3.5. Part Of Speech (POS)

Part of speech is simply a list of tags used to identify the type of words such as nouns, verbs, adverbs, adjectives and punctuation marks. The list of POS tags used in this project is presented in Appendix C. By applying this method, we represent each text in a vector of 45 values, where each value is the percentage of appearance of the corresponding POS tag in the text. NLTK [2] is used to tag the texts.

3.6. Stopwords (SW)

Stopwords is a predefined list of the most common words used in a language (*e*.g. 'i', 'me' and 'the' in English). They are used in most of texts without giving additional information to a document in particular. The list of stopwords used for this thesis is from NLTK[3] and is presented in Appendix A. Using this method to vectorize the text yields a vector of dimension

[1]`http://www.wjh.harvard.edu/~inquirer/`

[2]`http://www.nltk.org/book/ch05.html`

[3]`https://www.nltk.org/book/ch02.html#code-unusual`

179 where each coefficient represents the percentage of appearance of a given stopword in the text.

3.7. Function words (FW)

Function words include determiners, prepositions, pronouns, verbs, auxiliary, and question words. They express a grammatical relationship between words in a sentence. The list of function words used for this project is presented in Appendix B and was obtained from Schler et al. [2006]. For instance, 'about', 'first' and 'from' belong to the list. A text vectorization using this method gives a vector of dimension 467 where each value is the percentage of appearance of a function word in the text.

3.8. Document embedding (EMB)

Word embeddings is a trending topic in computational linguistics. It consists in representing words in a vector space, such that related words are closely represented in this space. For example, the distance between the vectors of "dog" and "cat" should be shorter than the one between the vectors of "dog" and "computer". There are several ways of representing words into vectors using pre-trained models that are ready to use such as GloVe[4], fastText[5] and Word2Vec[6].

Document embeddings is an extension of word embeddings. The difference is that word-embedding techniques project a word into a vector but document-embedding technique learns how to project a whole document into a vector based on the representation of its words. In this project, each text is considered as a document and is represented in a vector of dimension 100 using Flair (Akbik et al. [2018]). This feature can provide an idea about texts that are discussing the same topics since the distance between their vectors will be small.

[4]`https://nlp.stanford.edu/projects/glove/`

[5]`https://fasttext.cc/`

[6]`https://radimrehurek.com/gensim/`

3.9. Vocabulary Richness (VR)

VR is one of the measures used in authorship attribution (Stamatatos [2009]) which gives information about the diversity of the author's vocabulary. For this project, 5 measures are chosen to represent each text from the corpus:

- Yule's K measure : $10^4 \left(-\frac{1}{N} + \sum_{i=1}^{N} V(i,N) \left(\frac{i}{N}\right)^2\right)$
- Simpson's D measure : $\sum_{i=1}^{V(N)} V(i,N) \frac{i}{N} \frac{i-1}{N-1}$
- Brunet's W measure : $N^{V(N)^{-a}}$ with $a = -0.172$
- Sichel's S measure : $\frac{V(2,N)}{V(N)}$
- Honoré's H measure : $100 \frac{\log(N)}{1-\frac{V(1,N)}{V(N)}}$

where

– N is the number of tokens

– $V(N)$ is the vocabulary size

– $V(i,N)$ is the number of elements occurring i times

The five measures above are embedded in a 5-dimensional vector for each text.

3.10. Topic modeling

Topic modeling is a statistical method that automatically discovers the topics written about in numerous texts and gives a percentage of appearance for each topic. Latent Dirichlet Allocation (LDA) [7] (Blei et al. [2003]) is one of the models used for topic modeling. When using LDA we must choose the number of topics to extract. For this project, LDA is used for topic modeling with varying the number of topics. n number of topics is chosen to be extracted from the corpus. After the extraction of these topics, each text can be represented by a vector of:

- probabilities of each topic. As a result, a vector of dimension n for each text is obtained.
- one-hot encoding over the probabilities vector which gives 0 to a topic if it does not appear in a text and 1 otherwise; therefore a vector of dimension n for each text is obtained.

[7] `http://blog.echen.me/2011/08/22/introduction-to-latent-dirichlet-allocation/`

3.11. Stylometric Features (SF)

This method vectorizes a text in a vector of dimension 28 containing:

(1) Count of characters

(2) Percentage of alphabetic characters

(3) Percentage of digit characters

(4) Percentage of uppercase characters

(5) Percentage of lowercase characters

(6) Percentage of lowercase emojis

(7) Percentage of each of the 12 punctuation marks from the list: ! ? ; : . , ' " ( ) - _

(8) Count of words

(9) Percentage of short words ($| w | < 4$)

(10) Percentage of long words ($| w | \geq 4$)

(11) Average number of words per sentence

(12) Average number of characters per sentence

(13) Percentage of stopwords

(14) Percentage of function words

(15) Ratio of number of distinct words to the total number of words : $\frac{\text{count(set(words))}}{\text{count(words)}}$

(16) Percentage of hapax (words that occur exactly once) in the text

(17) Percentage of dislegomena (words that occur twice) in the text

3.12. n-gram features

n-gram-based features can be used to vectorize a text by extracting a list that contains the m most used n-grams in the corpus. Then, for each text, we extract the count or the proportion for each n-gram of the list. Each text is therefore represented in a vector of dimension m. In this study, we consider a character, word and POS tags level n-gram.

3.13. Discriminative n-gram Indexing

In the traditional n-gram based features, we extract a list which contains the m most used n-grams. Then, for each text, we extract the proportion for each n-gram of that list and we represent the text in a vector of dimension m. This type of features tends to ignore the existence of too many n-grams, and that's why it loses information.

Furthermore, it consumes a lot of memory and it can also be time demanding. Consequently, the following method is used to deal with all of the n-grams seen during the training.

The main goal of our method is to give a discriminative index of usage for each n-gram for each author or each class of authors (i.e., gender classes or age range classes) based on a hyper-parameter S. This hyper-parameter is used to determine the index of each n-gram. The least discriminative n-gram will get a discriminative index equal to 0. The most discriminative one will get an index equal to $S - 1$. To do so, we follow these steps, also illustrated in Figure 3.1:

(1) We extract an n-gram profile for each class of authors, which is a list of all n-grams, seen during the training, with their number of occurrences. For the gender classification problem using unigram tokens, considering "the" was used 1000 times by the male class and 750 times by the female class.

(2) For each n-gram, we extract its probability to be used by each class of authors. Following our example, the output for the token "the" would be 57.14% and 42.86% for the male and female classes respectively.

(3) An n-gram is deemed discriminative if one of its probabilities from the previous step is higher than $\frac{100}{\# \text{ classes}}$.

(4) We assign each discriminative n-gram to the class with the highest probability and we give it a discriminative index based on the percentages from the previous step. In order to get a discriminative index, we split the interval $[\frac{100}{\# \text{ classes}}, 100]$ into S slices. After splitting the interval, we fetch for the index of the range of percentage to which belongs the n-gram's percentage. With our example, for the gender classification, our scale is $S = 200$ slices in the interval [50 and 100] so 50.01% will get the index 0, 50.26% will get the index 1 and 100% will get the index 199. The token "the" would be assigned to the male class with a discriminative index equals to 28 since the interval of percentage [57.00,57.25] has the index 28, indicating a moderate bias in favor of the male class.

After indexing all n-grams seen in the training, we start by representing each text following these steps:

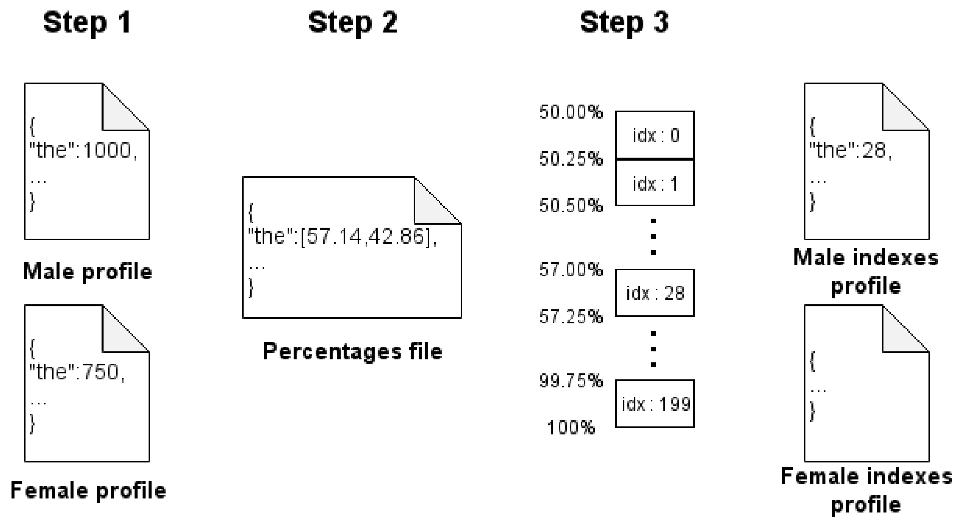

Fig. 3.1. Steps for n-gram indexing with $S = 200$

(1) We initialize a vector of S zeros for each class of authors. So if we have 2 classes, we initialize 2 vectors each one of S zeros.

(2) For each n-gram from the text, we check if the n-gram was seen during the training. If so, we add its number of occurrences to its index in the vector of the class to which it was assigned.

(3) We concatenate all vectors together. If we have 2 classes, the dimension of the concatenated vector would therefore be $S \times 2$. Following the same example above, each text would be represented in a vector of dimension 400.

(4) We normalize the concatenated vector by dividing each element by the sum of all elements.

As a result for this method each text is represented in a vector of dimension $S\times$ number of classes. For this thesis, different values of S will be compared later on.

Chapter 4

Methodology

4.1. Corpus description

The blogger corpus, used for this thesis, was generated by Schler et al. [2006]. It consists of 681,288 blog posts gathered from 19,320 bloggers in 2004 from `Blogger.com`. The blog posts were written in English but still include some noise like Arabic and Chinese letters.

For the gender classification problem, in our corpus, the number of male and female bloggers is equal, so our corpus is well balanced for this type of classification. For the age classification, there are 8,240 bloggers in their "10s" (ages between 10 and 19), 8,086 in their "20s" (ages between 20 and 29), and 2,994 bloggers in their "30s" or higher.

Figure 4.1 shows two examples of blog posts. A 45 year old man wrote the first example, and a 33 year old woman wrote the second one.

urlLink Google News For those of you you have the google tool bar on your browser you can add blogger to it and gain access right into posting to the House Ki'RK Blogger. Try it it's fun!
I have awesome health insurance. Total costs billed so far: $21,847 Total out of pocket: $128 Not everything has been billed yet but this whole injury is probably not going to cost me more than $200. Finally, after years of paying into various health insurance policies I get my money's worth.

Fig. 4.1. Example of two blog posts from the corpus

This corpus was chosen because of its size, since data scarcity is a significant limitation in machine learning and deep learning. In particular, this corpus being composed of almost

700,000 blog posts, we consider it is enough data to train the models and test them within a reasonable time. Furthermore, this corpus is freely available and accessible online and well-annotated.

Figure 4.2 shows the number of bloggers of each age, varying between 13 and 48 since these are, respectively, the minimum and the maximum ages in our corpus. Based on this figure, we conclude that the corpus is perfectly balanced in terms of bloggers' gender. However, it is not well balanced for each range of age. Figure 4.3 shows the percentage of blog posts with a given range of words as a function of gender and age of the authors. This gives an overview about who tend to write longer or shorter texts.

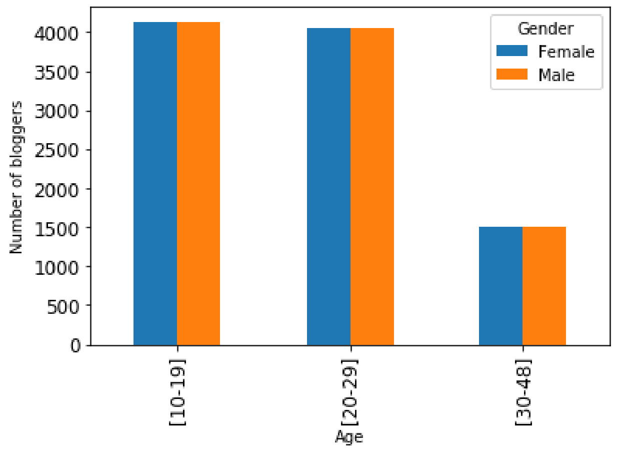

Fig. 4.2. Number of bloggers male and female as function of their age

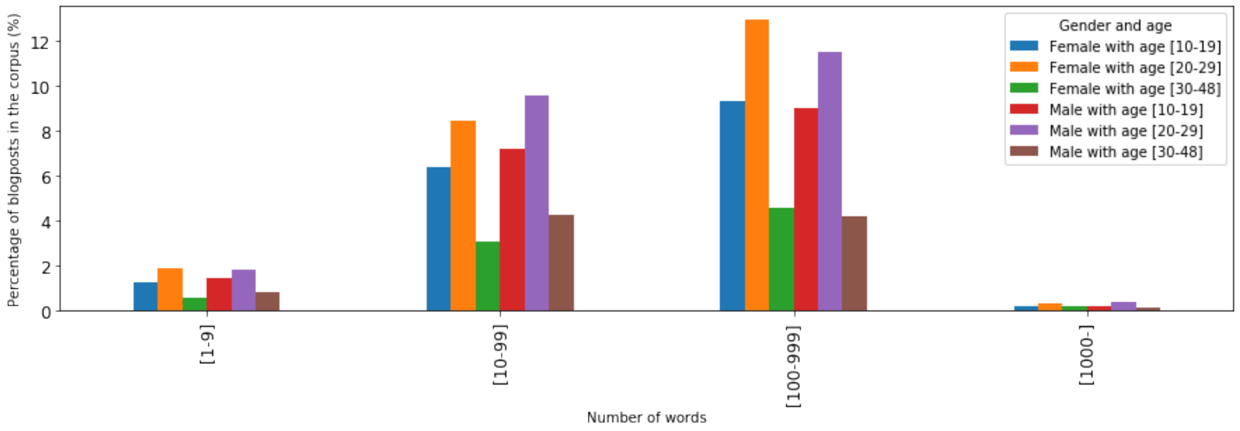

Fig. 4.3. Percentage of blog posts for each range of number of words split by gender and age of authors

Figure 4.2 shows that the corpus is well distributed in term of bloggers' gender and age. Also, figure 4.3 shows that a difference of length exist for blog posts written by bloggers from different genders and ages. Table 4.2 shows more statistics from the corpus.

Tab. 4.1. Statistics from the corpus

Number of blog posts	681,288
Average length of blog posts (in tokens)	192
Size of the vocabulary (in tokens)	1,050,890

4.2. Preprocessing and filtering

Expectedly, this dataset shows some noise and we therefore resorted to few preprocessing methods. For our dataset, we followed these steps:

(1) We deleted all line breaks and tabulation spaces.

(2) We replaced all URL links by a "urlLink" token. Even though the URL links were replaced by a unique token by the creators of the corpus, we found some URL links that were not changed.

(3) We deleted all posts that contained only "urlLink" tokens.

(4) We replaced all links of type "mailto:address@something.something ", which are used to send emails, by "email_urlLink".

These preprocessing steps reduced the number of blog posts by less than 1%.

As mentioned previously, some blog posts weren't written in English. To reduce the number of these posts, we scored a blog post according to a fixed range of Unicode codes from the ASCII table to accept alphanumeric and punctuation and accentuated "e" characters. We also accepted emojis. Hard filtering was applied; whenever an unwanted character appears, we remove the blog post. This scoring method was applied after doing the preprocessing explained above. By so, the number of blog posts is reduced by 5%.

In our experiments:

- NLTK [1] was used in order to tokenize texts and to tag the words
- Flair (Akbik et al. [2018]) was used for document embeddings

[1] `http://www.nltk.org/book/ch05.html`

- Some regular expressions were used in the preprocessing
- PyTorch was used for deep learning models
- LDA was used for topic modeling
- `ScikitLearn`[2] was used for feature-based learning algorithms and TF-IDF features

We split our corpus into 65% as training corpus, 15% as validation corpus and 20% as test corpus for all of the experiments. To analyze the content of the validation and test corpora, we check the percentage of unseen tokens for each blog post. Table 4.2 shows some statistics extracted to analyze the splits described above. By unseen token we mean that the token was not seen during training.

Tab. 4.2. Statistics about the splits made over the corpus

Corpus	#blogs	$\mid V \mid$	Average length of blog posts in number of tokens	Average percentage (%) of unseen tokens per blog post
train	410,103	477,976	192	-
validation	102,526	218,029	191	1.14
test	128,158	245,292	193	1.13

$\mid V \mid$ represents the length of the vocabulary

[2]`https://scikit-learn.org/`

Chapter 5

Results

This chapter presents the results we obtain for the two tasks considered in this thesis: gender and age classification.

Schler et al. [2006] report an accuracy of 80.1% for the gender classification task obtained by applying a 10-fold cross-validation using a Multi-Class Real Winnow algorithm over a set of features partially described. They also report 76.15% for the age classification task. We did our best to reproduce their work without success, due to the lack of description of their features. They use style and content features such as POS, function words and blog specific features: blog words and hyperlinks. For blog words, they mention that they have a list of neologisms, but the details of the list used and the techniques involved are not further described. We contacted the authors who provided us with their list of function words. We do not get similar results using this list. They observe that bloggers in their 10s tend to use more pronouns, blog words and tend to write more words about school such as **maths**, and **homework**. Authors who belong to the second class tend to use more words like **dating**, **bar**, **semester**, **beer**, and **college**. They also observe that bloggers in their 30s and above tend to write longer texts and talk more about **money**, **job**, and **family** with using more prepositions and determiners compared to the authors from other classes.

Later on, the authors report a lower accuracy of 76.10% on the gender classification task in Argamon et al. [2009], which we also find hard to reproduce. This result was obtained by combining both style and content features. As style features, they combine function words with POS and they infer that the most discriminative ones are determiners and prepositions for male bloggers, and pronouns for female ones. The most discriminative content features are words related to technology for males, and words related to relationships and personal life for females. Their results were obtained using a Bayesian Multinomial Regression algorithm. The paper does not specify how the corpus was split for the experiments. Here again, we were not able to reproduce their level of performance, although our best results were close with at best 74.18% on the test corpus. For the age classification, they report an accuracy of 77.70%. They observe that young bloggers (age between 10 and 19) tend to write more without apostrophes which is a discriminative feature for this class and they give the example of **im**, **thats**, and **dont** which should be written as **i'm**, **that's**, and **don't** respectively. The same class of authors tend to use more words like **haha**, **school**, and **wanna** which is a content feature which moderates bias in favor of this class. For the other classes, they observe that the style features are identical. Authors in their twenties and older tend to use prepositions and determiners. However, there are some discriminative features which are part of the content features. Bloggers aged between 20 and 29 tend to use more words such as **apartment**, **office**, and **work**. However, bloggers in their 30s and above tend to use more words like **wife**, **husband** and **daughter**.

By analyzing the methodologies of these two publications, we observe that they both use a carefully designed list of words. In our work, we want to delve deeper into the idea of extracting a carefully designed bag of words and we propose a different data-driven procedure.

In this chapter, we investigate the performances of families of features and their combinations and we compare a number of feature-based machine learning algorithms as well as deep learning ones. We start by discussing the gender classification task results then we explore the age ones. All the experiments presented in this chapter were computed on the validation corpus. At the end of each section, we present the results on the test corpus for the best performing model on the validation data set.

5.1. Gender classification results

This first section of this chapter discusses the results obtained for gender classification task. For this task two classes are considered: male and female. The corpus is well balanced for this task. Consequently, the baseline is 50%.

5.1.1. Style and content features

Families of features used in this chapter were described in Chapters 1 and 3. Some of them are style features and some are content ones. To determine how useful each one is, we use a classifier, feeding it only with said standalone features. Since the classification depends not only on the features but also on the type of the classifier, we also use multiple classifiers such as Gaussian Naive Bayes (GNB), Random Forest (RF), Perceptron, Support Vector Machines (SVM) and Logistic Regression (LR) algorithms. Table 5.1 shows the accuracy of each of the feature families on the validation corpus using a LR classifier since it is the best performing classifier in most of the cases.

Tab. 5.1. Accuracy (%) on the validation corpus for each individual feature using a LR classifier with d being the dimension of the feature vector considered

Type	Feature(s)	d	Accuracy (%)
Content	SW	179	59.86
	FW	467	60.26
	LP	26	57.63
	DP	10	50.93
	EMB	100	62.98
	TF-IDF (unigram of words)	477,976	**71.17**
Style	POS	45	59.21
	VR	5	49.62
	SF	28	57.78
Both	GI	184	61.51
	LIWC	76	59.59

Based on table 5.1 we observe that the best performance we obtain (71.17%) is the one when using LR over TF-IDF vectorizer for unigram of tokens. For Function Words (FW), using the same list as Schler et al. [2006] gives an accuracy of 60.26%. This score differs from theirs due to a different method for text vectorization using this list. Our observation here supports the one made by Argamon et al. [2009] which claims that content features outperform style ones, since the POS feature scores 59.21% using LR.

Combining content and style features can improve the performance. We do this by concatenating multiple feature vectors. For instance, if we concatenate POS and EMB each text would be represented in a vector of 145 elements. For each number of (#) families of features used we add the best performing feature each time based on its performance, on the test set, using a LR classifier that performed the best according to Table E.1, Appendix E. The results of these combinations on the validation corpus are shown in table 5.2.

Tab. 5.2. Greedy search for the best combination using LR with d being the dimension of the features vector considered

# families of features	Feature(s)	d	Accuracy (%)
1	TF-IDF	477,976	71.17
2	+ EMB	478,076	71.54
3	+ GI	478,260	71.77
4	+ FW	478,727	**71.97**
5	+ SW	478,906	71.96

The best gain is less than 1%, compared to the individual performance of TF-IDF with unigram of words, for a significant increase in terms of memory and time for training. By combining multiple features, the accuracy keeps increasing up to a point, then starts dropping. We stopped our experiments at this point. The combination leading to the best performance (till this point) is a combination of several content features, except for GI which includes some style features.

Based on these results we conclude that content features outperform style features. Hereafter, we will focus on extracting a specific type of content features: n-gram features which are commonly used in Natural Language Processing (NLP).

5.1.2. n-gram features

The basic type of content features in NLP for a classification task is the one based on n-grams. In this section, we focus on the performance of this type of features.

During our experiments, we start by evaluating basic models using bag of n-grams which are based on count of each n-gram from a pre-defined list of n-grams. By using this method each text is represented in an m-dimensional vector with m being the number of top n-grams considered. Character-level and word-level n-grams are considered for these features while varying the values of m and n. Considering all unigrams of tokens leads to an accuracy of 68.76% on the validation corpus using count-based vectors, however, this variant is too memory demanding.

As explained in section 3.13, the Discriminative n-gram Indexing (DnI) method affects an n-gram to a specific class and gives it an index showing how discriminative that n-gram is for a specific class. Several experiments are made using this method for different n-gram levels (characters, words, POS tags). For each level, experiments are made for incremental value of n till the accuracy of the chosen classifier drops. This method has a hyperparameter S which is the number of slices considered in the range [50%,100%], in this case, and also the most discriminative n-gram has an index equal to $S - 1$. For this task each text would be represented in a vector of $S \times 2$ elements.

We start our experiments by fixing $S = 200$ using LR classifier while also varying n and the n-gram level. We see that the best performance on the validation corpus is 68.77% obtained using exclusively character n-grams of size 9 (where $n == 9$).

We hypothesize that varying the configuration over Discriminative n-gram Indexing (DnI) (the value of S and the classifier used) will lead to better results. S is a hyper-parameter of DnI which is related to the index affected for each n-gram. First, we study the impact of varying the value of S using a LR classifier. An example of this variation over character level 9-grams, since its the optimal value for n with this feature, is shown in Table 5.3.

Tab. 5.3. Accuracies on the validation corpus of LR classifier over character level 9-grams while varying S

S	1	5	25	50	200
Accuracy (%)	65.93	**69.31**	68.91	68.85	68.77

Based on the results shown in table 5.3, we see that the value of S has an impact on the performances especially when using a small value of S and this variation could lead to a small increase in performances. We see that when generalizing the information by representing the proportion of each gender in just one element ($S = 1$) we obtain the lowest accuracy. The optimal value here is 5.

GNB, MNB, BNB, RF, Perceptron, and LR classifiers are used over DnI. SVM is also used in the cases where we find promising results obtained by LR. We study the performance of each learning algorithm over these features for different values of n, for character-based. The result of this variation is shown in Figure 5.1 with $S = 5$, since it is the optimal parameter.

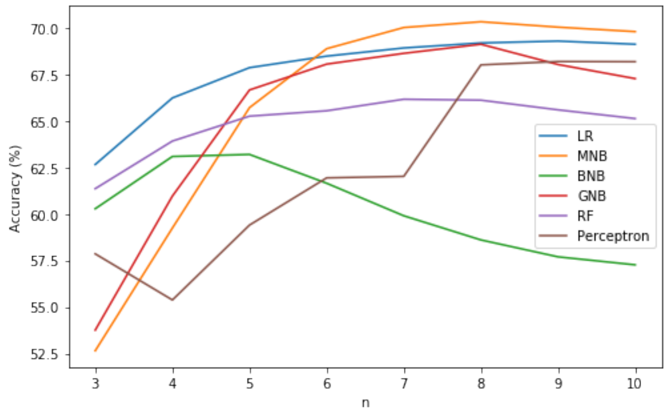

Fig. 5.1. Comparison of different learning algorithms over DnI features with $S = 5$ while varying n for character-based n-grams

Through this figure, we observe that MNB is the best classifier for the values of $n \geq 6$. We also observe that the classifier used has an impact on the results obtained. The best result we obtain on the validation corpus, at this point, using DnI is 70.35% using a MNB classifier over character-level 8-grams with $S = 5$ (from Table E.2 Appendix E). We conclude that several configurations of specific values of n and S using a classifier other than LR obtain better performances.

The result obtained on the validation corpus using DnI features over character-level 9-grams with $S = 5$ is promising, as shown in Table 5.3. This S value is applied for a greedy search to find the combination leading to the best performance. The selection of the classifier is made based on the accuracy of the LR classifier. Just by adding character 8-grams features (second best performing) to the 9-grams features (best performing) drops the results.

As we mentioned, some classifiers works better than other ones on specific features. The outputs of these classifiers could be used as input to a Majority Vote (MV) system. The classifiers are chosen based on their accuracies on the validation corpus. The best performance obtained on the validation corpus using this system is 71.00% using the combination of the best 13 classifiers. More details are shown in Table E.3 Appendix E

Concerning TF-IDF, seeing the results obtained using it in section 1.1 motivated us to try it over our corpus. It is also considered as content feature and gives a weight showing the importance of a word in a text. Several variants of parameters are tried for this type of feature over word and character-level n-grams and for different values and ranges of n using a LR classifier (like shown in Table E.4, Appendix E).

Several parameters matter in TF-IDF, from the results shown in Table E.4 Appendix E, we observe that the number of n-grams considered has a huge impact on the performances of these features. The best results obtained on the validation corpus are those of 6-grams in character level. We study the performance of this feature by using not the totality of the 6-grams but the most frequents. We vary the number of the considered 6-grams and we test its performance on the validation corpus. The result of this variation is shown in figure 5.2.

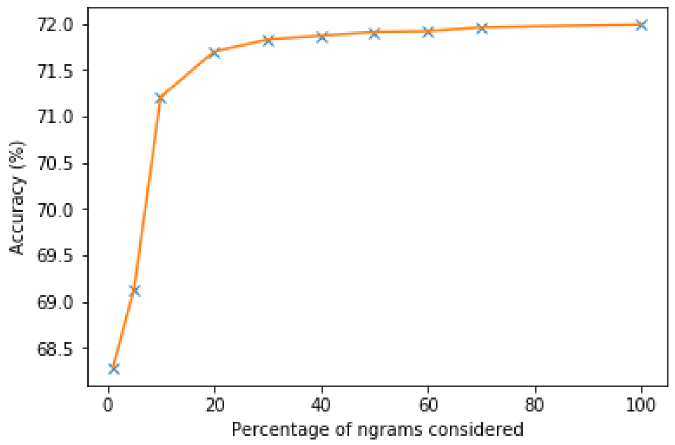

Fig. 5.2. Accuracy (%) of LR classifier on validation corpus with varying the percentage of 6-grams character based

If we consider a small number of n-grams, the performance we obtain is low, as illustrated in figure **??**. We also observe that using just 50% of the n-grams leads to lowering the performance by 0.08% compared to the one using 100% of the n-grams. However, it is less time and hardware demanding.

The best result we obtain on validation corpus using TF-IDF is 72.41% using character level n-gram with $n \in \{3,4,5\}$.

Similarly to what we did for DnI, a MV system is applied over the outputs of LR classifiers trained on TF-IDF features. Here again, we apply a greedy search method over the classifiers trained on TF-IDF. The best result obtained here is 73.61% using the combination of the best 13 classifiers from Table E.4 Appendix E. More details of the experiments are shown in Table E.5 Appendix E.

These two methods, TF-IDF and DnI, could be directly compared to each other since they use the same information but computed differently. The comparison is shown in Table 5.4 when varying the value of n for character and word n-gram levels.

Tab. 5.4. Comparison of DnI and TF-IDF accuracies (%) on the validation corpus for gender classification using LR over some examples of features with d being the dimension of the features vector considered

Feature	DnI ($S = 5$)		TF-IDF	
	Accuracy (%)	d	Accuracy (%)	d
character 3-grams	62.67	**10**	**69.48**	107,916
character 9-grams	69.31	**10**	**71.13**	55,857,568
word 1-gram	67.95	**10**	**71.17**	477,976
word 2-grams	67.36	**10**	**67.52**	13,410,706

From table 5.4, we observe that TF-IDF outperforms DnI using a LR classifier in all of the configurations studied. This difference is conceptually due to the fact that DnI considers just the number of times an n-gram appears in the collection. And, TF-IDF takes into account the number of texts in which an n-gram appears. We observe also a huge difference between the dimensions of vectors used to represent the text for each of the two methods. For example, in the case of word bigrams, DnI represents texts in vectors of 10 elements and TF-IDF in vectors of 13,410,706 elements. Nevertheless, TF-IDF obtains just 0.16% more than DnI.

Seeing the promising results obtained using a Majority Vote (MV) system over TF-IDF and DnI which obtained (at its best performances on validation corpus) 73.61% and 71.00% respectively. We try MV system using outputs of different classifiers some of them trained on TF-IDF and other on DnI. The best performance we obtain on the validation corpus here is 74.02% using the combination of 21 classifiers: 12 of them trained on TF-IDF and 9 on DnI. More details of these experiments are shown in Table E.6 Appendix E.

5.1.3. Topic modelling

From the beginning of this chapter we studied several content and style features and we observed that content features are better. From Table 5.1, we see that General Inquirer (GI), which includes a combination of both and was explored in 3.4, gives good performance. This type of feature in particular uses a list of words split by their domain. This information is more studied in this section using topic modeling features.

Topic modeling is a method used to extract the proportion of each topic in certain text. Several representations for a text can be done using the topic modeling method. A text could be represented using a vector of probability for each topic. It could also be represented by applying a one-hot enconding over the probability vectors. The number of the topics extracted from the train dataset can have an impact on the prediction results, that is why we vary this parameter. The performance on the validation corpus of the two representation methods while varying the number of topics is shown in table 5.5. Based on these results, we infer that the one-hot encoding representation of topics does not provide as much information to the classifier as the other method. Thus, increasing the number of topics does not always lead to better performances. We also vary the classifiers to see the impact of this variation. The detailed results are shown in Table E.7 and Table E.8, Appendix E and still the LR classifier is the best performing one. We conclude that this type of information is not very discriminative to predict the author's gender.

Tab. 5.5. Comparison of LR results using two representation ways topic modeling features for gender classification

Number of topics	50	100	150	200
Accuracy (%) of using probabilities representation	60.00	59.37	59.57	59.49
Accuracy (%) of using one-hot encoding representation	59.06	59.17	59.47	59.64

Since the best performance obtained using this features is 60.00% using LR over probabilities vectors with 50 topics, we analyzed the information behind this result. The most informative topic about female authors is the one related to shopping which contains words like buy, shirt, dress, and pants. More topics show bias in favor of female authors which are related to food, cooking, events and relationships. The topic which provides more information about male authors is the one related to technology which regroups words such as computer, system, program, and Windows. Other topics are more biased in favor of male are related to famous persons, player, politics and business.

5.1.4. Deep learning methods

All the previous sections focused on feature-based learning and our best result is 74.91% using MV system over the predictions of selected classifiers, see the end of subsection 5.1.2. In the current section, we focus on using a Feed Forward Neural Network (FFNN) over features from the previous sections, as well as deep-learning methods.

5.1.4.1. *Feed Forward Neural Network (FFNN)*

Many parameters can impact the performance of a Feed Forward Neural Network (FFNN) such as the learning rate, the number of epochs, the optimizer, and the batch size. We consider several variants of these parameters and we try their combinations while applying an early stopping method.

From section 5.1.1, we saw that EMB features gives the best trade-off between performance and memory usage. We decide to use this feature to see how much a FFNN could learn from it with several combinations of parameters. We computed more than 30 experiments for different:

- numbers of hidden layers $\in \{1, \ldots, 5\}$
- number of epochs $\in \{10, 20, 30, 40, 50, 100\}$
- learning rates $\in \{0.1, 0.01, 0.001, 0.0001\}$
- optimizers such as Adagrad, Adadelta, RMSprop, Adam, and SGD

When we use just EMB, the best result we obtain is 64.92% (on the validation dataset using 1 hidden layers composed of 128 hidden units, number of epochs equals to 100, learning rate equals to 0.0001 using an Adam optimizer with a batch size of 1,024).

When we use FFNN over several combinations of features from table 5.1 while applying a grid search method to obtain the best parameters, the best result is 67.28% (over the validation dataset by combining 5 features using the parameters: 1 hidden layer composed of 2048 hidden units, learning rate equals to 0.001, number of epochs to 50 and Adam as an optimizer with batch size equals to 1,024). The same combination obtained 64.21% using a logistic regression.

Based on these presented results we can see that a FFNN can learn more information than other classifiers used.

5.1.4.2. *Recurrent Neural Network (RNN)*

The previous models used pre-computed features. The model we focus on here is a deep-based model which learns to extract features from the texts in order to classify them. We try to vary some parameters to see the performance of this type of NN on our data. GloVe[1], a pre-trained model for word embeddings, is used to create the lookup table for our vocabulary. We start by trying the performance of an unidirectional RNN and then the bidirectional RNN with varying the number of hidden states and the number of recurrent layers. An early stopping method is applied to the RNN. We compute 54 configurations, varying:

- BS (batch size) $\in \{32,64\}$
- #HL (number of hidden recurrent layers) $\in \{1,2,3\}$
- #HS (number of hidden states per hidden layer) $\in \{10,50,100\}$
- LR (value of learning rate) $\in \{0.01,0.001\}$

Tables F.1 and F.2 show the performances of the RNN with unidirectional and bidirectional recurrent layer(s) respectively with LR=0.01. Tables F.3 and F.4 show the performances for LR=0.001.

We observe that the value of LR has a huge impact on the performance of the RNN, While values of #HL and #HS have a small impact on the results. We also see that the bidirectional RNN outperforms the unidirectional because it is learning much more information from the texts since it is browsing them in two ways.

The best results are obtained using a bidirectional RNN, with the parameters LR=0.001, BS=64, #HL=2, and #HS=10, with accuracies of 63.29% over the validation dataset. Comparing the results of this model with the previous one, we infer that it is not learning to extract the right features from the texts.

[1]`https://nlp.stanford.edu/projects/glove/`

5.1.4.3. *Recurrent Convolutional Neural Network (RCNN)*

Recurrent Convolutional Neural Network (RCNN) is another deep-based model which learns to extract features from the text. Several parameters have impact on this type of NN such as:

- #HS is the number of hidden states in the recurrent layer
- BS is the batch size
- LR is the value of learning rate

16 variants of these parameters were tried with:

- LR $\in \{0.01, 0.001\}$
- #HS $\in \{5, 10, 50, 100\}$
- BS $\in \{32, 64\}$

The results of these configurations are shown in Table F.5 and Table F.6 for different values of LR. The best result obtained over the validation corpus is 68.28% for a LR value of 0.001. Based on these results and the tables mentioned above, we see that the value of LR has an impact on the RCNN performances. However, the other parameters do not. RCNN gives better performances than RNN because it represents a words based on its representation from a pre-trained model and its context.

The best result is obtained with the parameters LR=0.001, BS=64, and #HS=100, with accuracies of 68.28% over validation dataset. RCNN also outperforms FFNN so we conclude that it is learning the right features.

Even if RCNN outperforms the other 2 deep learning models, it did not get better results than those of TF-IDF and we stick with best configuration of using a MV system over classifiers trained on TF-IDF features and others trained on Discriminative *n*-gram Indexing features which scored 74.02% on the validation corpus. Thus, we conclude that a features-based model outperformed the deep models for this task. All the results shown so far were obtained on the validation corpus like we already explained at the beginning of the chapter. Now, we must evaluate the best performing models on the test corpus. Table 5.6 shows the performance on validation and test corpora of the best performing model on the validation corpus for each feature.

Tab. 5.6. Accuracies (%) on validation and test corpora of the best performing models

Model	Validation	Test
LR trained on TF-IDF using {1,2}-grams of tokens	71.71	71.83
LR trained on TF-IDF using {3,4,5}-grams of characters	72.41	72.67
MNB trained on DnI features using 8-grams of characters	70.34	70.35
MV using classifiers trained on DnI	71.00	71.08
MV using classifiers trained on TF-IDF	73.61	73.78
MV using classifiers trained on TF-IDF and others on DnI	**74.02**	**74.18**
FFNN	67.28	67.02
RNN	63.29	63.46
RCNN	68.28	68.32

LR stands for Logistic regression, MV for Majority Vote, DnI for Discriminative n-gram Indexing, MNB for Multinomial Naive Bayes

The best model obtained for this first task scores 74.02% on the validation corpus (from section 5.1.2) and 74.18% on the test corpus. We conclude that our model is good at generalizing the prediction since the difference of performances between validation and test is tiny.

5.2. Age classification results

The second section of this chapter shows some of the results obtained for age classification task. For this classification task, the corpus is not well balanced and a majority class prediction leads to an accuracy of 46%.

5.2.1. Style and content features

Here we start by computing the performances of families of features, from table 5.1, to see which one is best performing for this task. Since the classification depends not only on the feature but also on the type of classifier, we also use multiple classifiers such as GNB, RF, Perceptron, SVM and LR. The results are shown in Table G.1.

Here again, TF-IDF obtains the best performances scoring 67.95% on the validation corpus using a LR classifier over unigrams of tokens. Our observations here also endorse those made by Schler et al. [2006] and Argamon et al. [2009], since the content features outperforms the style features.

A greedy method is followed in order to see which combination of features obtains the best accuracy. Since LR obtains the best performances over TF-IDF (unigrams of tokens), we use it in this experiments. Table G.2 Appendix G shows the accuracy for each number of families of features. Combining 5 features leads to the best accuracy scoring 68.64% on the validation corpus. Combining both content and style features leads to a small increase in the results since one of the features combined is a style-based feature (POS).

Here after, we will focus on extracting more content features since it outperforms the style ones.

5.2.2. *n*-gram features

In the previous subsection, we saw that using LR over a combination of features leads to the best performances. In the current section we focus on studying the performances of LR over *n*-gram features. We start our experiments by computing bag of *n*-grams features. These features could be used to represent each text in a count-based vector. By considering all unigrams of tokens we obtain a score of 66.48% on the validation corpus. Here after we focus on studying the performances of several configurations for TF-IDF and Discriminative *n*-gram Indexing (D*n*I).

In section 5.1.2, we evaluated TF-IDF and Discriminative *n*-gram Indexing (D*n*I), each one individually. Then we compared them and finally we combined them. For this section we follow the same steps.

Discriminative *n*-gram Indexing (D*n*I), explored in section 3.13, which assign an *n*-gram to a class and gives it an index showing how much is that *n*-gram discriminative for that class, is used for this classification task. Several variants of this method are considered in our experiments in different *n*-grams levels (character, word, POS tags) with fixing $S = 200$. S is the hyperparameter of D*n*I which represents the number of splits considered in the range of [33.34%,100%] (for this task) and each text is represented in a vector of $S \times 3$ elements. The best result is 66.53% using LR over 8-grams character based features.

A variation of S could lead to a small increase in the performances. Table 5.7 shows an example of varying the value of S over character level 8grams.

Tab. 5.7. Accuracies (%) on the validation corpus of LR classifier over character level 8-grams while varying S

S	1	5	25	50	200
Accuracy (%)	63.04	**67.28**	66.72	66.70	66.53

Based on Table 5.7, we see that this variation leads to an increase in the performances. More detailed results of varying n, S and the classifiers over DnI features for different n-grams levels is shown in Table G.3, Appendix G.

The best performance obtained on the validation corpus while varying S, n and the classifier used is 67.28% using $S = 5$ and 8-grams of character with a LR classifier.

As $S = 5$ is the optimal parameter, this optimal value is used for a greedy search method in order to obtain the best combination of DnI features leading to the best performances. Just by combining character 9-grams features (best performing) and character 8-grams features (second best performing) the accuracy drops scoring 67.21%.

Several classifiers are performing well individually with certain features. The output of those classifiers could be used for a Marjory Vote (MV) system. In order to determine the best combination of outputs leading to the best performances, a greedy search method is applied using the same optimal value of $S = 5$. The best performance obtained on the validation corpus using this system is 68.14% by combining the outputs of 5 classifiers. The detailed results are shown in Table G.4 Appendix G.

TF-IDF gave the best individual performing results for gender classification task. We reuse it for age classification. Several variants of parameters are tested for this type of feature for word and character level n-grams and for different values and ranges of n like shown in Table G.5, Appendix G. The best result obtained on the validation corpus here is 72.20% using LR over {5,6,7}-grams in a character-base.

Using a MV system over the output of LR classifiers trained on TF-IDF features increased the performances for the gender classification task. The same system is used for age classification by doing a greedy search method to determine the combination of outputs leading to the best performances. Just by adding the outputs of 2 classifiers to the outputs of best performing classifier (trained over character {5,6,7}-grams) drops the results and obtains 72.11% on the validation corpus. For this task, combining the outputs of several LR trained on TF-IDF features does not improve the results like it was the case of the gender classification task.

As argued in section 5.1.2, TF-IDF could be directly compared to DnI since they use the same kind of features. The comparison is shown in Table 5.8.

Tab. 5.8. Comparison of DnI and TF-IDF accuracies (%) on the validation corpus for age classification using LR over some examples of features with d being the dimension of the features vector considered

Feature	DnI (S=5)		TF-IDF	
	Accuracy (%)	d	Accuracy (%)	d
character 3-grams	51.24	**15**	**64.30**	108,144
character 8-grams	67.28	**15**	**71.46**	34,657,850
word 1-gram	62.39	**15**	**67.95**	477,837
word 2-grams	64.93	**15**	**67.36**	13,433,218

The results from table 5.8 show that TF-IDF outperforms DnI with a significant difference of scores. The difference of their scores when using bigrams in word level is 2.43% when DnI uses only a 15 dimensional vector to represent the texts while TF-IDF uses vectors of 13,433,218 elements.

5.2.3. Topic modelling

These features are reused for the age classification to see if we can capture whether authors from different ages write about different topics. A text could be represented in two ways. The first one is a vector of probability of each topic. The second one is a one-hot encoding of the first method. The result of comparing these two methods using LR classifier is shown in Table 5.9.

Based on table 5.9, topic modeling features provide some information to the classifier which leads to a better results than the baseline but it does not outperform other n-gram features results. The classifiers used over this type of feature were varied and the detailed results are shown in Tables G.7 and G.8, Appendix G, for using probabilities vectors and one-hot

Tab. 5.9. Comparison of LR results on the validation corpus using two representation ways topic modeling features for age classification

Number of topics	50	100	150	200
Accuracy (%) of using probabilities representations	44.16	46.00	46.00	**46.19**
Accuracy (%) of one-hot encoding representation	43.35	43.56	44.57	45.37

encoding respectively. The best result we obtain using this features for this task is 54.92% using a RF classifier with 100 topics.

For the age classification task more than 10,000 classifiers were generated. In this section we showed the most important ones in term of information provided to us. These classifiers showed the level of complexity of identifying the age of the author based on the content of his text or the style of writing. For this task, deep learning models were not computed since they did not outperform feature-based models in the gender classification task. Table 5.10 shows the performance on validation and test corpora of the best performing model on the validation corpus for each feature.

Tab. 5.10. Accuracies (%) on validation and test corpora of the best performing models

Model	Validation	Test
LR trained on TF-IDF using {5,6,7}-grams of characters	**72.20**	**72.04**
LR trained on TF-IDF using {1,2}-grams of tokens	71.21	71.20
LR trained on DnI features using 8-grams of characters	67.28	66.92
MV using classifiers trained on DnI	68.14	67.98

LR stands for Logistic Regression, MV for Majority Vote, DnI for Discriminative n-gram Indexing

The best results obtained is 72.20% on the validation corpus leading to an accuracy of 72.04% on the test corpus. We can see that this system is good at generalizing the predictions since the gap between validation's accuracy and test's accuracy is small.

In this fifth chapter of this thesis, we showed many experiments we did and we found interesting to show. The best performances were 74.18% and 72.04% for the gender and age classification tasks respectively over the test dataset.

Chapter 6

Analysis

In this masters thesis, two tasks were considered: gender classification and age classification. Many features were approached in order to observe which ones are more deterministic and provide more information for each task. The results were explored in Chapter 5. Following the same evaluation metric as the previous chapter, the accuracy is considered in this chapter.

The results of gender classification task were presented in section 5.1. Based on these results we infer that stylometric features do not provide enough information in order to predict the gender of the author. For example, POS features, provide information about the type of the tokens used, gives an accuracy of 59.21% which is almost 10% higher than the majority-class classifier. Another example which is the stylometric features, explored in section 3.11, studies the tendency of uses of certain features related to the style of writing and obtains a performance of 57.78%.

Moreover, content-based features provide more information comparing to style-based ones in order to predict the gender of the author as shown in the results explored in section 5.1. The Discriminative n-gram Indexing (DnI) features support also our observation by scoring more than 70% of accuracy by using features based on character levels. In addition, generalizing more the information of n-gram features, by using all n-grams seen during the training, provides more information to the classifiers and gives more information about which n-grams are more biased in favor of certain class. Applying a Majority Vote (MV) over the outputs of classifiers trained on DnI also improves the result and scores 71.00% over the validation dataset and 71.08% over the test corpus. An overview over the lengths, number of tokens, of the misclassified blog posts by this MV system is shown in figures 6.1 and 6.2 for validation and test corpora respectively.

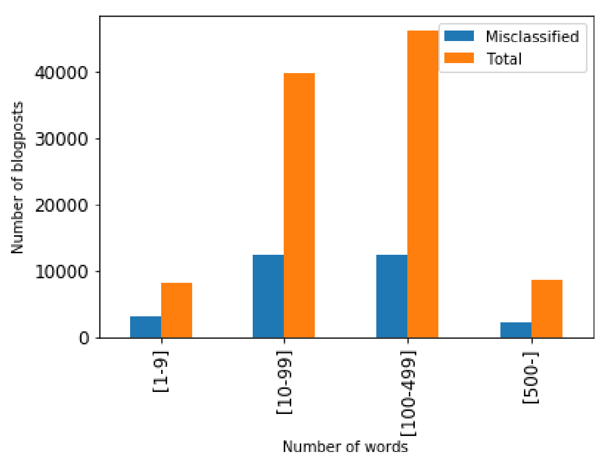

Fig. 6.1. Number of misclassified blog posts from the validation corpus split by the majority vote system using classifiers trained on discriminative n-gram indexing

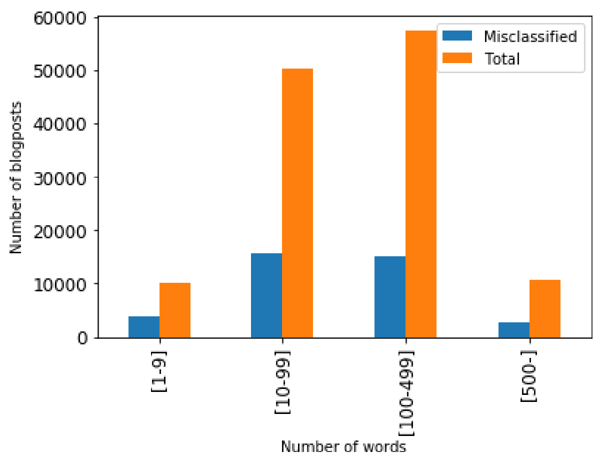

Fig. 6.2. Number of misclassified blog posts from the test corpus split by the majority vote system using classifiers trained on discriminative n-gram indexing

Almost 40% of texts having number of tokens less than 10 are misclassified for both datasets. For the rest of ranges of number of tokens, an average of 25% of the texts are misclassified. Therefore, the number of tokens have a small impact on the classification.

The discriminative n-gram indexing obtained good results comparing to basic n-gram features for the task of gender classification due to its consideration for all n-grams seen during training. However, it is important to see whether the number of unseen tokens, during training, have an impact on the misclassification or not. To do so, we analyze the number of unseen 8-grams character using a Multinomial Naive Bayes (MNB) classifier with $S = 5$, since its the optimal configuration for these features scoring an accuracy of 70.35%. This analysis is shown in Figure 6.3.

The highest percentage of misclassified texts shown in figure 6.3 is the one of texts composed of 100% of unseen 8-grams. However, these texts represent just 0.02% of the test corpus so they do not have a huge impact on the performances of the model. In addition to that, misclassified texts composed of 100% of seen n-grams represent 6.44% of the test corpus. After studying the content of these texts, we see that an average of more than 90% of their 8-grams are not affected to their true class profile. This is a weak element for DnI method and solving it is considered as part of the future works. Through figure 6.3 we infer also that

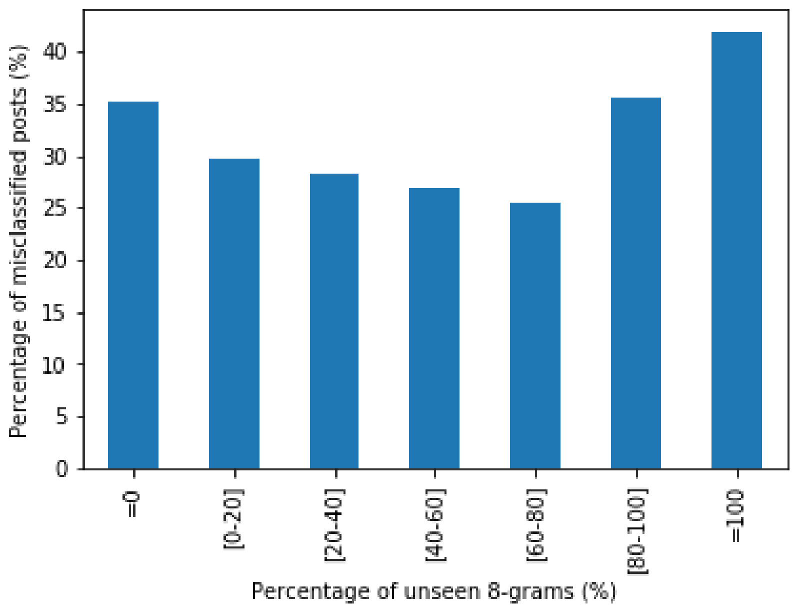

Fig. 6.3. Percentage of misclassified posts from test corpus for each range of percentages of unseen 8-grams during training

increasing the percentage of unseen n-grams does not always lead to increase the percentage of misclassified texts.

For this task, several configurations of TF-IDF were computed and the best result obtained 72.41% using a LR classifier over {3,4,5}-grams of characters. Using the outputs of several classifiers trained on TF-IDF as input to a Majority Vote (MV) system leads to at its best performances to 73.61% on the validation corpus by combining the outputs of 13 classifiers.

Combining the outputs of specific classifiers based on their accuracies and putting them through a MV system leads an increase in the results. For this task, the best accuracy was obtained using this system over the outputs of classifiers trained on TF-IDF and others on DnI features and we obtained 74.02% and 74.18% over validation and test datasets respectively. Figures 6.4 and 6.5 show an overview over the number of misclassified blog posts regrouped by ranges of number of words for validation and test datasets respectively.
Through figures 6.4 and 6.5 we can see that the highest percentages of misclassified texts is the one of short texts (having less than 10 tokens). 35% of these texts were misclassified for both datasets. For the rest of ranges an average of almost 25% of texts were misclassified. An analysis was made for many configurations of classifiers over word and character based

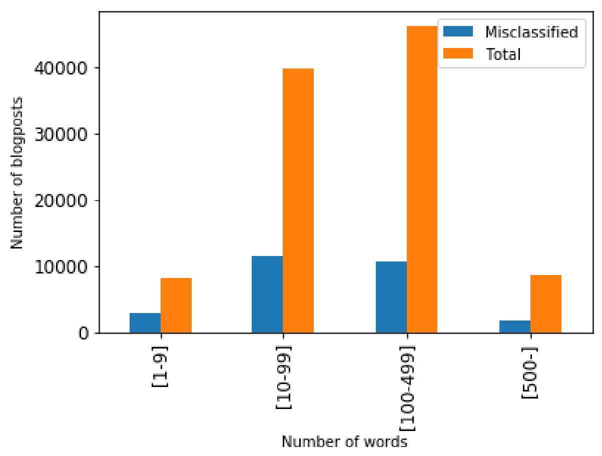

Fig. 6.4. Number of misclassified blog posts from the dev corpus by the best performing system for gender classification regrouped by ranges of number of words

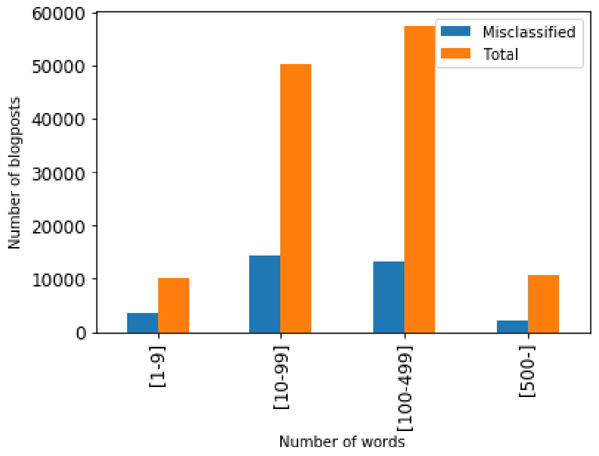

Fig. 6.5. Number of misclassified blog posts from the test corpus by the best performing system for gender classification regrouped by ranges of number of words

features in order to solve the misclassification of short texts. However, the problem is still remaining. We are considering solving it as one of the future works.

By analyzing the content of the misclassified texts of the best model for gender classification we see that:

- 6% of these texts have less than 10 tokens. An average of almost 50% of their tokens are the "urlLink" token. We infer that more cleaning should be done for the corpus since it is hard to classify a text which is composed of "urlLink" and another token.
- Some texts are ambiguous which even a human being cannot make an educated guess whether it is likely to be written by which gender. This ambiguity consists of lack of information which could be extracted from the content. Some of these texts are just composed of blog words such as a sequence of laughs "HAHAHA" but some of them were written by male bloggers and some by female ones.
- Other texts were misclassified because of the author's writing style being neutral or ambiguous.

Content-based features outperform the stylometric-based ones for the age classification task as well, as shown in the results explored in section 5.2. Combining multiple content and style features leads to a maximum performance of 68.64%. More content based features were studied. n-gram based features were used in order to study the impact of their information for age classification. DnI was also used for this task and gave at best an accuracy of 68.14% by using a MV system over the output of 5 classifiers trained on these features.

Using a LR classifier with TF-IDF features gave the best performances in terms of accuracy like explored in section 5.2. Applying this method led to scores almost similar to those cited in the state of art using the same method. For age classification task, the best individual performance scored 72.20% over character level n-grams for $n \in \{5,6,7\}$.

The results of the best performing TF-IDF for both tasks are analyzed to see which tokens provided more information to our classifier. This analysis is made through seeing tokens with the most information gain based on TF-IDF. After analyzing the most weighted tokens for each class for both tasks, we see that many miss-spelled words give a moderate bias in favor of one of the classes. We observe also that a significant number of pseudonyms provide information on who wrote the text. For example, some blog words could be more discriminative for one of the classes like "xoxo", which means kisses, were more used by female authors. Other words provide more information about a class which are the repetition of certain character in the token, for example "OMGGG" is different from "OMG" and each one could be used by a different class. As a solution for this, a normalization for the blog posts could be used and we are considering it for a future work.

For both tasks, having a first intuition that several topics are more discussed by one of the gender or one of the ages classes is correct. However, this information is not that discriminative enough to determine the gender or the age of the author. At its best performance it obtained 60.00% for gender classification (see section 5.1.3) and 54.92% for age classification (see section 5.2.3).

Based on these observations we infer that using a carefully designed list of bag-of-words extracted based on the topics could lead to better results and we are considering it as a future work axes. This solution could also solve the problem of mis-classifying short texts.

Deep learning models were tested for gender classification task but they did not outperform other models which are feature-based models.

Conclusion

Authorship Attribution is an NLP research topic which exists since the 1960s. Its main goal is to identify the author of a document. This is the main topic we worked on in this thesis. It is also seen as a classification task for texts. Two classification tasks were considered: gender and age.

Several variants of features and learning algorithms presented in this thesis were used for those tasks. Two types of features were compared; content-based and style-based features. The combination of both was also considered.

Our conclusion for the gender classification task is that content features works better for this task and the difference between its performance and the best one of style features is significant. Combining several content features leads to an increase in the results and just by adding a style feature the accuracy drops. The best result obtained for this task is 74.18% using a Majority Vote system over the combination of outputs of classifiers trained on TF-IDF features and others trained on Discriminative n-gram Indexing (DnI) features (the method explored in section 3.13). Topic modeling features, explored in section 3.10, provides some information about the texts and the best accuracy obtained using this features is 60.00%. Several deep learning models were tried for this task. The best result obtained on the validation dataset is 68.25% using a recurrent convolutional neural network. We infer here that deep learning models do not learn the right features for this type of classification.

We conclude that predicting the gender of the author is much more complicated than we thought it would be. The difficulty is due to the overlapping of features extracted from texts written by male and female bloggers. Thus, small details matter since using some words which are more likely to be used by the other gender gets the text closer to the other profile.

Our conclusion for the age classification task is that content-based features work better than style-based features. Combining both leads to an increase of the results. The best result was obtained using a LR classifier over TF-IDF features. For this task it scored at its performances 72.04% over the test corpus. Topic modeling features were also tried for this task. The information provided by these features is not that discriminative for the classifiers to be able to get a high score for the classification. The best score obtained using this feature is 54.92%.

For this task, we conclude that there is too much overlapping between texts written by bloggers from different ages. This overlapping is more noticeable for texts written by authors in their 20s and 30s.

Through the results and the analysis made in this thesis, we see that the two considered tasks are complex. Identifying the gender and the age of the author based on the content of the texts or the writing style is possible under certain condition. For instance, our models were not performing enough for short texts since there is not much information to extract from those texts.

Deep learning models were tested for gender classification task but they were not as good as extracting features and training feature-based models over them.

For our future work we are considering going in depth to solve the problem of misclassification of short texts. We are also considering applying pre-trained models, like BERT, that have shown to obtain good results for multiple and different NLP tasks. Furthermore, we are considering testing our features over other benchmarks. Finally, we will try to work more on the discriminative n-gram indexing method in order to make it better for multi-class classification tasks.

Bibliography

Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649, 2018.

Waheed Anwer, Imran Bajwa, Muhammad Choudhary, and Shabana Ramzan. An empirical study on forensic analysis of urdu text using lda based authorship attribution. *IEEE Access*, 7:3224–3234, 01 2019. doi: 10.1109/ACCESS.2018.2885011.

Shlomo Argamon, Moshe Koppel, James Pennebaker, and Jonathan Schler. Automatically profiling the author of an anonymous text. *Communications of the ACM*, 52(2):119–123, 2 2009. ISSN 0001-0782. doi: 10.1145/1461928.1461959.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null):993–1022, March 2003. ISSN 1532-4435.

Leo Breiman. Bagging predictors, 1994.

Miguel Ángel Álvarez Carmona, Adrián Pastor López-Monroy, Manuel Montes y Gómez, Luis Villaseñor Pineda, and Hugo Jair Escalante. Inaoe's participation at pan'15: Author profiling task. In *CLEF*, 2015.

Na Cheng, R. Chandramouli, and K. P. Subbalakshmi. Author gender identification from text. *Digit. Investig.*, 8(1):78–88, July 2011. ISSN 1742-2876. doi: 10.1016/j.diin.2011.04.002. URL `http://dx.doi.org/10.1016/j.diin.2011.04.002`.

Fouryears. The mystery of early stopping. URL `http://fouryears.eu/2017/12/06/the-mystery-of-early-stopping/`. Accessed: 2019-09-19.

Helena Gomez Adorno, Ilia Markov, Grigori Sidorov, Juan-Pablo Posadas-Durán, Miguel Sanchez-Perez, and Liliana Chanona-Hernandez. Improving feature representation based on a neural network for author profiling in social media texts. *Computational Intelligence and Neuroscience*, 2016:13 pages, 10 2016. doi: 10.1155/2016/1638936.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

Gilad Gressel, S. Thara, A Aravind, and Prabaharan Poornachandran. Ensemble learning approach for author profiling notebook for pan at clef 2014. 2014.

Andreas Grivas, Anastasia Krithara, and George Giannakopoulos. Author profiling using stylometric and structural feature groupings. In *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum, Toulouse, France, September 8-11, 2015.*, 2015. URL `http://ceur-ws.org/Vol-1391/68-CR.pdf`.

Robert Gunning. *The Technique of Clear Writing.* 1952.

Graeme Hirst, Xuan Le, Ian Lancashire, and Regina Jokel. Longitudinal detection of dementia through lexical and syntactic changes in writing: a case study of three British novelists. *Digital Scholarship in the Humanities*, 26(4):435–461, 05 2011. ISSN 2055-7671. doi: 10.1093/llc/fqr013. URL `https://doi.org/10.1093/llc/fqr013`.

Magdalena Jankowska, Vlado Keselj, and Evangelos E. Milios. Cng text classification for authorship profiling task. In *CLEF 2013 Evaluation Labs and Workship*, 2013.

Yoon Kim. Convolutional neural networks for sentence classification, 2014.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. Computational methods in authorship attribution, 2008.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2267–2273. AAAI Press, 2015. ISBN 0-262-51129-0. URL `http://dl.acm.org/citation.cfm?id=2886521.2886636`.

G. Harry Mc Laughlin. Smog grading – a new readability formula. 1969.

Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014. URL `http://arxiv.org/abs/1405.4053`.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, Dec 1989. ISSN 0899-7667. doi: 10.1162/neco.1989.1.4.541.

Ilia Markov, Helena Gomez Adorno, Juan-Pablo Posadas-Durán, Grigori Sidorov, and Alexander Gelbukh. Author profiling with doc2vec neural network-based document embeddings (preprint version). pages 117–131, 08 2017. doi: 10.1007/978-3-319-62428-0_9.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

Frederick Mosteller and David L Wallace. Inference in an authorship problem: A comparative study of discrimination methods applied to the authorship of the disputed federalist papers. *Journal of the American Statistical Association*, 58(302):275–309, 1963.

Rebekah Overdorf and Rachel Greenstadt. Blogs, twitter feeds, and reddit comments: Cross-domain authorship attribution. *Proceedings on Privacy Enhancing Technologies*, 2016, 07 2016. doi: 10.1515/popets-2016-0021.

James W. Pennebaker, Ryan L. Boyd, Kayla Jordan, and Kate Blackburn. The development and psychometric properties of liwc2015. 2015. doi: 10.15781/T29G6Z. URL `http://hdl.handle.net/2152/31333`.

A. Rocha, W. J. Scheirer, C. W. Forstall, T. Cavalcante, A. Theophilo, B. Shen, A. R. B. Carvalho, and E. Stamatatos. Authorship attribution for social media forensics. *IEEE Transactions on Information Forensics and Security*, 12(1):5–33, Jan 2017. ISSN 1556-6013. doi: 10.1109/TIFS.2016.2603960.

Frank Rosenblatt. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. 1962.

Upendra Sapkota, Steven Bethard, Manuel Montes y Gómez, and Thamar Solorio. Not all character n-grams are created equal: A study in authorship attribution. 2015.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James Pennebaker. Effects of age and gender on blogging. *Proceedings of 2006 AAAI Spring Symposium on Computational Approaches for Analyzing Weblogs*, 2006.

E. A. Smith and R. 1. Senter. Automated readability index. 1967.

Efstathios Stamatatos. A survey of modern authorship attribution methods. *J. Am. Soc. Inf. Sci. Technol.*, 60(3):538–556, March 2009. ISSN 1532-2882.

Philip Stone, Robert Bales, J. Namenwirth, and Daniel Ogilvie. The general inquirer: A computer system for content analysis and retrieval based on the sentence as a unit of information. *Behavioral Science*, 7:484 – 498, 10 2007. doi: 10.1002/bs.3830070412.

Towardsdatascience. Understanding bidirectional rnn in pytorch. a. URL `https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66`. Accessed: 2019-09-17.

Towardsdatascience. Counting no. of parameters in deep learning models by hand. b. URL `https://towardsdatascience.com/counting-no-of-parameters-in-deep-learning-models-\by-hand-8f1716241889`. Accessed: 2019-06-06.

Towardsdatascience. The kernel trick in support vector classification, c. URL `https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f`. Accessed: 2019-09-13.

Özlem Uzuner, Boris Katz, and Thade Nahnsen. Using syntactic information to identify plagiarism. In *Proceedings of the Second Workshop on Building Educational Applications Using NLP*, EdAppsNLP 05, pages 37–44, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1609829.1609836`.

P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, Oct 1990. doi: 10.1109/5.58337.

Appendix A

List of stopwords

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

Appendix B

List of function words

'a', 'about', 'above', 'according', 'accordingly', 'actual', 'actually', 'after', 'afterward', 'afterwards', 'again', 'against', 'ago', 'ah', "ain't", 'all', 'almost', 'along', 'already', 'also', 'although', 'always', 'am', 'among', 'an', 'and', 'another', 'any', 'anybody', 'anyone', 'anything', 'anywhere', 'are', "aren't", 'around', 'art', 'as', 'aside', 'at', 'away', 'ay', 'back', 'be', 'bear', 'because', 'been', 'before', 'being', 'below', 'beneath', 'beside', 'besides', 'better', 'between', 'beyond', 'bid', 'billion', 'billionth', 'both', 'bring', 'but', 'by', 'came', 'can', "can't", 'cannot', 'canst', 'certain', 'certainly', 'come', 'comes', 'consequently', 'could', "couldn't", 'couldst', 'dear', 'definite', 'definitely', 'despite', 'did', "didn't", 'do', 'does', "doesn't", 'doing', "don't", 'done', 'dost', 'doth', 'doubtful', 'doubtfully', 'down', 'due', 'during', 'e.g.', 'each', 'earlier', 'early', 'eight', 'eighteen', 'eighteenth', 'eighth', 'eighthly', 'eightieth', 'eighty', 'either', 'eleven', 'eleventh', 'else', 'enough', 'enter', 'ere', 'erst', 'even', 'eventually', 'ever', 'every', 'everybody', 'everyone', 'everything', 'everywhere', 'example', 'except', 'exeunt', 'exit', 'fact', 'fair', 'far', 'farewell', 'few', 'fewer', 'fifteen', 'fifteenth', 'fifth', 'fifthly', 'fiftieth', 'fifty', 'finally', 'first', 'firstly', 'five', 'for', 'forever', 'forgo', 'forth', 'fortieth', 'forty', 'four', 'fourteen', 'fourteenth', 'fourth', 'fourthly', 'from', 'furthermore', 'generally', 'get', 'gets', 'getting', 'give', 'go', 'good', 'got', 'had', 'has', "hasn't", 'hast', 'hath', 'have', "haven't", 'having', 'he', "he'd", "he'll", "he's", 'hence', 'her', 'here', 'hers', 'herself', 'him', 'himself', 'his', 'hither', 'ho', 'how', "how's", 'however', 'hundred', 'hundredth', 'i', "i'd", "i'm", "i've", 'if', 'in', 'indeed', 'instance', 'instead', 'into', 'is', "isn't", 'it', "it'd", "it'll", "it's", 'its', 'itself', 'last', 'lastly', 'later', 'less', 'let', "let's", 'like', 'likely', 'many', 'matter', 'may', 'maybe', 'me', 'might', 'million', 'millionth', 'mine', 'more', 'moreover', 'most', 'much', 'must', "mustn't", 'my', 'myself', 'nay', 'near', 'nearby', 'nearly', 'neither', 'never', 'nevertheless', 'next', 'nine', 'nineteen', 'nineteenth', 'ninetieth', 'ninety', 'ninth', 'ninthly', 'no', 'nobody', 'none', 'noone', 'nor', 'not', 'nothing', 'now', 'nowhere', 'o', 'occasionally', 'of', 'off', 'oft', 'often', 'oh', 'on', 'once', 'one', 'only', 'or', 'order', 'other', 'others', 'ought', 'our', 'ours', 'ourselves', 'out', 'over', 'perhaps', 'possible', 'possibly', 'presumable', 'presumably', 'previous', 'previously', 'prior', 'probably', 'quite', 'rare', 'rarely', 'rather', 'result', 'resulting', 'round', 'said', 'same', 'say', 'second', 'secondly', 'seldom', 'seven', 'seventeen', 'seventeenth', 'seventh', 'seventhly', 'seventieth', 'seventy', 'shall', 'shalt', "shan't", 'she', "she'd", "she'll", "she's", 'should', "shouldn't", 'shouldst', 'similarly', 'since', 'six', 'sixteen', 'sixteenth', 'sixth', 'sixthly', 'sixtieth', 'sixty', 'so', 'soever', 'some', 'somebody', 'someone', 'something', 'sometimes', 'somewhere', 'soon', 'still', 'subsequently', 'such', 'sure', 'tell', 'ten', 'tenth', 'tenthly', 'than', 'that', "that's", 'the', 'thee', 'their', 'theirs', 'them', 'themselves', 'then', 'thence', 'there', "there's", 'therefore', 'these', 'they', "they'd", "they'll", "they're", "they've", 'thine', 'third', 'thirdly', 'thirteen', 'thirteenth', 'thirtieth', 'thirty', 'this', 'thither', 'those', 'thou', 'though', 'thousand', 'thousandth', 'three', 'thrice', 'through', 'thus', 'thy', 'till', 'tis', 'to', 'today', 'tomorrow', 'too', 'towards', 'twas', 'twelfth', 'twelve', 'twentieth', 'twenty', 'twice', 'twill', 'two', 'under', 'undergo', 'underneath', 'undoubtedly', 'unless', 'unlikely', 'until', 'unto', 'unusual', 'unusually', 'up', 'upon', 'us', 'very', 'was', "wasn't", 'wast', 'way', 'we', "we'd", "we'll", "we're", "we've", 'welcome', 'well', 'were', "weren't", 'what', "what's", 'whatever', 'when', 'whence', 'where', "where's", 'whereas', 'wherefore', 'whether', 'which', 'while', 'whiles', 'whither', 'who', "who's", 'whoever', 'whom', 'whose', 'why', 'wil', 'will', 'wilst', 'wilt', 'with', 'within', 'without', "won't", 'would', "wouldn't", 'wouldst', 'ye', 'yes', 'yesterday', 'yet', 'you', "you'd", "you'll", "you're", "you've", 'your', 'yours', 'yourself', 'yourselves'

Appendix C

List of POS tags

’NN’, ’IN’, ’JJ’, ’DT’, ’RB’, ’,’, ’PRP’, ’VBZ’, ’.’, ’PRP$’, ’VBP’, ’CC’, ’NNS’, ’VB’, ’TO’, ’VBN’, ’VBG’, ’VBD’, ’‘‘’, "’’", ’CD’, ’MD’, ’WRB’, ’WDT’, ’JJS’, ’RP’, ’:’, ’JJR’, ’FW’, ’WP’, ’UH’, ’(’, ’)’, ’RBS’, ’POS’, ’RBR’, ’EX’, ’$’, ’PDT’, ’NNP’, ’SYM’, ’NNPS’, ’WP$’, ’#’, ’LS’

Appendix D

Features description from Cheng et al. [2011]

Tab. D.1. Detailed features from Cheng et al. [2011]

# of features	Description
Character-based features	
1	Number of characters c
1	$\frac{\text{Number of letters [a-z]}}{c}$
1	$\frac{\text{Number of upper-cased characters}}{c}$
1	$\frac{\text{Number of digital characters [0-9]}}{c}$
1	$\frac{\text{Number of white spaces}}{c}$
1	$\frac{\text{Number of white spaces}}{c}$
1	$\frac{\text{Number of tabulation spaces}}{c}$
23	$\frac{\text{Number of special characters (\%,\&,etc)}}{c}$
Word-based features	
1	Number of words n
1	Average number of characters per word
1	Vocabulary richness $\frac{\text{Number of distinct words}}{n}$
1	$\frac{\text{Number of words length higher than 6 characters}}{n}$
1	$\frac{\text{Number of words length in [1-3] characters}}{n}$
1	$\frac{\text{Hapax legomen}}{n}$

Tab. D.1 – continued from previous page

# of features	Description
1	$\frac{\text{Hapax dislegomena}}{n}$
1	Yule's K measure (explored in subsection 3.9)
1	Simpson's D measure (explored in subsection 3.9)
1	Sichel's S measure (explored in subsection 3.9)
1	Honore's R measure (explored in subsection 3.9)
1	Entropy measure
1	$\frac{\text{Number of net abbreviation}}{n}$
20	$\frac{\text{Count of words of different lengths}}{n}$
68	LIWC features (explored in subsection 3.3)

Syntactic features	
1	$\frac{\text{count(')}}{c}$
1	$\frac{\text{count(,)}}{c}$
1	$\frac{\text{count(.)}}{c}$
1	$\frac{\text{count(:)}}{c}$
1	$\frac{\text{count(;)}}{c}$
1	$\frac{\text{count(?)}}{c}$
1	$\frac{\text{count(???)}}{c}$
1	$\frac{\text{count(,)}}{c}$
1	$\frac{\text{count(!)}}{c}$
1	$\frac{\text{count(!!!)}}{c}$
1	$\frac{\text{count(...)}}{c}$

Structural features	
1	Number of lines (l)
1	Number of sentences (s)
1	Number of paragraphs (p)
1	$\frac{s}{p}$

Tab. D.1 – continued from previous page

# of features	Description
1	$\frac{n}{p}$
1	$\frac{c}{p}$
1	$\frac{n}{s}$
1	$\frac{\#\text{ sentences starting with upper cased character}}{s}$
1	$\frac{\#\text{ sentences starting with lower cased character}}{s}$
1	$\frac{\#\text{ blank lines}}{l}$
1	Mean length of non blank lines
1	$\mathbb{1}_{\text{greeting words exists in the text}}$
1	$\mathbb{1}_{\text{farewell words exists in the text}}$
Function words	
3	$\frac{\#\text{ articles}}{n}$
4	$\frac{\#\text{ pro-sentence}}{n}$
74	$\frac{\#\text{ pronouns}}{n}$
47	$\frac{\#\text{ auxiliary verbs}}{n}$
22	$\frac{\#\text{ conjunctions}}{n}$
109	$\frac{\#\text{ interjections}}{n}$
124	$\frac{\#\text{ adpositions}}{n}$
9	$\frac{\#\text{ gender specific words}}{n}$

with count($char$) return the number of occurrences of the $char$ in the text.

Appendix E

Results for n-gram features for gender classification

Tab. E.1. Accuracy (%) for each features individually with each algorithm for gender classification with d being the dimension of the feature vector considered

Feature(s)	d	GNB	RF	Per	SVM	LR
LIWC	76	53.96	58.43	50.65	59.26	**59.59**
POS	45	52.95	58.21	50.49	59.15	**59.21**
SW	179	50.88	57.74	50.65	59.76	**59.86**
FW	467	50.91	57.74	50.65	60.19	**60.26**
EMB	100	58.19	60.06	50.84	62.95	**62.98**
VR	5	49.35	**52.69**	50.65	50.65	49.62
GI	184	54.23	58.73	55.27	61.23	**61.51**
SF	28	54.01	58.78	50.19	57.57	**57.78**
LP	26	53.37	56.43	50.65	57.58	**57.63**
DP	10	49.80	**51.73**	50.78	50.60	50.93

Tab. E.2. Accuracies (%) for different discriminative n-grams features with n being the length of the sequence considered and S the parameter of the discriminative indexing method

Features n-grams			S	MNB	BNB	GNB	RF	Perceptron	SVM	LR
Word n-grams	Character n-grams	POS n-grams								
1	-	-	1	60.01	50.86	61.87	60.34	63.63	-	62.86
1	-	-	5	66.72	66.87	67.29	65.95	67.39	-	67.95
1	-	-	25	67.2	68.25	65.15	64.64	62.26	-	67.78
1	-	-	50	67.17	67.72	64.92	63.81	58.53	-	67.71
1	-	-	200	67.07	66.26	63.64	62.98	59.24	-	67.55
2	-	-	1	65.51	51.33	64.58	63.23	64.09	-	65.3
2	-	-	5	68.69	63.27	67.11	65.32	63.43	-	67.36
2	-	-	25	68.3	66.76	60.61	58.56	64.91	-	67.04
2	-	-	50	68.28	66.62	65.7	56.36	65.29	-	67.03
2	-	-	200	68.25	65.35	64.81	58.0	65.29	-	66.93
-	3	-	5	52.65	60.29	53.76	61.37	57.86	-	62.67
-	3	-	200	53.5	63.1	53.7	60.77	52.95	-	62.51
-	4	-	5	59.25	63.1	60.97	63.93	55.38	-	66.25
-	4	-	200	60.78	66.14	58.94	61.76	55.52	-	65.62
-	5	-	5	65.73	63.21	66.68	65.27	59.41	-	67.88
-	5	-	200	66.11	67.84	63.07	61.9	64.29	-	67.0
-	6	-	5	68.9	61.66	68.07	65.56	61.95	-	68.5
-	6	-	200	68.78	67.97	67.5	62.4	59.34	-	67.65
-	7	-	1	64.64	50.72	64.11	62.45	62.08	-	64.5
-	7	-	5	70.04	59.91	68.65	66.18	62.03	-	68.94
-	7	-	25	69.9	68.16	66.48	64.0	65.96	-	68.43
-	7	-	50	69.86	68.71	66.6	57.0	66.59	-	68.32
-	7	-	200	69.82	67.92	66.72	61.08	66.55	-	68.31
-	8	-	1	65.37	50.73	64.86	63.28	58.77	-	65.26
-	8	-	5	70.35	58.61	69.14	66.13	68.03	-	69.21

Tab. E.2 – continued from previous page

Features n-grams			S	MNB	BNB	GNB	RF	Perceptron	SVM	LR
Word n-grams	Character n-grams	POS n-grams								
-	8	-	25	70.2	67.75	65.73	62.25	66.48	-	68.83
-	8	-	50	70.19	68.37	65.95	58.55	65.58	-	68.76
-	8	-	200	70.16	67.39	66.27	57.28	65.63	-	68.73
-	9	-	1	66.05	50.75	65.53	63.32	64.88	-	65.93
-	9	-	5	70.06	57.7	68.05	65.61	68.21	-	69.31
-	9	-	25	69.94	67.28	63.45	63.17	66.84	-	68.91
-	9	-	50	69.9	67.94	63.52	56.61	65.34	-	68.85
-	9	-	200	69.88	66.67	63.93	62.50	65.47	-	68.77
-	10	-	1	66.41	50.82	65.83	63.02	65.43	-	66.28
-	10	-	5	69.82	57.27	67.29	65.14	68.2	-	69.14
-	10	-	25	69.68	66.73	65.14	61.93	65.59	-	68.78
-	10	-	50	69.67	67.37	65.29	62.99	65.09	-	68.73
-	-	1	1	50.65	50.65	52.01	52.57	51.24	-	52.82
-	-	1	5	50.65	50.65	51.65	52.45	51.33	-	52.83
-	-	1	25	50.66	52.44	51.11	53.15	50.66	-	53.26
-	-	1	50	50.65	52.07	51.05	53.59	50.65	-	53.33
-	-	1	200	50.66	52.33	51.06	53.81	50.66	-	53.81
-	-	2	1	50.76	50.65	54.79	54.84	50.65	-	55.98
-	-	2	5	50.89	55.2	51.3	55.4	50.65	-	56.41
-	-	2	25	51.28	56.31	50.01	57.1	50.99	-	57.71
-	-	2	50	51.34	56.27	49.93	57.13	50.91	-	57.69
-	-	2	200	51.33	54.65	50.1	56.92	50.77	-	57.72
-	-	3	1	53.03	50.65	58.37	56.08	50.65	-	58.93
-	-	3	5	53.87	56.04	52.95	56.8	51.61	-	59.3
-	-	3	25	56.2	57.72	51.12	58.26	50.81	-	60.04
-	-	3	50	56.24	58.02	51.12	58.38	53.13	-	60.02

Tab. E.2 – continued from previous page

Features n-grams			S	MNB	BNB	GNB	RF	Perceptron	SVM	LR
Word n-grams	Character n-grams	POS n-grams								
-	-	3	200	56.34	56.9	51.71	57.91	53.59	-	59.94
-	-	4	1	57.58	50.82	59.52	57.27	52.69	-	59.94
-	-	4	5	58.62	55.37	57.3	56.73	53.11	-	58.09
-	-	4	25	59.2	57.11	53.12	56.52	53.04	-	58.27
-	-	4	200	59.22	58.16	52.61	56.1	55.1	-	58.17

Tab. E.3. Greedy search to find the best combination using majority vote system over outputs of classifiers trained on Discriminative n-gram Indexing with $S = 5$

Number of classifiers combined	Feature	Classifier	Accuracy (%)
1	Character 8-grams	MNB	70.35
3	+ Character 9-grams + Character 7-grams	MNB MNB	70.45
5	+ Character 10-grams + Character 9-grams	MNB LR	70.47
7	+ Character 8-grams + Character 8-grams	LR GNB	70.51
9	+ Character 10-grams + Character 7-grams	LR LR	70.60
11	+ Character 6-grams + Word 2-grams	MNB MNB	70.83
13	+ Character 7-grams + Character 6-grams	GNB LR	**71.00**
15	+ Character 9-grams + Character 10-grams	Perceptron Perceptron	70.91

Each row corresponds to the two classifiers added for each experiment and the total number of classifiers combined

Tab. E.4. TF-IDF results for gender classification with n being the length of the sequence considered while considering all n-grams for each feature

n-gram level	n	LR accuracy (%)
Word	1	71.17
	2	67.52
	3	61.82
	{1,2}	71.71
	{1...3}	71.43
Character	3	69.48
	4	71.60
	5	71.86
	6	71.99
	7	71.92
	8	71.62
	9	71.13
	{3,..., 5}	**72.41**
	{6,..., 8}	72.32

Tab. E.5. Greedy search for the best combination using majority vote system over outputs of LR classifiers trained on TF-IDF

Number of classifiers combined	*n*-gram level	Accuracy (%)
1	Character {3...5}-grams	72.41
3	+ Character {6...8}-grams + Character 6-grams	72.64
5	+ Character 7-grams + Character 5-grams	72.67
7	+ Word {1...2}-grams + Character 8-grams	72.95
9	+ Character 4-grams + Word {1...3}-grams	73.34
11	+ Word 1-gram + Character 9-grams	73.47
13	+ Character 3-grams + Word 2-grams	**73.61**

Each row corresponds to the two classifiers added for each experiment and the total number of classifiers combined and the accuracy obtained combining the classifiers

Tab. E.6. Greedy search for the best combination using majority vote system over outputs of LR classifiers trained on TF-IDF and others trained on Discriminative n-gram Indexing (DnI)

Number of classifiers combined	Feature	n-gram level	Accuracy (%)
1	TF-IDF	Character {3...5}-grams	72.41
3	TF-IDF TF-IDF	+ Character {6...8}-grams + Character 6-grams	72.64
5	TF-IDF TF-IDF	+ Character 7-grams + Character 5-grams	72.67
7	TF-IDF TF-IDF	+ Word {1...2}-grams + Character 8-grams	72.95
9	TF-IDF TF-IDF	+ Character 4-grams + Word {1...3}-grams	73.34
11	TF-IDF TF-IDF	+ Word 1-gram + Character 9-grams	73.47
13	DnI DnI	+ Character 8-grams (MNB) + Character 9-grams (MNB)	73.78
15	DnI DnI	+ Character 7-grams (MNB) + Character 10-grams (MNB)	73.89
17	TF-IDF DnI	+ Character 3-grams + Character 9-grams (LR)	73.97
19	DnI DnI	+ Character 8-grams (LR) + Character 8-grams (LR)	73.99
21	DnI DnI	+ Character 10-grams (LR) + Character 7-grams (LR)	**74.02**
23	DnI DnI	+ Character 6-grams (MNB) + Word 2-grams (MNB)	73.90

Each row corresponds to the two classifiers added for each experiment and the total number of classifiers combined and the accuracy obtained combining the classifiers

Tab. E.7. Accuracies (%) using vector of probability for each topic topic

# topics	MNB	BNB	GNB	RF	Perceptron	LR
50	59.51	58.56	57.57	57.72	51.88	60.00
100	58.88	58.82	56.70	57.76	56.56	59.37
150	59.06	58.84	57.13	57.47	55.19	59.57
200	58.53	58.75	56.42	57.24	50.66	59.49

Tab. E.8. Accuracies (%) using vector of binary representation for each topic

# topics	MNB	BNB	GNB	RF	Perceptron	LR
50	58.68	58.56	57.93	56.23	54.87	59.06
100	58.65	58.82	58.24	56.47	55.07	59.17
150	59.01	58.84	58.68	56.37	53.45	59.47
200	59.05	58.75	58.91	56.16	53.50	59.64

Appendix F

Deep learning results for gender classification

Tab. F.1. RNN unidirectional performances with LR=0.01

BS	#HS	#HL	Train (%)	Valid (%)	Test (%)
32	10	1	60.15	59.91	59.80
		2	59.91	60.57	60.60
		3	52.62	54.21	54.18
	50	1	59.96	59.93	60.30
		2	58.05	60.02	59.97
		3	50.57	52.05	52.11
	100	1	57.91	59.79	59.69
		2	57.33	59.40	59.13
		3	49.97	50.32	50.27
64	10	1	55.17	56.03	55.64
		2	55.09	55.73	55.79
		3	54.24	56.04	55.88
	50	1	50.76	52.15	52.13
		2	51.71	52.81	52.66
		3	50.73	52.82	52.71
	100	1	50.69	52.15	52.14
		2	50.16	51.05	50.71
		3	50.34	51.96	51.93

Tab. F.2. RNN bidirectional performances with LR=0.01

BS	#HS	#HL	Train (%)	Valid (%)	Test (%)
32	10	1	61.71	61.49	61.62
		2	62.64	62.29	62.50
		3	62.66	62.49	62.33
	50	1	61.20	61.30	61.30
		2	60.09	61.27	61.42
		3	59.63	60.93	60.97
	100	1	59.77	60.80	61.15
		2	57.07	59.67	59.84
		3	55.94	58.89	58.97
64	10	1	56.34	56.75	56.82
		2	55.13	56.72	56.92
		3	54.24	55.85	55.77
	50	1	52.91	53.56	53.54
		2	52.43	57.56	57.43
		3	51.98	55.17	55.31
	100	1	50.39	51.19	50.91
		2	51.19	53.41	53.01
		3	50.50	52.02	51.83

Tab. F.3. RNN unidirectional performances with LR=0.001

BS	#HS	#HL	Train (%)	Valid (%)	Test (%)
64	10	1	60.03	60.38	60.27
		2	60.78	60.86	60.74
		3	60.36	60.64	60.72
	50	1	59.78	60.36	60.48
		2	58.91	60.31	60.14
		3	59.01	60.15	59.98
	100	1	56.27	59.14	59.02
		2	55.44	59.17	59.25
		3	56.28	58.64	58.54

Tab. F.4. RNN bidirectional performances with LR=0.001

BS	#HL	#HS	Train (%)	Valid (%)	Test (%)
64	10	1	62.59	62.12	62.32
		2	63.57	63.29	63.46
		3	63.48	62.95	63.17
	50	1	61.56	62.08	62.02
		2	61.55	61.92	61.88
		3	61.19	61.42	61.50
	100	1	60.24	61.69	62.01
		2	59.50	61.32	61.40
		3	59.05	60.27	60.40

Tab. F.5. Performances of RCNN with LR=0.01

BS	#HS	Train (%)	Valid (%)	Test (%)
32	5	63.70	63.58	63.72
	10	63.95	63.83	63.96
	50	63.69	63.71	63.75
	100	63.50	63.36	63.63
64	5	65.90	64.79	64.99
	10	66.75	65.20	65.24
	50	65.87	65.00	65.34
	100	66.21	64.84	65.28

Tab. F.6. Performances of RCNN with LR=0.001

BS	#HS	Train (%)	Valid (%)	Test (%)
32	5	69.31	66.55	66.78
	10	70.89	68.05	68.32
	50	70.68	67.16	67.37
	100	70.15	67.14	67.91
64	5	68.74	66.44	66.33
	10	69.86	67.20	67.27
	50	73.65	68.13	68.28
	100	74.59	68.28	68.32

Appendix G

Results for n-gram features for age classification

Tab. G.1. Accuracy (%) for each features individually with each algorithm for age classification with d being the dimension of the feature vector considered

Type	Feature	d	GNB	RF	Perceptron	SVM	LR
Content	SW	179	22.39	**53.46**	48.19	52.85	43.97
	LP	26	46.46	**52.97**	46.95	51.47	42.59
	DP	10	46.39	46.25	**46.96**	46.98	42.37
	FW	467	35.96	**53.93**	50.97	54.2	45.89
	EMB	100	49.31	**57.22**	50.22	56.16	50.06
	TF-IDF (unigrams of words)	477,837	-	60.82	-	-	**67.95**
Style	POS	45	48.30	**54.95**	48.56	52.68	44.28
	VR	5	17.90	**46.93**	35.14	36.69	35.14
	SF	28	47.01	**56.06**	46.96	24.68	43.31
Both	LIWC	76	33.99	**55.93**	50.31	54.23	44.55
	GI	184	35.51	41.82	38.22	**54.85**	33.25

Tab. G.2. Greedy search for the best combination using LR with d being the dimension of the features vector considered

# families of features	Feature(s)	d	Accuracy (%)
1	TF-IDF (unigrams of words)	477,837	67.95
2	+ EMB	477,937	68.18
3	+ FW	478,404	68.46
4	+ LIWC	478,480	68.57
5	+ POS	478,525	**68.64**
6	+ SW	478,704	68.62

Tab. G.3. Accuracies (%) for different Discriminative n-grams Indexing (DnI) with n being the length of the sequence considered and S the parameter of DnI

Features n-grams			S	MNB	BNB	GNB	RF	Perceptron	LR
Word n-grams	Character n-grams	POS n-grams							
1	-	-	1	46.97	48.44	60.55	60.81	48.66	57.69
1	-	-	5	49.67	63.27	55.05	64.21	60.04	62.39
1	-	-	25	50.68	64.74	53.68	63.26	51.94	62.08
1	-	-	50	50.91	62.98	53.03	61.77	60.62	61.84
1	-	-	200	50.87	59.74	52.54	60.8	60.89	61.39
2	-	-	1	48.37	49.66	64.03	61.54	64.83	63.5
2	-	-	5	59.15	53.81	62.09	63.98	58.66	64.93
2	-	-	25	60.63	65.44	48.69	58.82	59.44	64.26
2	-	-	50	60.82	64.81	49.19	55.59	58.37	64.15
2	-	-	200	60.97	62.6	57.22	55.94	57.19	63.93
3	-	-	1	54.01	49.47	55.93	60.23	47.31	64.43
3	-	-	5	59.87	43.96	45.97	61.31	62.05	62.39
3	-	-	25	60.92	61.42	55.3	56.52	61.2	62.5
3	-	-	50	61.04	61.44	56.38	56.21	55.53	62.37

Tab. G.3 – continued from previous page

Features n-grams			S	MNB	BNB	GNB	RF	Perceptron	LR
Word n-grams	Character n-grams	POS n-grams							
3	-	-	200	61.13	59.56	49.24	57.6	61.33	62.16
-	3	-	1	46.96	46.96	50.91	55.41	47.67	48.8
-	3	-	5	47.0	55.19	55.69	59.41	53.96	51.24
-	3	-	25	47.04	53.11	54.21	60.62	59.92	51.68
-	3	-	50	47.04	49.86	33.73	60.39	53.41	51.82
-	3	-	200	47.03	45.21	31.83	59.19	53.06	51.2
-	4	-	1	46.96	46.98	58.57	58.52	50.13	54.55
-	4	-	5	47.66	57.67	58.75	62.19	46.9	59.43
-	4	-	25	48.09	59.8	54.52	61.09	45.89	59.12
-	4	-	50	48.11	57.11	51.15	60.49	40.17	58.95
-	4	-	200	48.05	51.54	45.41	59.38	56.49	58.3
-	5	-	1	46.96	47.5	60.08	60.64	56.84	58.2
-	5	-	5	49.64	55.8	58.36	63.23	52.57	63.67
-	5	-	25	50.61	64.08	52.66	60.54	52.76	63.35
-	5	-	50	50.68	62.61	50.72	58.41	59.95	63.12
-	5	-	200	50.71	57.66	51.15	57.84	57.56	62.7
-	6	-	1	46.97	47.73	62.03	62.04	62.79	60.87
-	6	-	5	52.06	51.05	63.06	63.9	60.82	66.14
-	6	-	25	53.73	65.07	53.33	59.7	61.93	65.56
-	6	-	50	54.04	65.56	49.37	56.83	61.38	65.44
-	6	-	200	54.14	61.02	52.72	58.78	61.9	64.99
-	7	-	1	47.1	47.73	63.55	62.38	63.27	62.63
-	7	-	5	55.07	46.3	62.88	64.62	63.09	67.02
-	7	-	25	57.27	65.46	57.25	60.98	65.52	66.54
-	7	-	50	57.64	66.77	55.77	56.84	65.77	66.39
-	7	-	200	57.78	63.15	54.54	52.72	65.78	66.13
-	8	-	1	47.65	47.74	64.28	60.87	57.37	63.04

Tab. G.3 – continued from previous page

Features *n*-grams			S	MNB	BNB	GNB	RF	Perceptron	LR
Word *n*-grams	Character *n*-grams	POS *n*-grams							
-	8	-	5	57.89	42.21	62.84	65.02	65.59	67.28
-	8	-	25	59.9	65.29	58.23	61.29	65.17	66.72
-	8	-	50	60.22	66.98	57.26	57.6	65.71	66.7
-	8	-	200	60.37	63.92	57.67	53.77	65.7	66.53
-	9	-	1	48.9	47.7	64.65	59.92	54.87	64.72
-	9	-	5	59.55	39.43	61.48	64.85	64.25	67.22
-	9	-	25	61.21	64.78	53.08	62.98	64.34	66.52
-	9	-	50	61.53	66.92	58.37	55.12	65.6	66.42
-	9	-	200	61.58	64.16	56.18	52.27	65.07	66.31
-	10	-	1	51.07	47.72	63.24	61.83	50.47	65.79
-	10	-	5	60.23	38.24	60.51	63.48	62.53	67.03
-	10	-	25	61.7	64.11	53.25	54.81	58.35	66.49
-	10	-	50	61.94	66.63	54.33	56.85	65.07	66.25
-	10	-	200	62.03	64.83	53.67	59.11	59.13	66.11
-	-	1	1	46.96	46.96	46.96	45.84	46.96	31.78
-	-	1	5	46.96	46.96	46.86	45.28	46.96	31.7
-	-	1	25	46.96	46.69	46.88	45.46	46.97	33.17
-	-	1	50	46.96	46.6	20.43	46.66	46.93	35.04
-	-	1	200	46.96	47.16	24.25	47.85	46.87	36.27
-	-	2	1	46.96	46.96	47.2	49.39	47.34	38.93
-	-	2	5	46.96	47.2	35.68	51.74	49.3	40.51
-	-	2	25	46.97	47.53	19.71	53.25	46.96	42.13
-	-	2	50	46.97	45.23	19.38	53.55	46.95	42.63
-	-	2	200	46.98	39.77	20.09	53.09	47.46	42.68
-	-	3	1	46.96	46.96	52.04	51.83	48.1	44.92
-	-	3	5	47.01	49.94	51.91	54.08	48.64	45.9
-	-	3	25	47.11	47.91	38.58	55.18	51.13	46.19

Tab. G.3 – continued from previous page

Features n-grams			S	MNB	BNB	GNB	RF	Perceptron	LR
Word n-grams	Character n-grams	POS n-grams							
-	-	3	50	47.13	43.88	37.24	55.24	52.91	46.14
-	-	3	200	47.18	38.93	24.13	54.7	47.19	46.36
-	-	4	1	46.96	46.96	53.99	53.64	47.83	50.8
-	-	4	5	47.73	49.67	49.19	52.13	49.69	47.88
-	-	4	25	48.31	48.47	41.77	52.88	49.22	47.92
-	-	4	50	48.34	45.88	41.29	52.79	49.81	47.97
-	-	4	200	48.34	42.0	41.15	52.87	49.07	48.03
-	-	5	1	46.99	46.96	55.9	54.0	51.43	54.14
-	-	5	5	50.42	44.96	49.07	51.53	52.39	49.58
-	-	5	25	51.66	50.87	44.65	48.46	52.4	49.58
-	-	5	50	51.73	50.18	46.02	49.98	52.13	49.6
-	-	5	200	51.74	47.03	47.48	48.87	52.0	49.56

Tab. G.4. Greedy search to find the best combination using majority vote system over outputs of classifiers trained on Discriminative n-gram Indexing with $S = 5$

Number of classifiers combined	Feature	Classifier	Accuracy (%)
1	Character 8-grams	LR	67.28
3	+ Character 9-grams	LR	67.49
	+ Character 10-grams	LR	
5	+ Character 7-grams	LR	**68.14**
	+ Character 6-grams	LR	
7	+ Character 8-grams	Perceptron	67.93
	+ Character 8-grams	RF	

Each row corresponds to the two classifiers added for each experiment and the total number of classifiers combined

Tab. G.5. TF-IDF results for gender classification with n being the length of the sequence considered while considering all n-grams for each feature

n-gram level	n	LR accuracy (%)
Word	1	67.95
	2	67.36
	3	60.81
	{1,2}	71.21
	{1...3}	70.78
Character	3	64.3
	4	68.83
	5	70.71
	6	71.60
	7	71.66
	8	71.46
	9	71.24
	{1,..., 4}	67.97
	{5,..., 7}	**72.20**

Tab. G.6. Greedy search for the best combination using majority vote system over outputs of LR classifiers trained on TF-IDF

Number of classifiers combined	n-gram level	Accuracy (%)
1	Character {5...7}-grams	**72.20**
3	+ Character 7-grams + Character 6-grams	72.11

Each row corresponds to the two classifiers added for each experiment and the total number of classifiers combined and the accuracy obtained combining the classifiers

Tab. G.7. Accuracies (%) using vector of probability for each topic topic

# topics	MNB	BNB	GNB	RF	Perceptron	LR
50	48.08	51.2	47.75	54.37	47.0	44.16
100	48.35	49.75	47.26	54.92	47.73	46.0
150	47.43	49.62	44.9	54.45	47.97	46.0
200	47.43	49.33	45.19	54.27	47.19	46.19

Tab. G.8. Accuracies (%) using vector of binary representation for each topic

# topics	MNB	BNB	GNB	RF	Perceptron	LR
50	52.78	51.2	48.94	51.91	44.36	43.35
100	52.93	49.75	44.25	52.55	41.5	43.56
150	53.32	49.62	43.92	52.57	47.55	44.57
200	53.08	49.33	44.11	52.39	44.02	45.37