ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

On Authorship Attribution

Relatore:
Chiar.mo Prof.
DANILO MONTESI

Correlatore:
Dott.
FLAVIO BERTINI

Presentata da:
Gabriele Calarota

Sessione III
Anno Accademico 2019-2020

“When I was in college,
I wanted to be involved in things that would change the world”
Elon Musk

Sommario

Abstract

Contents

CHAPTER 2

Authorship attribution's tasks

"Science is the systematic classification of experience."
George Henry Lewes

The task of determining or verifying the authorship of an anonymous text based solely on internal evidence is a very old one, dating back at least to the medieval scholastics, for whom the reliable attribution of a given text to a known ancient authority was essential to determining the text's veracity. More recently, this problem of authorship attribution has gained greater prominence due to new applications in forensic analysis, humanities scholarship, and electronic commerce, and the development of computational methods for addressing the problem. Statistical authorship attribution has a long history, culminating in the use of modern machine learning classification methods. In the simplest form of the problem, we are given examples of the writing of a number of candidate authors and are asked to determine which of them authored a given anonymous text. In this straightforward form, the authorship attribution problem fits the standard modern paradigm of a text categorization problem [24] [39]. The components of text categorization systems are by now fairly well understood: Documents are represented as numerical vectors that capture statistics of potentially relevant features of the text, and machine learning methods are used to find classifiers that separate documents that belong to different classes. In the next section we will briefly present the approach to this task in the era before machine learning. Later, we will discuss the different approaches of this particular task, whether to approach it as a profile-based or an instance-based problem, depending on which domain are we tackling: single-domain or cross-domain. In the last section, we will introduce to the latest state of the art approaches for this particular classification task.

2.1 History of methodologies

The first attempts to quantify the writing style go back to the 19th century, with the pioneering study of Mendenhall (1887) on the plays of Shakespeare, followed by statistical studies in the first half of the 20th century by Yule (1938, 1944) and Zipf (1932) [45] [47] [49]. Later, the detailed study by Mosteller and Wallace (1964) on the authorship of "The Federalist Papers" (a series of 146 political essays written by John Jay, Alexander Hamilton, and James Madison, 12 of which claimed by both Hamilton and Madison) was undoubtedly the most influential work in authorship attribution [41]. Their method was based on Bayesian statistical analysis of the frequencies of a small set of common words (e.g., "and," "to," etc.) and produced significant discrimination results between the candidate authors. Essentially, the work of Mosteller and Wallace (1964) initiated nontraditional authorship attribution studies, as opposed to traditional human-expert-based methods. Since then and until the late 1990s, research in authorship attribution was dominated by attempts to define features for quantifying writing style, a line of research known as *"stylometry"* [14]. Hence, a great variety of measures, including sentence length, word length, word frequencies, character frequencies, and vocabulary richness functions, had been proposed. Rudman (1998) estimated that nearly 1,000 different measures had been proposed by the late 1990s [35]. The authorship attribution methodologies proposed during that period were computer-assisted rather than computer-based, meaning that the aim was rarely at developing a fully automated system. In certain cases, there were methods that achieved impressive preliminary results and made many people think that the solution of this problem was too close. The most characteristic example is the CUSUM (or QSUM) technique [27] that gained publicity and was accepted in courts as expert evidence; however, the research community heavily criticized it and considered it generally unreliable [15]. Actually, the main problem of that early period was the lack of objective evaluation of the proposed methods. In most of the cases, the testing ground was literary works of unknown or disputed authorship (e.g., the Federalist Papers case), so the estimation of attribution accuracy was not even possible. The main methodological limitations of that period concerning the evaluation procedure were the following:

- The textual data were too long (usually including entire books) and probably not stylistically homogeneous.
- The number of candidate authors was too small (usually two or three).
- The evaluation corpora were not controlled for topic.
- The evaluation of the proposed methods was mainly intuitive (usually based on subjective visual inspection of scatterplots).

- The comparison of different methods was difficult due to lack of suitable benchmark data.

Since the late 1990s, things have changed in authorship attribution studies. The vast amount of electronic texts available through Internet media (e-mail messages, blogs, online forums, etc.) have increased the need for efficiently handling this information. This fact had a significant impact in scientific areas such as information retrieval, machine learning, and natural language processing (NLP). The development of these areas influenced authorship attribution technology as described:

- Information retrieval research developed efficient techniques for representing and classifying large volumes of text.

- Powerful machine learning algorithms became available to handle multidimensional and sparse data, allowing more expressive representations. Moreover, standard evaluation methodologies have been established to compare different approaches on the same benchmark data.

- NLP research developed tools able to analyze text efficiently and provided new forms of measures for representing the style (e.g., syntax-based features).

More importantly, the plethora of available electronic texts revealed the potential of authorship analysis in various applications [25] in diverse areas including intelligence (e.g., attribution of messages or proclamations to known terrorists, linking different messages by authorship) [1], criminal law (e.g., identifying writers of harassing messages, verifying the authenticity of suicide notes) and civil law (e.g., copyright disputes) [7], and computer forensics (e.g., identifying the authors of source code of malicious software) [10] in addition to the traditional application to literary research (e.g., attributing anonymous or disputed literary works to known authors) [5]. Hence, (roughly) the last decade can be viewed as a new era of authorship analysis technology, this time dominated by efforts to develop practical applications dealing with real-world texts (e.g., e-mail messages, blogs, online forum messages, source code, etc.) rather than solving disputed literary questions. Emphasis has now been given to the objective evaluation of the proposed methods as well as the comparison of different methods based on common benchmark corpora. In addition, factors playing a crucial role in the accuracy of the produced models are examined, such as the training text size [26], the number of candidate authors [19], and the distribution of training texts over the candidate authors [40].

2.2 Training's approach

In every authorship-identification problem, there is a set of candidate authors, a set of text samples of known authorship covering all the candidate authors (*training corpus*), and a set of text samples of unknown authorship (*test corpus*); each one of them should be attributed to a candidate author. In this survey, we distinguish the authorship attribution approaches according to whether they treat each training text individually or cumulatively (per author). In more detail, some approaches concatenate all the available training texts per author in one big file and extract a cumulative representation of that author's style (usually called the author's profile) from this concatenated text. That is, the differences between texts written by the same author are disregarded. Such approach just described is called *"profile-based approach"* and early work in authorship attribution has followed this practice [33]. On the other hand, another family of approaches requires multiple training text samples per author to develop an accurate attribution model. That is, each training text is individually represented as a separate instance of authorial style. Such *"instance-based approaches"*[1] are described in the next section, followed by hybrid approaches attempting to combine characteristics of profile-based and instance-based methods. We then compare these two basic approaches and discuss their strengths and weaknesses across several factors.

2.2.1 Profile-based approach

One way to handle the available training texts per author is to concatenate them in one single text file. This large file is used to extract the properties of the author's style. An unseen text is, then, compared with each author file, and the most likely author is estimated based on a distance measure. It should be stressed that there is no separate representation of each text sample but only one representation of a large file per author. As a result, the differences between the training texts by the same author are disregarded. Moreover, the stylometric measures extracted from the concatenated file may be quite different in comparison to each of the original training texts.

A typical architecture of a profile-based approach is depicted in figure 2.1. Note that x denotes a vector of text representation features. Hence, x_A is the profile of Author A, and x_u is the profile of the unseen text. The profile-based approaches have a very simple training process. Actually, the training phase just comprises the extraction of profiles for the candidate authors. Then, the attribution model is usually based on a distance function that computes the differences of the profile of an unseen text and the profile of each author. Let $PR(x)$ be the profile of text x and $d[PR(x), PR(y)]$ the distance between the profile of text x and the profile of text y. Then, the most likely author of an unseen text x is given in 2.1, where A is the set of candidate authors and x_a is the concatenation of all training texts for author a.

$$author(x) = \arg_{a \in A} min\ d(PR(x), PR(x_a) \tag{2.1}$$

[1]Note that this term should not be confused with instance-based learning methods[32]

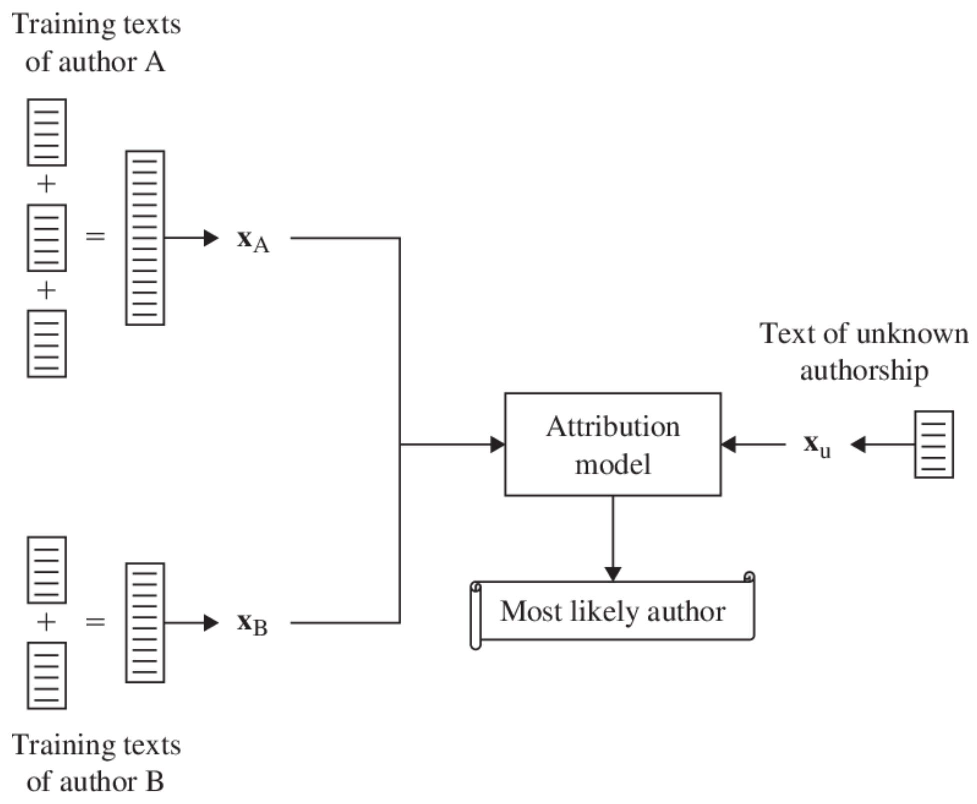
Training texts of author A
$\mathbf{x}_A$
Text of unknown authorship
Attribution model
$\mathbf{x}_u$
Most likely author
$\mathbf{x}_B$
Training texts of author B

Figure 2.1: Typical architecture of profile-based in authorship attribution approaches [41]

2.2.2 Instance-based approach

The majority of the modern authorship-identification approaches considers each training text sample as a unit that contributes separately to the attribution model. In other words, each text sample of known authorship is an instance of the problem in question. A typical architecture of such an instance-based approach is shown in figure 2.2. In detail, each text sample of the training corpus is represented by a vector of attributes (x) following methods described earlier, and a classification algorithm is trained using the set of instances of known authorship (*training set*) to develop an attribution model. Then, this model will be able to estimate the true author of an unseen text.

Note that such classification algorithms require multiple training instances per class for extracting a reliable model. Therefore, according to instance-based approaches, in case we have only one, but a quite long, training text for a particular candidate author (e.g., an entire book), this should be segmented into multiple parts, probably of equal length. From another point of view, when there are multiple training text samples of variable length per author, the training text instance length should be normalized. To that end, the training texts per author are segmented to equal-sized samples [36]. In all these cases, the text samples should be long enough so that the text representation features can adequately represent their style. Various lengths of text samples have been reported in the literature. Sanderson and Guenter (2006) produced chunks of 500 characters [36].

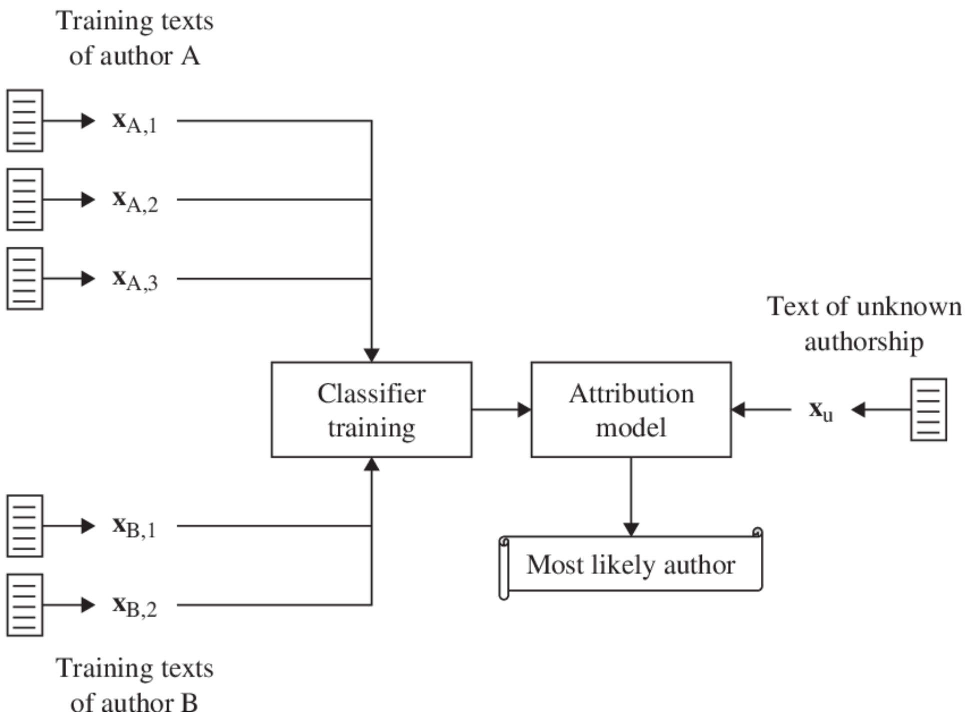

Figure 2.2: Typical architecture of instance-based in authorship attribution approaches [41]

Koppel et al. (2006) segmented the training texts into chunks of about 500 words [19]. Hirst and Feiguina (2007) conducted experiments with text blocks of varying length (i.e.: 200, 500, and 1000 words) and reported significantly reduced accuracy as the text-block length decreases [13]. Therefore, the choice of the training instance text sample is not a trivial process and directly affects the performance of the attribution model.

2.3 The real problem

Statistical authorship attribution has a long history, culminating in the use of modern machine learning classification methods. Nevertheless, most of this work suffers from the limitation of assuming a small closed set of candidate authors and essentially unlimited training text for each. Real-life authorship attribution problems, however, typically fall short of this ideal. Thus in [20] three scenarios are presented for which solutions to the basic attribution problem are inadequate. For example, we may encounter scenarios such as:

1. There is no candidate set at all. In this case, the challenge is to provide as much demographic or psychological information as possible about the author. This is the *profiling problem.*

2. There are many thousands of candidates for each of whom we might have a very limited writing sample. This is the *needle-in-a-haystack* problem.

3. There is no closed candidate set but there is one suspect. In this case, the challenge is to determine if the suspect is or is not the author. This is the *verification problem.*

In the following section we will show how machine learning methods can be adapted to handle the special challenges of each variant.

2.3.1 Profiling problem

Even in cases where we have an anonymous text and no candidate authors, we would like to say something about the anonymous author. That is, we wish to exploit the sociolinguistic observation that different groups of people speaking or writing in a particular genre and in a particular language use that language differently [6]. This authorship profiling problem is of growing importance in the current global information environment applications abound in forensics, security, and commercial settings. For example, authorship profiling can help police identify characteristics of the perpetrator of a crime when there are too few (or too many) specific suspects to consider. Similarly, large corporations may be interested in knowing what types of people like or dislike their products, based on analysis of blogs and online product reviews. The question we therefore ask is: How much can we discern about the author of a text simply by analyzing the text itself? It turns out that, with varying degrees of accuracy, we can say a great deal indeed. One of the approach to authorship profiling is to apply machine learning to text categorization. The process is as follows: First, we take a given corpus of training documents, each labeled according to its category for a particular profiling dimension. For example, when addressing classification by author gender, training documents are labeled as either *male* or *female.* Each document is then processed to produce a numerical vector, each of whose elements represents some feature of the text that might help discriminate the relevant categories. A machine learning method then computes a classifier that, to the extent possible, classifies the training examples correctly. Finally, the predictive power of the classifier is tested on out-of-training data. Essentially the same paradigm can be used for authorship attribution, where the training texts are known writings of given candidate authors.[16]

2.3.2 Needle-in-hay-stack problem

Consider now the scenario where we seek to determine the specific identity of a document's author, but there are many thousands of potential candidates. We call this the *needle-in-a-haystack* attribution problem. In this case, standard text-classification techniques are unlikely to give reasonable accuracy and may require excessive computation time to learn classification models; however, in [20] and [21] state that if we are willing to tolerate our system telling us it does not know the answer, we can achieve high accuracy for the cases where the system does give us an attribution it consider reliable. Recent works, [38] and [43] have addressed the problem with tackling a large set of candidate authors while keeping the documents very short, also known as "micro-messages" or "tweets"[2].

2.3.3 Verification problem

Consider the case in which we are given examples of the writing of a single author and are asked to verify that a given target text was or was not written by this author. As a categorization problem, verification is significantly more difficult than basic attribution, and virtually no work has been done on it (but see Halteren 2004), outside the framework of plagiarism detection Zu Eissen et al. 2007. If, for example, all we wished to do is to determine if a text had been written by Shakespeare or by Marlowe, it would be sufficient to use their respective known writings, to construct a model distinguishing them, and to test the unknown text against the model. If, on the other hand, we need to determine if a text was written by Shakespeare, it is difficult to assemble a representative sample of non-Shakespeare texts. The situation in which we suspect that a given author may have written some text, but do not have an exhaustive list of alternative candidates, is a common one [20]. The problem is complicated by the fact that a single author may vary his or her style from text to text or may unconsciously drift stylistically over time, not to mention the possibility of conscious deception. Thus, we must learn to somehow distinguish between relatively shallow differences that reflect conscious or unconscious changes in an author's style and deeper differences that reflect styles of different authors.

2.4 Identify the problem

Since authorship attribution has been studied for many decades now, we have witnessed the rise of many different subtasks. In the next part of this section we will illustrate one of the main difference between approaching a single domain or a cross domain dataset in order to get the stylometric markers of each authors. We will also discuss the difference between closed set and open set authorship attribution problem.

2.4.1 Single domain vs Cross domain

When dealing with a dataset with documents written by different authors, the first analysis to carry is to identify the different topics of each document written by the same author. This analysis is very important for the rest of the analysis, because previous

[2]As the name recall, the name of the posts on the popular social media platform Twitter. Its main characteristics is that originally only 140 characters per post were allowed (at the moment of writing, it has been upgraded up to 280 characters per tweet).

works have shown a degradation in terms of performance when approaching cross domain dataset, being suited for the single domain approach. The reason behind this is trivial: content words used by the same author will vary a lot when writing about different topics (such as: motors, science, literature,or politics). What should remain steady in the style of writing of an author across different topics should be function words, the use of punctuation ..etc Although a lot of work showed good results in both context ([37], [17], [34]) for the rest of this work, we will focus on single domain authorship attribution, but we will also show results of our model trained on a cross domain dataset[3].

2.4.2 Closed set vs Open set

The simplest kind of authorship attribution problem—and the one that has received the most attention—is the one in which we are given a small closed set of candidate authors and are asked to attribute an anonymous text to one of them. Usually, it is assumed that we have copious quantities of text by each candidate author and that the anonymous text is reasonably long. A number of recent survey papers amply cover the variety of methods used for solving this problem, known as *closed set authorship attribution.*
A significant fact that examination of the literature reveals is that nearly all research in the field only considers the simplest version of the problem.

Unfortunately, this "vanilla" version of the authorship attribution problem does not often arise in the real world. We often encounter situations in which our list of candidates might be very large and in which there is no guarantee that the true author of an anonymous text is even among the candidates. [22] Similarly, almost all work in authorship attribution has focused on the case in which the candidate set is a closed set—the anonymous text is assumed to have been written by one of the known candidates. The more general case, in which the true author of an anonymous text might not be one of the known candidates, reduces to the binary authorship verification problem: determine if the given document was written by a specific author or not. Some references of works tackling open set authorship attribution problem are shown in Koppel et al. (2011) and in Badirli et al. (2019). However, we decided to focus our work in tackling the closed set authorship attribution problem, but we will leave some thoughts for future work in the correspondent chapter.

[3] *The Guardian*, described in chapter 4

CHAPTER 3

Text characteristics analysis

The main problem in working with language processing is that machine learning algorithms cannot work on the raw text directly. So, we need some feature extraction techniques to convert text into a matrix(or vector) of features.

In order to identify the authorship of an unknown text document using machine learning the document needs to be quantified first. The simple and natural way to characterize a document is to consider it as a sequence of tokens grouped into sentences where each token can be one of the three: word, number, punctuation mark. To quantify the overall writing style of an author, stylometric features are defined and studied in different domains. Mainly, computations of stylometric features can be categorized into five groups as lexical, character, semantic, syntactic, and application specific features. Lexical and character features mainly considers a text document as a sequence of word tokens or characters, respectively. This makes it easier to do computations comparing to other features. On the other hand, syntactic and semantic features require deeper linguistic analysis and more computation time. Application specific features are defined based on the text domains or languages. These five features are studied and the methods to extract them are also provided for interested readers. Moreover there's a sixth characteristic regarding only hand-written text, which was used for years in the past and it's still studied nowadays [31], which is the *graphological analysis*. Although the problem of recognition of handwriting text is still far from its final solution, in this work, we will focus only on the first 5 characteristics because the main focus since digitalization era has been on studies of digital text characteristics analysis.

3.1 Character Features

Based on these features a sentence consists of a characters sequence. Some of the character-level features are alphabetic characters count, digit characters count, uppercase and lowercase character counts, letter and character n-gram frequencies. This type of feature extraction techniques has been found quite useful to quantify the writing style.[11] A more practical approach in character-level features are the extraction of n-gram characters. This procedure of extracting such features are language independent and requires less complex toolboxes. On the other hand, comparing to word n-grams approach the dimensional of these approaches are vastly increased and it has a curse of dimensional problem. A simple way of explaining what a character n-grams could be with the following example: assume that a word “thesis” is going to be represented by 2-gram characters. So, the resulting sets of points will be “th”, ”he”, ”es”, ”si”, ”is”.
A simple python algorithm is shown in 3.1:

```
def get_char_n_gram(text, n=2):
    """Convert text into character n-grams list"""
    return [text[i:i+n] for i in range(len(text)-n+1)]

# Examples character 2-grams

print(get_char_n_gram("thesis"))
>>Out: ['th', 'he', 'es', 'si', 'is']

print(get_char_n_gram("student", n=3))
>>Out: ['stu', 'tud', 'ude', 'den', 'ent']
```

Code Listing 3.1: Split word into character n-grams, parametric on n

In [37] have been identified 10 character n-grams categories, being proven as the most successful feature in both single-domain and cross-domain Authorship Attribution.
This 10 categories are grouped into 3 groups: Affix n-grams, Word n-grams, Punctuation n-grams.

3.1.1 Affix n-grams

Character n-grams are generally too short to represent any deep syntax, but some of them can reflect morphology to some degree. In particular, the following affix-like features are extracted by looking at n-grams that begin or end a word:

- **prefix**: A character n-gram that covers the first n characters of a word that is at least n+1 characters long.

- **suffix**: A character n-gram that covers the last n characters of a word that is at least n + 1 characters long.
- **space-prefix**: A character n-gram that begins with a space.
- **space-suffix**: A character n-gram that ends with a space.

3.1.2 Word n-grams

While character n-grams are often too short to capture entire words, some types can capture partial words and other word-relevant tokens. There's a distinction among:

- **whole-word**: A character n-gram that covers all characters of a word that is exactly n characters long.
- **mid-word**: A character n-gram that covers n characters of a word that is at least n + 2 characters long, and that covers neither the first nor the last character of the word.
- **multi-word**: N -grams that span multiple words, identified by the presence of a space in the middle of the n-gram.

3.1.3 Punctuation n-grams

The main stylistic choices that character n-grams can capture are the author's preferences for particular patterns of punctuation. The following features characterize punctuation by its location in the n-gram.

- **beg-punct**: A character n-gram whose first character is punctuation, but middle characters are not.
- **mid-punct**: A character n-gram with at least one punctuation character that is neither the first nor the last character.
- **end-punct**: A character n-gram whose last character is punctuation, but middle characters are not.

In [37] they've observed that in their data almost 80% of the n-grams in the punct-beg and punct-mid categories contain a space. They stated that "this tight coupling of punctuation and spaces is due to the rules of English orthography: most punctuation marks require a space following them". The 20% of n-grams that have punctuation but no spaces correspond mostly to the exceptions to this rule: quotation marks, mid-word hyphens, etc. They've conducted an experiment on RCV1 dataset both the *CCAT_10* *split* and the *CCAT_50 split.* They've also used the Guardian dataset as a cross-domain authorship attribution experiment. In their work they stated that for the single-domain the top four categories for authorship attribution are: *prefix*, *suffix*, *space-prefix* and *mid-word.* On the other hand, for cross-domain authorship attribution the top four categories have been proven to be: *prefix*, *space-prefix*, *beg-punct* and *mid-punct.* For both single-domain and cross-domain authorship attribution, *prefix* and *space-prefix* are strong features, and are generally better than the *suffix* features, perhaps because authors have more control over prefixes in English, while suffixes are often obligatory for grammatical reasons. For cross-domain authorship attribution, *beg-punct* and *mid-punct* they found to be the top features, likely because an author's use of punctuation is consistent even when the topic changes. For single-domain authorship attribution, *mid-word* was also a good feature, probably because it captured lexical information that correlates with authors' preferences towards writing about specific topic.

3.2 Lexical Features

Lexical features relate to the words or vocabulary of a language. It is the very plain way of representing a sentence structure that consists of words, numbers, punctuation marks. These features are very first attempts to attribute authorship in earlier studies [9], [2], [42]. The main advantage of Lexical features is that it is universal and can be applied to any language easily. These features consist of bag of words representation, word N-grams, vocabulary richness, number of punctuation marks, average number of words in a sentence, and many more. Even though the number of lexical features can vary a lot, not all of them are good for every authorship attribution problem. That is why, it is important to know how to extract these features and try out different combinations on different classifiers.

3.2.1 Bag of Words

It is the representation of a sentence with frequency of words. It is a simple and efficient solution but it disregards word-order information. At the very beginning, we applied this representation to our datasets, using the already implemented $CountVectorizer$ from $sklearn.feature_extraction.text$. We gave this hyper-parameter to the function:

- **max_df**=0.8
- **max_features**=10000
- **min_df**=0.02
- **ngram_range**=(1, 2)

In the approach, for each text fragment the number of instances of each unique word is found to create a vector representation of word counts. We capped the max number of features to 10'000 words and discarding the words with a higher frequency of 0.8 across the document and a lower frequency of 0.02. We've also taken into account both single word and word bi-grams. In order to give the reader a better perspective of the impact of this process for the task of authorship attribution, we choose two among the top ten most prolific authors in the RCV1 dataset: *David Lawder* and *Alexander Smith*. In 3.1 we can see the number of documents written by the two selected authors in the RCV1 corpus for the *CCAT* topic.

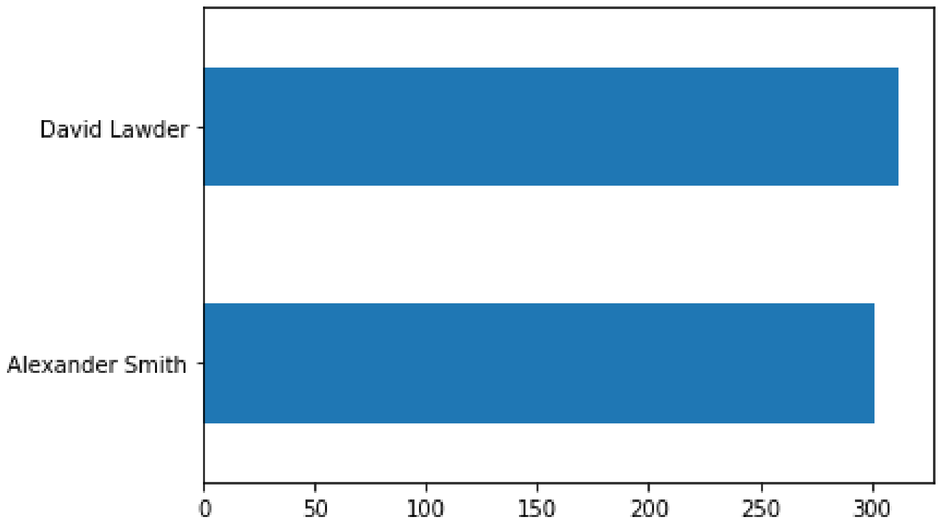

Figure 3.1: The number of documents written by David Lawder and Alexander Smith in the Reuters Corpus for the *CCAT* topic.

With a simple snippet of python code shown in 3.2, we can get the most common words for an author across all the documents we gathered in the dataset.

```
from collections import Counter

def get_most_common_words_in_df(df, n=20):
  """Split a collection of document in single word and order by
                                 frequencies across all documents"""
  most_common_words = Counter(" ".join(df["articles"]).split()).
                                 most_common(n)
  return most_common_words
```

```
# Examples

# 1. Top 20 words with their frequency for every document written by
                                 David Lawder
```

```
print(get_most_common_words_in_df(dataset[dataset['author']=='David
                                     Lawder']))
>>Out: [('the', 7844), ('to', 5133), ('of', 3875), ('and', 3746), ('a
                                     ', 3719), ('in', 3552), ('said',
                                     1749), ('that', 1720), ('for', 1574
                                     ), ('GM', 1453), ('on', 1286), ('at
                                     ', 1267), ('its', 1153), ('The',
                                     1098), ('with', 990), ('is', 957),
                                     ('it', 897), ('will', 875), ('by',
                                     868), ('from', 824)]

# 2. Top 20 words without their frequency for every document written
                                     by David Lawder

print([m[0] for m in get_most_common_words_in_df(dataset[dataset['
                                     author']=='David Lawder'])])
>>Out: ['the',
    'to',
    'of',
    'and',
    'a',
    'in',
    'said',
    'that',
    'for',
    'GM',
    'on',
    'at',
    'its',
    'The',
    'with',
    'is',
    'it',
    'will',
    'by',
    'from']
```

Code Listing 3.2: Top 20 most common words in a document or a collection of document by the same author

As expected top words are determiners that every writer use while constructing an English sentence. For example, for *David Lawder* top 20 words are *"the, to, of, and, a, in, said, that, for, GM, on, at, its, The, with, is, it, will, by, from"* but for *Alexander Smith* they are *"the, of, to, and, a, in, said, was, for, on, it, had, by, be, its, is, with, would, that, as"* in decreasing order. Even though the two sets are mostly the same, the

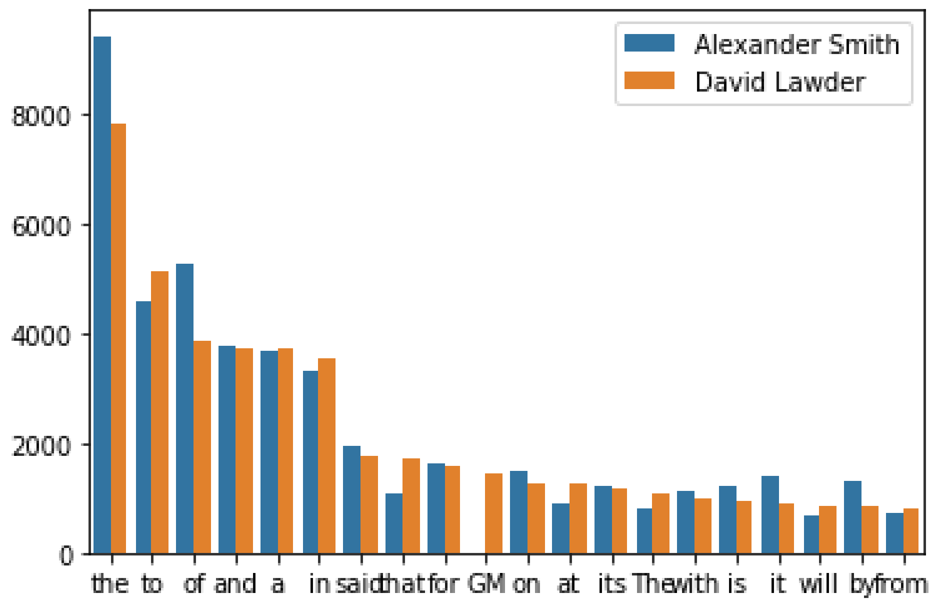

Figure 3.2: Frequency usage of the top 20 words used in the RCV1 corpus by David Lawder, compared to the frequencies of the same words in the corpus by Alexander Smith

orders and the frequency are different for most authors.

The main assumption with authorship attribution problems is that every authors word usage and content differs and based on these differences the work of one author can be differentiated from the other. In order to illustrate this assumption, in 3.2 we can see the top 20 words in the RCV1 corpus in the document written by David Lawder, compared to the same words in the documents of Alexander Smith.

In figure 3.3 & 3.4 we plotted using the Word Cloud identikit of David Lawder and Alexander Smith, generated by every document written by them in the RCV1 corpus.

3.2.2 Word N-grams

It is a type of probabilistic language model for predicting the next item in the form of a (n-1) order. Considering n-grams are useful since Bag of words miss out the word order when considering a text. For example, a verb "take on" can be missed out by Bag of words representation which considers "take" and "on" as two separate words. N-gram also establishes the approach of "Skip-gram" language model. An N-gram is a consecutive sub-sequence of length N of some sequence of sentence while a Skip-gram is a N-length sub-sequence where the components occur at a distance of at most k from each other [29]. In order to extract N-grams from a given text data a simple snippet of code is shown in 3.3, tested for the word grams on the documents written by David Lawder and Alexander Smith of the RCV1 corpus data. No pre-processing on the dataset, such as discarding stopwords, has been done while constructing the N-grams. For David Lawder "the, of the, General Motors Corp." are the most occurrent uni-gram, bi-gram and three-gram

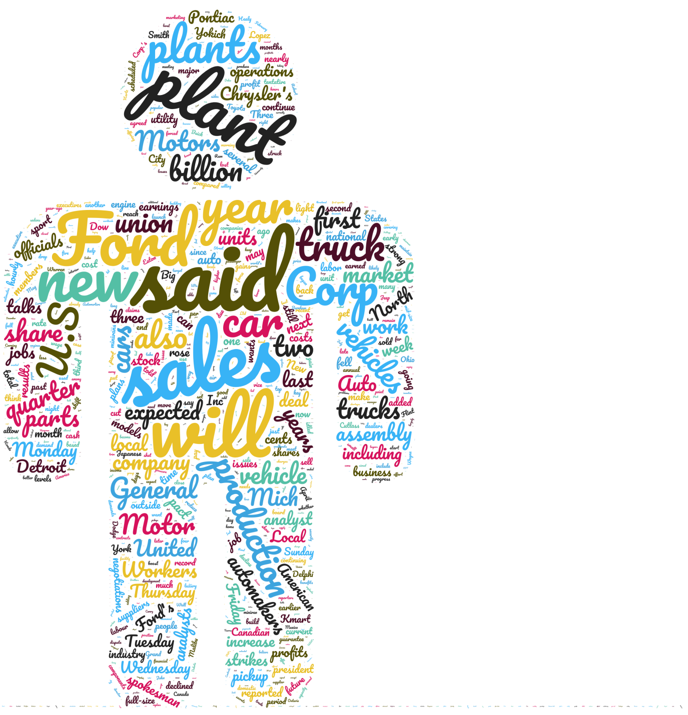

Figure 3.3: Image Generated on for every word in RCV1 corpus data for the documents written by David Lawder

respectively whilst in Alexander Smith documents they are "the, of the, said it would".

Code Listing 3.3: Get the top 3 most common grams in the corpus, for uni-grams, bi-grams and three-grams

```
from collections import Counter
from nltk import ngrams

def get_Xigram(text, n):
    """Get n-grams for a given text. The number of grams are controlled
                                by parameter n"""
    return list(ngrams(text.split(), n))

def get_top_3_gram(df):
    """Return a list of three elements, each one containing the top 3
                               most common grams in the corpus
                               given as a Dataframe parameter. The
                                three elements corresponds to the
                               uni-gram, bi-grams and three-grams.
                               """
    result = []
    for i in range(1,4):
        result.append(Counter(get_Xigram(" ".join(df["articles"]), i)).
                               most_common(3))
    return result
```

Figure 3.4: Image Generated on for every word in RCV1 corpus data for the documents written by Alexander Smith

```
print(get_top_3_gram(df_david_lawder))
print(get_top_3_gram(df_alexander_smith))
```

Table 3.1: Top 3 uni-grams, bi-grams and three-grams by David Lawder and Alexander Smith in the RCV1 corpus data

Author	Uni-gram	Bi-gram	Three-gram
David Lawder	the	of the	General Motors Corp.
David Lawder	to	in the	United Auto Workers
David Lawder	of	for the	Ford Motor Co.
Alexander Smith	the	of the	said it would
Alexander Smith	of	in the	the end of
Alexander Smith	to	said the	a result of

Table 3.1 contains top 3 uni-grams, bi-grams, three-grams from the authors David Lawder and Alexander Smith.

3.2.3 Vocabulary Richness

It is also referred as vocabulary diversity. It attempts to quantify the diversity of the vocabulary text. It is simply the ratio of V/N where V refers to the total number of unique tokens and N refers to the total number of tokens in the considered texts [41]. As an example, we applied this feature to the RCV1 CCAT_10 dataset[1]. A snippet of the code to extract such feature, is shown in 3.4.

Code Listing 3.4: Calculate vocabulary richness in RCV1 CCAT10 dataset

```python
tmp_df = dataset
tmp_df["num_unique_words"] = tmp_df["articles"].apply(lambda x: len
                                (set(str(x).split()))/len(str(x).
                                split()))

objects = {}
for author_i in tmp_df.author.unique():
  objects[author_i] = sum(tmp_df[tmp_df.author==author_i]['
                                num_unique_words'])/len(tmp_df[
                                tmp_df.author==author_i])

plt.bar(range(len(objects)), list(objects.values()), align='center'
                                )
plt.xticks(range(len(objects)), list(objects.keys()))
ax = plt.gca()
plt.setp(ax.get_xticklabels(), rotation=30, horizontalalignment='
                                right')
plt.show()
```

For this portion of the dataset, has been found the lowest vocabulary richness in *Marcel Michelson* and the highest in *Jim Gilchrist*. Overall distribution of vocabulary richness is plotted in Figure 3.5.

3.2.4 Stylometric features

These are features such as number of sentences in a text piece, number of words in a text piece, average number of words in a sentence, average word length in a text piece, number of periods, number of exclamation marks, number of commas, number of colons, number of semicolons, number of incomplete sentences, number of uppercase, title case, camel case, lower case letters. We computed these features comparing the stylometric differences for the documents belonging to David Lawder and the ones of Alexander Smith in the RCV1 corpus. Overall distribution of some of the features introduced here (such as: *number of punctuation, number of words upper case, number of words title, average length of the word, number of stopwords*) are applied and the resulting density measures are calculated for each of the two authors and shown in Table 3.2. Among these

[1]The top ten most prolific authors in the RCV1 corpus, selecting the documents belonging to the *CCAT* topic

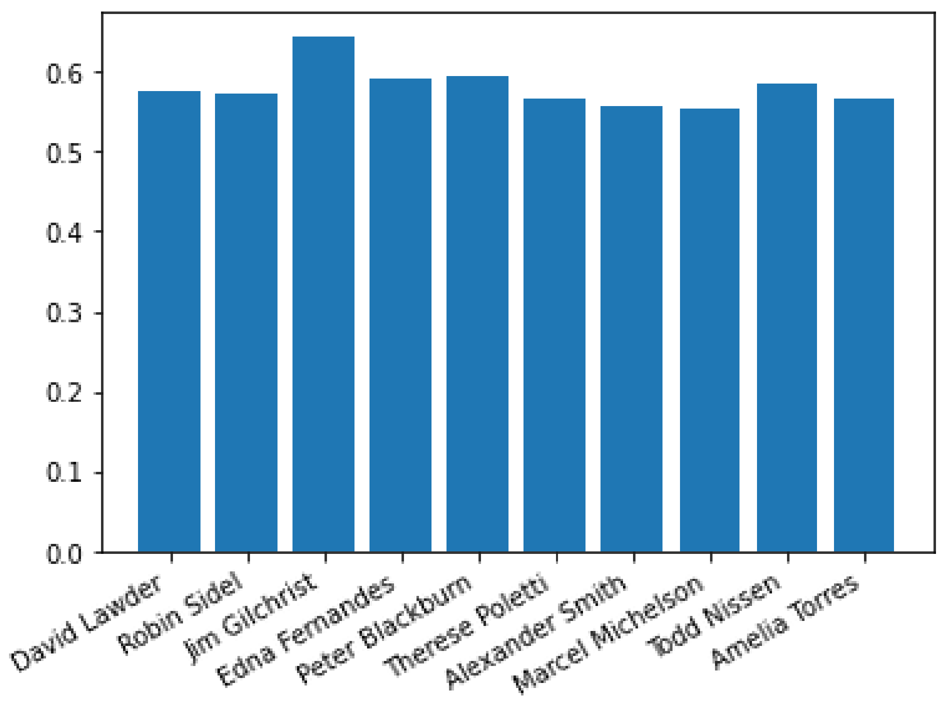

Figure 3.5: Bar Plot of vocabulary richness of RCV1 CCAT_10 authors across all their documents

five features introduced, number of punctuations and number of stop words usage varies the most among the authors and hence they can be better distinguisher comparing to other feature sets.

Table 3.2: Comparing some stylometric features between David Lawder and Alexander Smith calculated on the documents in the RCV1 corpus data and normalized by the number of documents

Stylometric Feature	David Lawder	Alexander Smith
Number of punctuation symbols	101.78	88.73
Number of uppercase words	12.59	9.89
Number of title words	76.59	68.30
Word length mean	5.09	5.11
Number of stopwords	191.38	234.97

3.2.5 Function Words

Function words are the words that have little meaning on their own but they're necessary to construct a sentence in English language. They express grammatical relationships among other words within a sentence, or specify the attitude or mood of the speaker. Some of the examples of function words might be prepositions, pronouns, auxiliary verbs, conjunctions, grammatical articles. Words that are not functions words are called as content words and they can also be studied to further analysis the use case in the authorship attribution problems. The search for a single invariant measure of textual style was natural in the early stages of stylometric analysis, but with the development of more sophisticated multivariate analysis techniques, larger sets of features could be considered. Among the earliest studies to use multivariate approaches was that of Mosteller and Wallace (1964), who considered distributions of FWs. The reason for using FWs in preference to others is that we do not expect their frequencies to vary greatly with the topic of the text, and hence, we may hope to recognize texts by the same author on different topics. It also is unlikely that the frequency of FW use can be consciously controlled, so one may hope that use of FWs for attribution will minimize the risk of being deceived [8]. Many studies since that of Mosteller and Wallace (1964) have shown the efficacy of FWs for authorship attribution in different scenarios [2], [3], [18], [48], confirming the hypothesis that different authors tend to have different characteristic patterns of FW use. Typical modern studies using FWs in English use lists of a few hundred words, including pronouns, prepositions, auxiliary and modal verbs, conjunctions, and determiners. Numbers and interjections are usually included as well since they are essentially independent of topic, although they are not, strictly speaking, FWs. Results of different studies using somewhat different lists of FW have been similar, indicating that the precise choice of FW is not crucial. Discriminators built from FW frequencies often perform at levels competitive with those constructed from more complex features.

3.2.6 Tf-Idf

It stands for term frequency-inverse document frequency. It is often used as a weight in feature extraction techniques. The reason why Tf-Idf is a good feature can be explained in an example. Let's assume that a text summarization needs to be done using few keywords. One strategy is to pick the most frequently occurring terms meaning words that have high term frequency (*tf*). The problem here is that, the most frequent word is a less useful metric since some words like 'a', 'the' occur very frequently across all documents. Hence, a measure of how unique a word across all text documents needs to be measured as well (*idf*). Hence, the product of *tf* x *idf* (3.3) of a word gives a measure of how frequent this word is in the document multiplied by how unique the word is with respect to the entire corpus of documents. Words with a high tf-idf score are assumed to provide the most information about that specific text [41].

$$TF(t) = \frac{\text{Number of times term t appears in a document}}{\text{Total numbers of terms in the document}} \tag{3.1}$$

$$IDF(t) = log_e(\frac{\text{Total number of documents}}{\text{Number of documents with term t in it}}) \tag{3.2}$$

$$Tf - Idf = TF(t) * IDF(t) \tag{3.3}$$

As an example, we buil a Tf-Idf model by considering documents alone within the text corpus for the authors David Lawder and Alexander Smith. In the model, not only the single forms of word tokens but their n-grams are considered as well. Table 3.3 provides the top 5 words with highest Tf-Idf scores for the two authors. Comparing between Table 3.1 and Table 3.3 new meaningful words have appeared that could serve as a new feature for each author such as “dow, kmart, coupe” for David Lawder or “hsbc, pensions, panel” for Alexander Smith.

Table 3.3: Top 5 TFIDF words n-grams by David Lawder and Alexander Smith in the RCV1 corpus data

Author	Token	Value	Author	Token	Value
David Lawder	**dow**	**0.702**	Alexander Smith	**hsbc**	**0.697**
David Lawder	kmart	0.658	Alexander Smith	pensions	0.601
David Lawder	south	0.559	Alexander Smith	bzw	0.592
David Lawder	coupe	0.539	Alexander Smith	panel	0.579
David Lawder	bags	0.517	Alexander Smith	read	0.570

3.3 Syntactic Features

For certain text grammatical and syntactic features could be more useful compared to lexical or character level features. However, this kind of feature extraction techniques requires specific usage of Part of Speech taggers. Some of these features consider the frequency of nouns, adjectives, verbs, adverbs, prepositions, and tense information (past tense,etc). The motivation for extracting these features is that authors tend to use similar syntactic patterns unconsciously [41]. Some researchers are also interested in exploring different dialects of the same language and building classifiers based on features derived from syntactic characteristic of the text. One great example is the work that aims to discriminate between texts written in either the Netherlandic or the Flemish variant of the Dutch language [44]. The feature set in this case consists of lexical, syntactic and word-n grams build on different classifiers and F1-score has been recorded for each cases. Employed syntactic features are function words ratio, descriptive words to nominal words ratio personal pronouns ratio, question words ratio, question mark ratio, exclamation mark ratio [44].

3.4 Semantic Features

Features that we discussed so far aim at analyzing the structural concept of a text such. Semantic feature extraction from text data is a bit challenging. That might explain why there is limited work in this area. One example is the work of Yang who has proposed combination of lexical and semantic features for short text classification [46]. Their approach consists of choosing a broader domain related to target categories and then applying topic models such as *Latent Dirichlet Allocation* to learn a certain number of topics from longer documents. The most discriminative feature words of short text are then mapped to corresponding topics in longer documents [46]. Their experimental results show significant improvements compared to other related techniques studying short text classification. Positivity, neutrality, and negativity index, and synonym usage preference are good examples of semantic features. Distributed representation of words, Word2Vec, is also an attempt to extract and represent the semantic features of a word, sentence, and paragraph [28]. The usage of Word2Vec in authorship attribution tasks has not yet been studied explicitly. Due to the application domain dependency of Word2Vec features their usage will be introduced when discussing application specific feature sets.

3.4.1 Positivity and Negativity index

In order to understand the general mood and the preference of positive and negative sentence structure in each author's work, a positivity and negativity score has been calculated for the authors *David Lawder* and *Alexander Smith* for the documents in the RCV1 corpora. The code is shown in 3.5, whereas in 3.6 we can see the results that points out Alexander Smith writing more negative articles than David Lawder. Both of the authors wrote the majority of the articles classified as positive than negative.

Code Listing 3.5: Compute sentence Positivity and Negativity scores

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')
# Pre-Processing
SIA = SentimentIntensityAnalyzer()
# Applying Model, Variable Creation
sentiment = pd.concat([df_david_lawder, df_alexander_smith])
sentiment['polarity_score']=sentiment.articles.apply(lambda x:SIA.
                                    polarity_scores(x)['compound'])
sentiment['neutral_score']=sentiment.articles.apply(lambda x:SIA.
                                    polarity_scores(x)['neu'])
sentiment['negative_score']=sentiment.articles.apply(lambda x:SIA.
                                    polarity_scores(x)['neg'])
```

```
sentiment['positive_score']=sentiment.articles.apply(lambda x:SIA.
                                    polarity_scores(x)['pos'])
sentiment['sentiment']=''
sentiment.loc[sentiment.polarity_score>0,'sentiment']='POSITIVE'
sentiment.loc[sentiment.polarity_score==0,'sentiment']='NEUTRAL'
sentiment.loc[sentiment.polarity_score<0,'sentiment']='NEGATIVE'
# Normalize for Size
auth_sent= sentiment.groupby(['author','sentiment'])[['articles']].
                                    count().reset_index()
for x in ['David Lawder', 'Alexander Smith']:
auth_sent.articles[auth_sent.author == x] = (auth_sent.articles[
                                    auth_sent.author == x]/\
auth_sent[auth_sent.author ==x].articles.sum())*100
ax= sns.barplot(x='sentiment', y='articles',hue='author',data=
                                    auth_sent)
ax.set(xlabel='Author', ylabel='Sentiment Percentage')
ax.figure.suptitle("Author by Sentiment", fontsize = 24)
plt.show()
```

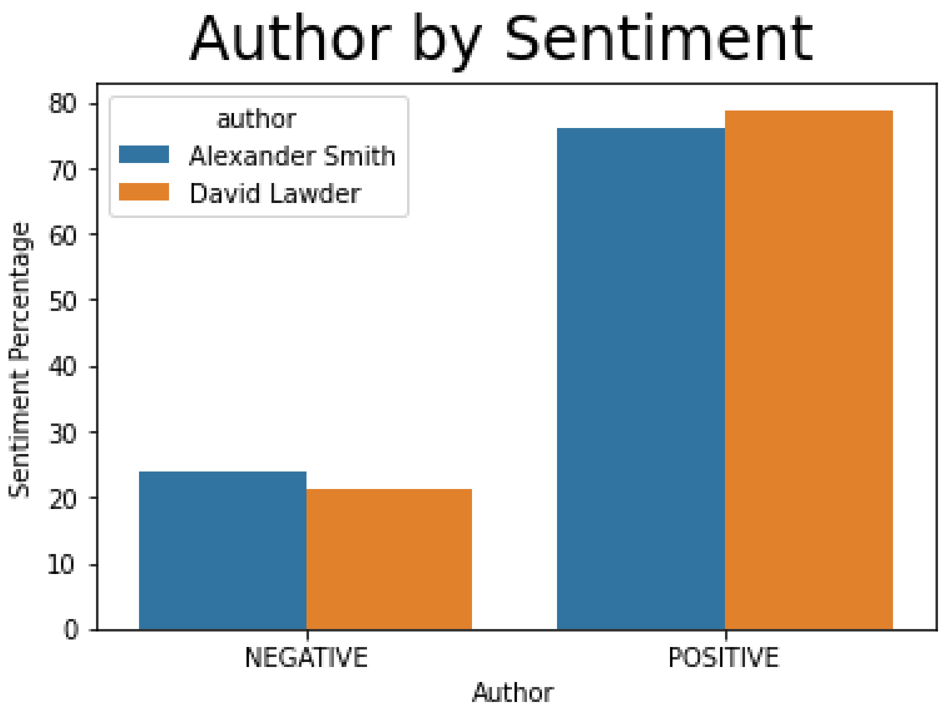

Figure 3.6: Bar Plot of sentiment analysis for 2 of the authors in the RCV1 CCAT_10 corpora across all their documents

3.5 Application Specific Features

When the application domain of the authorship attribution problems are different such as email messages or online forum messages, author style can be better characterized using structural, content specific, and language specific features. In such domains, the use of greetings and farewells, types of signatures, use of indentation, paragraph lengths, font color, font size could be good features [41].

3.5.1 Vector embeddings of words (Word2Vec)

Word2Vec[2] leverages the context of the target words. Essentially, we want to use the surrounding words to represent the target words with a Neural Network whose hidden layer encodes the word representation. Similar words are close to each other in the vector space. For example, it was shown in [30] that $vector[King] - vector[Man] + vector[Woman]$ results in the vector that is closest to the representation of the $vector[Queen]$. Figure 7.4 shows this representation in a simple way. The ways to make use of Word2Vec in the dataset is various. For example, a Word2Vec model can either be built by considering every authors text data separately, or can be imported using previously trained word vectors on other large text corpus. It can, then, be plotted into two dimensional vector space by using dimensionality reduction techniques (TSNE, for example[3]). We can also make use of pre-trained word vectors of Glove to see the difference of usages in such words between an author and a pre- trained word vector. Moving with the idea of training Word2Vec per author, one can also do a cosine distance measure for the same word or same sentence. There are two types of Word2Vec, Skip-gram and Continuous Bag of Words (CBOW). I will briefly describe how these two methods work in the following paragraphs.

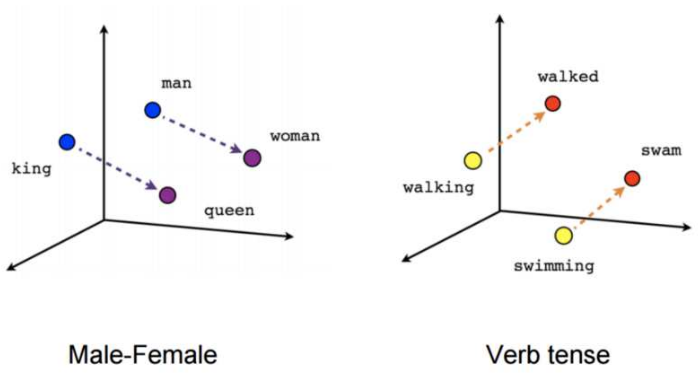

Figure 3.7: Examples of Word2Vec representation with vector distance

[2]`https://code.google.com/archive/p/word2vec/`

[3]t-Distributed Stochastic Neighbor Embedding (t-SNE) is a technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets

3.5.1.1 Skip-gram

For skip-gram, the input is the target word, while the outputs are the words surrounding the target words. For instance, in the sentence “I have a cute dog”, the input would be “a”, whereas the output is “I”, “have”, “cute”, and “dog”, assuming the window size is 5. All the input and output data are of the same dimension and one-hot encoded. The network contains 1 hidden layer whose dimension is equal to the embedding size, which is smaller than the input/output vector size. At the end of the output layer, a softmax activation function is applied so that each element of the output vector describes how likely a specific word will appear in the context. In mathematics, the softmax function, or normalized exponential function is a generalization of the logistic function that *squashes* a K-dimensional vector z of arbitrary real values to a K-dimensional vector δ(z) of real values, where each entry is in the range (0,1) and all the entries add up to 1. The target is a (K-1)-dimensional space, so one dimension has been lost.

With skip-gram, the representation dimension decreases from the vocabulary size (V) to the length of the hidden layer (N). Furthermore, the vectors are more “meaningful” in terms of describing the relationship between words. The vectors obtained by subtracting two related words sometimes express a meaningful concept such as gender or verb tense, as shown in the following figure (dimensionality reduced).

3.5.1.2 CBOW

Continuous Bag of Words (CBOW)[4] is very similar to skip-gram, except that it swaps the input and output. The idea is that given a context, we want to know which word is most likely to appear in it.
The biggest difference between Skip-gram and CBOW is that the way the word vectors are generated. For CBOW, all the examples with the target word as target are fed into the networks, and taking the average of the extracted hidden layer. For example, assume we only have two sentences, “He is a nice guy” and “She is a wise queen”. To compute the word representation for the word “a”, we need to feed in these two examples, “He is nice guy”, and “She is wise queen” into the Neural Network and take the average of the value in the hidden layer. Skip-gram only feed in the one and only one target word one-hot vector as input.

It is claimed that Skip-gram tends to do better in rare words. Nevertheless, the performance of Skip-gram and CBOW are generally similar.

[4] `https://iksinc.online/tag/continuous-bag-of-words-cbow/`

3.5.2 Vector embeddings of documents (Doc2Vec)

Distributed word representation in a vector space (word embeddings) is a novel technique that allows to represent words in terms of the elements in the neighborhood. Distributed representations can be extended to larger language structures like phrases, sentences, paragraphs and documents. The capability to encode semantic information of texts and the ability to handle high-dimensional datasets are the reasons why this representation is widely used in various natural language processing tasks such as text summarization, sentiment analysis and syntactic parsing [23].

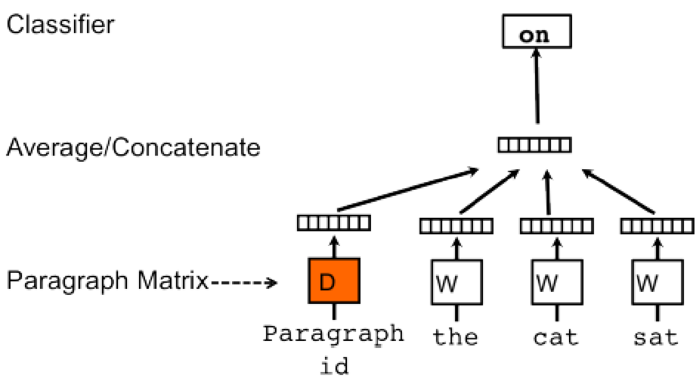

Figure 3.8: Paragraph Vector-Distributed Memory model

The goal of doc2vec is to create a numeric representation of a document, regardless of its length. Unlike words, documents do not come in logical structures such as words, so the another method has to be found. The concept that Mikilov and Le have used was simple, yet clever: they have used the word2vec model, but instead of using just words to predict the next word, we also added another feature vector, which is document-unique. When training the word vectors W, the document vector D is trained as well, and in the end of training, it holds a numeric representation of the document. The model above is called *Distributed Memory version of Paragraph Vector* (PV-DM) 3.8. It acts as a memory that remembers what is missing from the current context — or as the topic of the paragraph. While the word vectors represent the concept of a word, the document vector intends to represent the concept of a document. As in word2vec, another algorithm, which is similar to skip-gram may be used *Distributed Bag of Words version of Paragraph Vector* (PV-DBOW).

As we can see in 3.9, this algorithm is actually faster (as opposed to word2vec) and consumes less memory, since there is no need to save the word vectors.

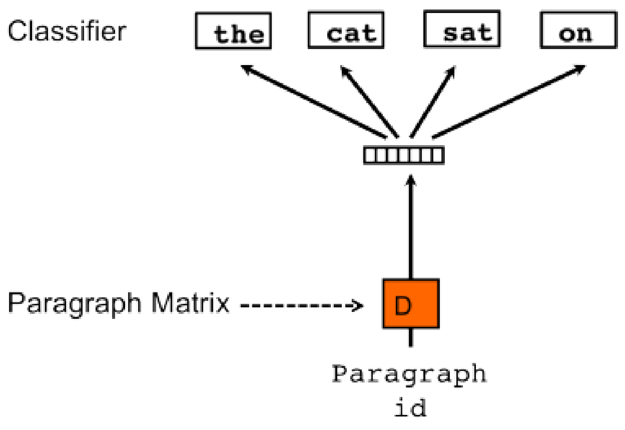

Figure 3.9: Paragraph Vector-Distributed Bag of Words model

Bibliography

[1] Ahmed Abbasi and Hsinchun Chen. Applying authorship analysis to extremist-group web forum messages. *IEEE Intelligent Systems*, 20(5):67–75, 2005.

[2] Shlomo Argamon and Shlomo Levitan. Measuring the usefulness of function words for authorship attribution. In *Proceedings of the 2005 ACH/ALLC Conference*, pages 4–7, 2005.

[3] Harald Baayen, Hans van Halteren, Anneke Neijt, and Fiona Tweedie. An experiment in authorship attribution. In *6th JADT*, volume 1, pages 69–75. Citeseer, 2002.

[4] Sarkhan Badirli, Mary Borgo Ton, Abdulmecit Gungor, and Murat Dundar. Open set authorship attribution toward demystifying victorian periodicals. *arXiv preprint arXiv:1912.08259*, 2019.

[5] John Burrows. 'delta': a measure of stylistic difference and a guide to likely authorship. *Literary and linguistic computing*, 17(3):267–287, 2002.

[6] JK Chambers, P Trudgill, and Natalie Schilling-Estes. The handbook of language variation and change (2nd). *Victoria: Blackwell Publishing*, 2004.

[7] Carole E Chaski. Who's at the keyboard? authorship attribution in digital evidence investigations. *International journal of digital evidence*, 4(1):1–13, 2005.

[8] Cindy Chung and James W Pennebaker. The psychological functions of function words. *Social communication*, 1:343–359, 2007.

[9] Neal Fox, Omran Ehmoda, and Eugene Charniak. Statistical stylometrics and the marlowe-shakespeare authorship debate. *Proceedings of the Georgetown University Roundtable on Language and Linguistics (GURT), Washington, DC, USA*, 2012.

[10] Georgia Frantzeskou, Efstathios Stamatatos, Stefanos Gritzalis, and Sokratis Katsikas. Effective identification of source code authors using byte-level information. In *Proceedings of the 28th international conference on Software engineering*, pages 893–896, 2006.

[11] Jack Grieve. Quantitative authorship attribution: An evaluation of techniques. *Literary and linguistic computing*, 22(3):251–270, 2007.

[12] H van Halteren. Linguistic profiling for authorship recognition and verification. 2004.

[13] Graeme Hirst and Ol'ga Feiguina. Bigrams of syntactic labels for authorship discrimination of short texts. *Literary and Linguistic Computing*, 22(4):405–417, 2007.

[14] David I Holmes. The evolution of stylometry in humanities scholarship. *Literary and linguistic computing*, 13(3):111–117, 1998.

[15] David I Holmes and Fiona J Tweedie. Forensic stylometry: A review of the cusum controversy. *Revue Informatique et Statistique dans les Sciences Humaines*, 31(1): 19–47, 1995.

[16] Patrick Juola. *Authorship attribution*, volume 3. Now Publishers Inc, 2008.

[17] Mike Kestemont, Efstathios Stamatatos, Enrique Manjavacas, Walter Daelemans, Martin Potthast, and Benno Stein. Overview of the cross-domain authorship attribution task at pan 2019. In *CLEF (Working Notes)*, 2019.

[18] Moshe Koppel, Jonathan Schler, and Kfir Zigdon. Determining an author's native language by mining a text for errors. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 624–628, 2005.

[19] Moshe Koppel, Jonathan Schler, Shlomo Argamon, and Eran Messeri. Authorship attribution with thousands of candidate authors. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 659–660, 2006.

[20] Moshe Koppel, Jonathan Schler, and Shlomo Argamon. Computational methods in authorship attribution. *Journal of the American Society for information Science and Technology*, 60(1):9–26, 2009.

[21] Moshe Koppel, Jonathan Schler, and Shlomo Argamon. Authorship attribution in the wild. *Language Resources and Evaluation*, 45(1):83–94, 2011.

[22] Moshe Koppel, Jonathan Schler, Shlomo Argamon, and Yaron Winter. The "fundamental problem" of authorship attribution. *English Studies*, 93(3):284–291, 2012.

[23] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

[24] David D Lewis and Marc Ringuette. A comparison of two learning algorithms for text categorization. In *Third annual symposium on document analysis and information retrieval*, volume 33, pages 81–93, 1994.

[25] David Madigan, Alexander Genkin, David D Lewis, Shlomo Argamon, Dmitriy Fradkin, and Li Ye. Author identification on the large scale. In *Proceedings of the 2005 Meeting of the Classification Society of North America (CSNA)*, 2005.

[26] Yuval Marton, Ning Wu, and Lisa Hellerstein. On compression-based text classification. In *European Conference on Information Retrieval*, pages 300–314. Springer, 2005.

[27] S Michaelson and A Morton. The qsum plot. Technical report, Internal Report CSR-3, 1990.

[28] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

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

[30] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751, 2013.

[31] Leonid A Mironovsky, Alexander V Nikitin, Nina N Reshetnikova, and Nikolay V Soloviev. Graphological analysis and identification of handwritten texts. In *Computer Vision in Control Systems-4*, pages 11–40. Springer, 2018.

[32] Tom M Mitchell. Artificial neural networks. *Machine learning*, 45:81–127, 1997.

[33] Frederick Mosteller and David L Wallace. *Inference and disputed authorship: The Federalist.* Stanford Univ Center for the Study, 2007.

[34] Rebekah Overdorf and Rachel Greenstadt. Blogs, twitter feeds, and reddit comments: Cross-domain authorship attribution. *Proceedings on Privacy Enhancing Technologies*, 2016(3):155–171, 2016.

[35] Joseph Rudman. The state of authorship attribution studies: Some problems and solutions. *Computers and the Humanities*, 31(4):351–365, 1997.

[36] Conrad Sanderson and Simon Guenter. Short text authorship attribution via sequence kernels, markov chains and author unmasking: An investigation. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 482–491, 2006.

[37] Upendra Sapkota, Steven Bethard, Manuel Montes, and Thamar Solorio. Not all character n-grams are created equal: A study in authorship attribution. In *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 93–102, 2015.

[38] Roy Schwartz, Oren Tsur, Ari Rappoport, and Moshe Koppel. Authorship attribution of micro-messages. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1880–1891, 2013.

[39] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

[40] Efstathios Stamatatos. Author identification: Using text sampling to handle the class imbalance problem. *Information Processing & Management*, 44(2):790–799, 2008.

[41] Efstathios Stamatatos. A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology*, 60(3):538–556, 2009.

[42] Sean Stanko, Devin Lu, and Irving Hsu. Whose book is it anyway? using machine learning to identify the author of unknown texts. *Machine Learning Final Projects*, 2013.

[43] Antônio Theóphilo, Luís AM Pereira, and Anderson Rocha. A needle in a haystack? harnessing onomatopoeia and user-specific stylometrics for authorship attribution of micro-messages. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2692–2696. IEEE, 2019.

[44] Chris van der Lee and Antal van den Bosch. Exploring lexical and syntactic features for language variety identification. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 190–199, 2017.

[45] Carrington B Williams. Mendenhall's studies of word-length distribution in the works of shakespeare and bacon. *Biometrika*, 62(1):207–212, 1975.

[46] Lili Yang, Chunping Li, Qiang Ding, and Li Li. Combining lexical and semantic features for short text classification. *Procedia Computer Science*, 22:78–86, 2013.

[47] G Udny Yule. A test of tippett's random sampling numbers. *Journal of the Royal Statistical Society*, 101(1):167–172, 1938.

[48] Ying Zhao and Justin Zobel. Effective and scalable authorship attribution using function words. In *Asia Information Retrieval Symposium*, pages 174–189. Springer, 2005.

[49] George Kingsley Zipf. Selected studies of the principle of relative frequency in language. 1932.

[50] Sven Meyer Zu Eissen, Benno Stein, and Marion Kulig. Plagiarism detection without reference collections. In *Advances in data analysis*, pages 359–366. Springer, 2007.

APPENDIX A

Code

A.1 Dataset estraction

A.1.1 RCV1

We used as parameter depth 2 and this list of categories: *Salute* (i.e. health in italian), *Medicina* (i.e. medicine in italian), *Procedure mediche* (i.e. medical procedures in italian), *Diagnostica medica* (i.e. medical diagnostics in italian) and *Specialità medica* (i.e. medical specialty in italian). The language of the result pages is by default ”it” (i.e. italian).

A.1.2 GDELT

To retrieve a single document, we parsed the HTML of every Wikipedia page with HTMLParser subclassed with a customized class MLStripper, that stripped away all HTML tags.

```python
class MLStripper(HTMLParser):
    def __init__(self):
        super().__init__()
        self.reset()
        self.fed = []

    def handle_data(self, d):
        self.fed.append(d)

    def get_data(self):
        return ''.join(self.fed)
```

```
def strip_tags(html):
    s = MLStripper()
    s.feed(html)
    return s.get_data()
```

A.2 Model

A.2.1 Feature extraction

Code A.1 shows how to load the model from the pre-trained vectors.

Code Listing A.1: How to load Wikipedia pre-trained model

```
# word2vec model
from gensim.models import Word2Vec
model = Word2Vec.load('wiki_iter=5_algorithm=skipgram_window=10
   _size=300_neg-samples=10.m')
```

A.2.2 Train model

Code A.2 shows how to build the vocabulary for the model and train it.

Code Listing A.2: How to build vocabulary and train model

```
# build vocabulary and train model
model = gensim.models.Word2Vec(documents,
                               size=200,
                               window=10,
                               min_count=2,
                               workers=10)
model.train(documents,
            total_examples=len(documents),
            epochs=10)
```

A.2.3 Evaluation

Code A.3 shows how to plot words' vectors. We used pyplot from *matplotlib*[1] module and PCA from *sklearn.decomposition*[2] module.

[1]https://matplotlib.org/api/pyplot_api.html

[2]http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

Code Listing A.3: How to plot words' vectors

```
from matplotlib import pyplot
from sklearn.decomposition import PCA
def plot(model, words):
        X = model[model.wv.vocab]
        pca = PCA(n_components=2)
        result = pca.fit_transform(X)
        pyplot.scatter(result[:, 0], result[:, 1])
        for i, word in enumerate(words):
                pyplot.annotate((word[0], float(round(word[1],
                   2))), xy=(result[i, 0], result[i, 1]))
        pyplot.show()
```

List of Figures

List of Tables