Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

On Authorship Attribution

Relatore:
Chiar.mo Prof.
DANILO MONTESI

Presentata da:
Gabriele Calarota

Correlatore:
Dott.
FLAVIO BERTINI

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 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 [10], 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.[5] 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 [11] 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 [11] 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 [4], [1], [13]. 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.

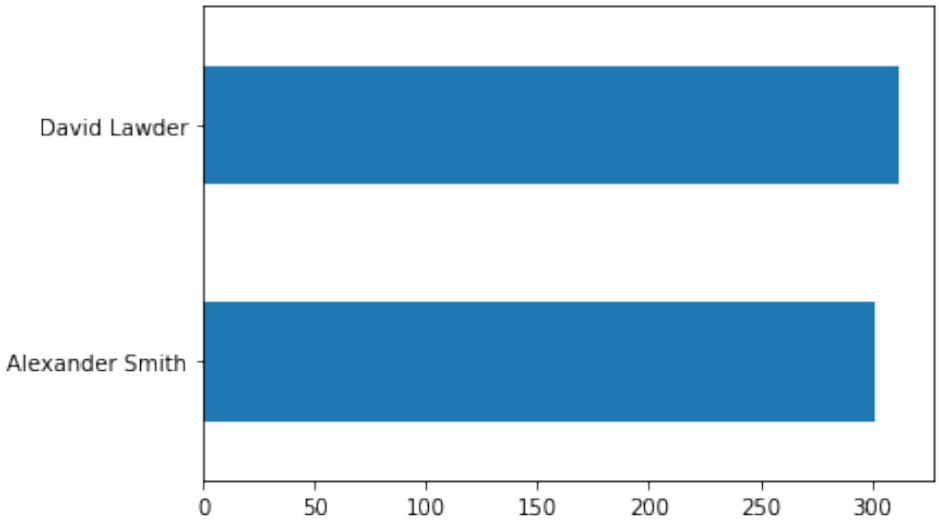

Figure 3.1: The number of documents written by David Lawder and Alexander Smith in the Reuters Corpus for the *CCAT* topic.

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

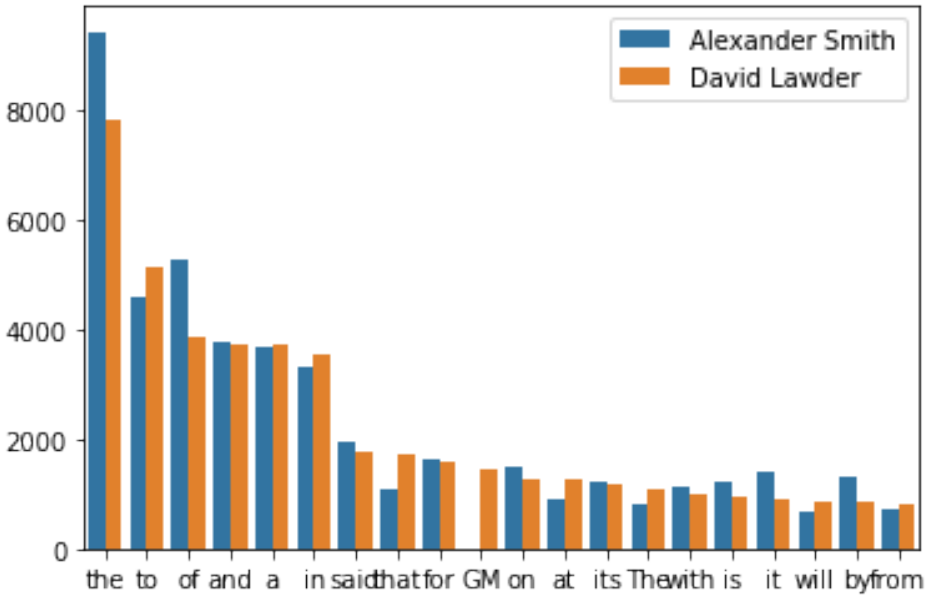

Figure 3.2: Frequency usage of the top 20 words used in the RCV1 corpus by David Lawder, compared to the frequencies of the same words in the corpus by Alexander Smith

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 [9]. 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 [12]. 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

```
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

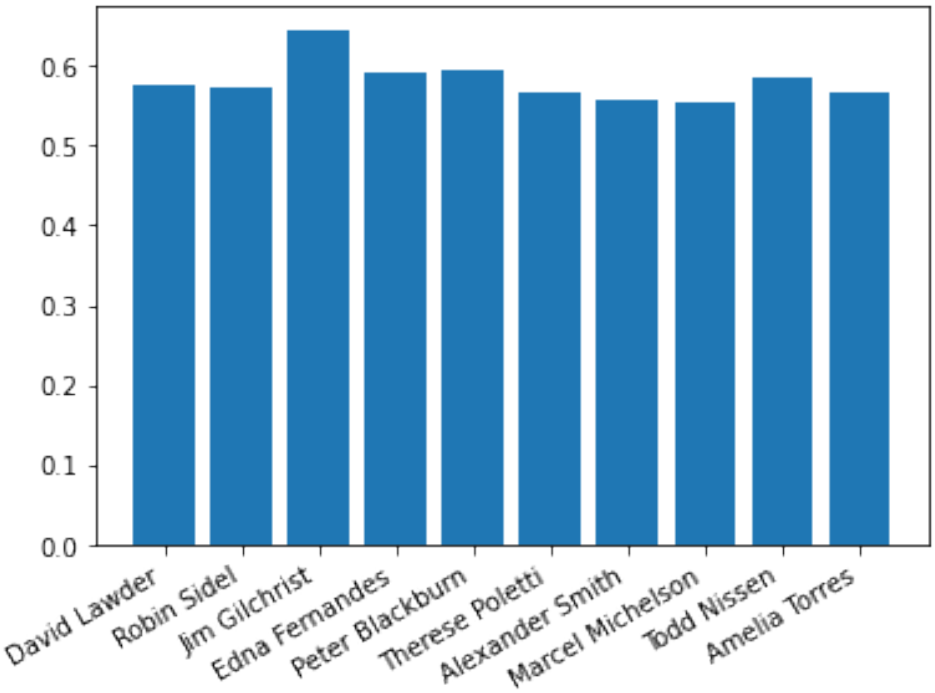

Figure 3.5: Bar Plot of vocabulary richness of RCV1 CCAT_10 authors across all their documents

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 [3]. Many studies since that of Mosteller and Wallace (1964) have shown the efficacy of FWs for authorship attribution in different scenarios [1], [2], [6], [16], 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 [12].

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

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

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

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 [12]. 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 [14]. 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 [14].

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 [15]. 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 [15]. 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 [8]. 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 model has been built. In the algorithm, the sentences that have positive polarity score have been labeled as positive and the negative polarity scored ones are labeled as negative.

3.4.2 Synonym Usage

The preference to use synonyms and antonyms in different text structure could be an identifiable feature in different tasks as well. However, extracting such features and modeling it could not be an easy task. The simple approach could be creating a domain knowledge where the pairs of synonyms and antonyms are paired. Then, a simple brute force approach can be used to find such words within a specific window size of sentences. Another approach is to employ the word vectors and represent a sentence with the average of all word vectors in the sentence. Using these two approaches an example model has been created. In the example model, given a synonym. its antonym set can be retrieved using NLTK wordnet library. Also, a similarity score can be calculated comparing the average vector forms of two sentences.

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 [12].

3.5.1 Vector embeddings of words (Word2Vec)

It gives the ability to represent a word in a vector dimension of your choosing. 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. We built a model based on profile based Word2Vec training and using TSNE to decrease it to two dimensions. In the model, our baseline approach is to extract A. Doyle and E. Wharton's text data and train Word2Vec on both of these authors text sets separately. Then, we have checked the word closeness for "listen" in both of these authors using 300 dimensional word vectors. Figure 4.10 shows the closest words in 2 dimensions for E. Wharton. The same comparison can also be done between 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. The measured cosine distance for A. Doyle and E. Wharton regarding the usage of "listen" is 0.094. In order to apply this strategy on sentence level, we can have a few ways to do so. One way is by simply taking the scaled average of Word2Vec vectors in the sentence. Another one is to employ Tf-Idf score of each word as a gain when calculating a sentence vector. We then take the scaled average of all word vectors in the text piece. For the simplicity, we only consider taking the average vector without Tf-Idf gain for now. In this case, "her lips were parted" has been compared for both A. Doyle and E. Wharton. The cosine distance has been recorded as 0.258 which is much larger than the distance for the word "listen". The reason is that "her lips were parted" is an exact phrase that is extracted from A. Doyle whereas "listen" is a common verb for both authors. The same comparison can also be done by considering the Glove's pretrained Word2Vec. The cosine distance for "listen" between A. Doyle and pretrained set is found as 0.015 whereas for E. Wharton, it is 0.012. As for "her lips were parted", the cosine difference for A. Doyle and pretrained set is -0.009, and for E. Wharton, it is 0.012. This implies that E. Wharton uses "listen" close to the pretrained set which was trained on large corpus of text data.

As for sentence comparison, the sentence average vector is closer to A. Doyle stating that this sentence is more likely to be written by A. Doyle then E. Wharton. The way to achieve this comparison criteria has been provided here for readers to move forward with this methodology.

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.

Distributed representations can be extended to model not only words, but also larger language structures like phrases, sentences and documents [7].

Bibliography

[1] 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.

[2] 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.

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

[4] 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.

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

[6] 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.

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

[8] 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.

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

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

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