Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Engenharia Informática

Dissertação

Authorship Attribution using Co-Occurrence Networks

David Laranjo Pinto

Orientador(es) | Lígia Maria Ferreira

Évora 2021

A dissertação foi objeto de apreciação e discussão pública pelo seguinte júri nomeado pelo Diretor da Escola de Ciências e Tecnologia:

Presidente	Irene Pimenta Rodrigues (Universidade de Évora)
Vogais	Carlos Correia Ramos (Universidade de Évora) (Arguente)
	Lígia Maria Ferreira (Universidade de Évora) (Orientador)

Évora 2021

Contents

List of Figures

List of Tables

Sumário

Atribuição de Autoria utlizando Redes de Co-Ocorrencia

Nesta tese é abordada a tarefa de Atribuição de Autoria como uma tarefa de classificação. As metodologias utilizadas representam textos em grafos. Destes, várias medidas são extraídas, sendo utilizadas como amostras para o classificador. Já existem alguns trabalhos que também se focam nesta metodologia. Esta tese foca-se num método que divide o texto em várias partes e trata cada uma como um grafo. Deste, são extraídas as medidas, que são tratadas como uma série temporal, da qual são extraídos momentos. Assim, os momentos compõem o vetor final, representativo de todo o texto. A partir da metodologia aqui descrita surgem mais duas variações. A primeira variação omite o passo das séries temporais, e, por consequência, as várias medidas de cada grafo são utilizadas diretamente como amostras. A segunda variação representa todo o texto como um só grafo. As metodologias são testadas com *corpus* em Inglês e Português, com número variado de textos.

Palavras chave: Atribuição de Autoria, Processamento de Lingua Natural, Grafos, Redes de co-ocorrencia, Classificação

Abstract

Authorship Attribution using Co-Occurrence Networks

This thesis approaches the task of Authorship Attribution as a classification task. This is done using methodologies that represent text documents in graphs, from which several measures are extracted, to be used as samples for the classifier. There have been some works that also focus on this methodology. This thesis focuses on a methodology which splits the texts in multiple parts and treats each as a separate graph, from which measures are extracted. Each graph's measures are treated as a time-series and moments are extracted. These moments make the final vector, representative of the entire text. This methodology is explored and extended with 2 variations. The first variation skips the time-series step, resulting in the various measures from each graph being used directly as samples. The second variation models the entire text as one graph. The methodologies are tested in corpus in both English and Portuguese, with varying number of texts.

Keywords: Authorship Attribution, Natural Language Processing, Graphs, Co-Occurrence Networks, Classification

1 Introduction

The task of Authorship Attribution can be described as the process of identifying the most likely author of a text of unknown authorship, given a collection of texts of known authorship.

The idea is to take advantage of the fact that different authors write in different ways, allowing one to "model" the way each author writes, thus being able to identify the author of a text of unknown authorship.

Authorship Attribution is not a new task. It has a long history, starting from the 19$^{\text{th}}$ century. One popular example is the controversy concerning authorship of some texts traditionally attributed to Shakespeare, in 1947.

This problem was originally resolved by experts, who looked at the text in a certain way - looking at verbs, the language, use of certain tenses more than others, etc. In 1964, there was a shift. The new techniques focused in other aspects that don't require experts. That is, the use of modern statistics and modern computers allows a more practical and sophisticated way to investigate the authorship question. There are now also more applications of authorship: source code, copyright disputes, emails, blogs and forum messages.

In the authorship field, there are other tasks that can potentially aid in the task of Authorship Attribution. Tasks such as:

- Author Recognition, which allows one to find the "real" author of some text that has been heavily distributed, helping, by association with the detection of plagiarism
- Author Verification, which can be described as: given some text of unknown authorship and a collection of books written by one author, decide if the text was written by this author
- Author profiling or characterization, which tries to infer information about the author like age, education and sex.
- Detection of stylistic inconsistencies, which looks for inconsistencies in the text as in the case of, for example, collaborative writing

Depending on the implementation, it's also possible to get author recommendations: Given a reader's favorite author, the recommended authors are those who write similarly.

Over the years many ways to model how an author writes have surfaced: one can analyze the frequency of words or expressions, the use of certain words, the punctuation, the vocabulary size, word length, sentence length, among others.

This task can be considered a classification problem. A classifier is trained with the modeled data from the training texts and the author labels. The predicted author is then predicted with the modeled data from the testing text.

This problem can be divided into two sub-fields: Closed set Authorship Attribution and Open set Authorship Attribution. Closed set Authorship Attribution is what was described until this point: all possible authors are known and are a possibility while in Open set Authorship Attribution there's an extra option: none of the known authors is the writer. In this thesis only Closed set Authorship Attribution will be addressed.

This thesis directly explores a previous work, building upon it. The previous worked uses a methodology that splits the text in multiple parts. A graph is then used to represent each part and various measures are extracted. The many parts and their respective extracted measures can be considered a time-series, from which moments are extracted, giving a final vector. As such, each text results in one sample to use.

This is in contrast with two variations also explored in this thesis. In the first variation the time-series step is skipped, resulting in each text being represented by as many samples as the number of parts. In the second variation the text is never split, there is only one graph and it represents the entire text.

The rest of this thesis is organized as follows: Chapter 2 discusses existing methodologies. The formal definition of the problem and the methodology used are defined in Chapter 3. The corpus used to experiment are defined in Chapter 4. This chapter also includes important details about each corpus, like the number of tokens, and the number of documents. In Chapter 5 all experiments and approaches are explained. This chapter also explains the evaluation procedure. The experimental results obtained are discussed in Chapter 6. A discussion of the work, its conclusions and future work are in Chapter 7.

2 State of The Art

For the task of Authorship Attribution there are many ways to model how an author writes or even to process that information.

Most of the existing work focuses in the English language [Sta17], [GAPDSP18], [Rho15], [GASP$^+$16], [Juo08], [ABR19], [PB18], [QHZ17], [Sei13], [AAO17], [Ama15], [PS17]. Most work also focuses only in Closed set Authorship Attribution [Sta09].

Several works have confirmed the difficulty that comes when the unknown texts are about a different topic than the initial texts. This problem has been studied in [SSV18], where the authors explore how various aspects of the text change with different topics and what features are more consistent. To resolve the topic issue, one can take the approach of modifying the text using rules, in hopes of making it more topic independent [Sta17].

One could divide existing approaches - of modeling how an author writes - in two categories.

In the first category of approaches, features are extracted using only the text. The most prevalent technique is *n*-grams, with different variations. The authors of [ABR19] use word-*n*-grams and construct a bag-of-words representation (which represents each text by a vector, with each dimension encoding the number of occurrences of each *n*-gram); in [IR19], word-*n*-grams, POS (part-of-speech refers to the grammatical category of the word) bigrams and word/POS pairs are taken into account; in [Sta17], word-*n*-grams are used directly; in [GAPDSP18], word-*n*-grams are used alongside POS; in [BBU07], word-*n*-grams are used to construct a bag-of-words; in [SSV18], both word-*n*-grams and character-*n*-grams are utilized; in [CMVPMR06], word-*n*-grams are used; in [Rho15] word-*n*-grams are utilized alongside Convolutional Neural Networks.

In summary, one could conclude that just *n*-grams seem to be insufficient. The strength of these approaches lies in the combination of multiple features. This is the approach taken by [PB18], where the authors calculate *n*-grams, POS *n*-grams, function words (words that are used to make sentences grammatically correct), among others. The difference from other works is that the final features are chosen using a consensus.

Still in the first category, one must not forget the recent boom in word embeddings. It has allowed various authors to experiment with this technology. The appearance of word embeddings allows the characterization of each word with a vector, encoding it's meaning and context; from this concept one can build sentence embeddings and document (whole text) embeddings, if one has a technique to merge embeddings.

In [QHZ17] the base unit is a word embedding, particularly from GloVe. To identify the author, various deep learning models are tested, at the sentence level and article level. At the sentence level, a Gated recurrent unit (GRU) is used. At the article, that is to say, whole text level, a GRU, a Long short-term memory (LSTM) and Siamese Network were used. The shortcomings, as stated in the paper, are the usage of averaged word embeddings; one could try other approaches such as concatenation. Another use of embeddings is by [GAPDSP18], where the authors train a doc2vec model. This model is capable of transforming the whole text, regardless of size, to a single vector (of fixed size).

In the second category of approaches the text is transformed into some other structure. That structure is then used for either classification or for extracting further features. There has been some study in this category by both [AAO17] [GASP$^+$16].

In [AAO17] the authors split the text in multiple partitions. Each partition is then represented in a graph, from which measures are extracted. In order to compare different sized texts, the various graph measures are treated as time-series, allowing for an extraction of 4 moments. Finally, these moments are then used as features.

In [GASP$^+$16] the authors encode various features of different types, such as lexical, morphological, syntactic and semantic into the graphs. The Authorship Attribution is done by comparing a graph (generated by the unknown text) to author graphs.

There has also been work done to reduce the dimensionality of the features. One can use techniques such as Latent Dirichlet allocation (LDA) or umap [MHM20]. The use of LDA, however, introduces a potential problem: it requires a choice of number of topics, which makes this approach corpora dependent [ABR19].

From the task of Authorship Verification, which can be adapted for Authorship Attribution, one approach involves having two collections of texts. The first collection is composed of ”imposter” texts, written by other authors, while the second contains texts written by the author that one is trying to verify authorship. Both collections are compared to the unknown text and, if the verified texts are more similar than the ”imposter” texts, one can, given some threshold, verify the author [PS17].

3 Authorship Attribution using Co-Occurrence Networks

The problem will now be formulated formally. For this problem, one needs a collection $A = [a_1, a_2, ..., a_n]$ of authors, a set of texts $T = [t_1, t_2..., t_k]$ written by the authors, and a mapping from a text $t_i \in C$ to it's author $a_n \in A$. With these, the goal is: given a random text t predict the most probable author $a_n \in A$ that wrote it.

The overall process can be described as follows:

1. Preprocess and tokenize the text t
2. Model the text t into graphs
3. Extract measures from the graphs to use as features
4. Transform / Remove some features

5. Use the features to train/predict from a classifier

For the task at hand - Authorship Attribution - the following 3 architectures are tested:

1. *By Time Series*: The architecture defined by [AAO17]; it serves as a baseline. Each text is divided in multiple partitions, each with the same number of tokens; each partition is characterized by 12 different measures, which can be represented as 12 time series. Finally, from each time series, 4 moments are used as features, resulting in 48 features. This architecture is fully explained in Section 5.1
2. *By Partition*: This is a variation of the *By Time Series* architecture. Each text is also divided in multiple partitions, each with the same number of tokens; each partition is characterized by 12 different measures. In this architecture, however, each partition is considered a sample, and its features are the 12 measure values. In other words, the time series step is skipped. This architecture is fully explained in Section 5.2.
3. *By Document*: This is another variation of the *By Time Series* architecture, but is also referenced in [AAO17]. The text is not split in partitions; instead, each text is considered as a "partition", and thus is represented as one graph. From this graph are then extracted the 12 measures. In this architecture, each text is considered a sample, and its features are the 12 measure values. This architecture is fully explained in Section 5.3.

3.1 Preprocess and tokenize the text

For preprocessing and tokenizing the texts, the spaCy [HMVLB20] library (version v3.0) is used. The models used are `en_core_web_md` and `pt_core_web_md`.

Since some texts are of length superior to spaCy's maximum, the tagger couldn't be used directly. To bypass this issue, the texts were first split in sentences.

Now each of the sentences is subject to tokenization, that is, splitting the text into words (or tokens). For example, `"You were told, no doubt."` becomes the sequence of (; separated) tokens: [`"` ; `You` ; `were told` ; `,` ; `no` ; `doubt` ; `.` ; `"`].

As suggested by [AAO17], using the lemma instead of the verbatim text gives more insight when dealing with word association. The lemma of a word can be described as the "base form". For example, the lemma of both `best` and `good` is the same: `good`. This is important because we want to be able to infer information regarding the relationship of words, not the words themselves. For the same reason, any punctuation can be discarded.

In languages like English or Portuguese, valid sentences have an innate structure that introduces words (in English) such as "the, is, for". These words, called *stop words*, don't help differentiate between authors since they are used by everyone and, as such, can be removed. Following the same example as above, and removing the stop words, the tokens now become: [`tell`, `doubt`].

3.2 Separate the text in partitions

In architectures other than *By Document*, the tokenized text is separated in partitions of some `X` size. Since the last partition is not guaranteed to have the same size as the rest, it is discarded. The value of

`X` is corpus dependent and is calculated through trial-and-error, trying to maintain a good ratio between too long partitions and too many. For example, if the tokenized text is `[good, time, bad, time, age, wisdom, age, foolishness]` and `X = 5`, there would be only one partition with the tokens `[good, time, bad, time, age]`.

3.3 Create co-occurrence graphs for each partition

The partitions are transformed into graphs, allowing several graph related measurements to be extracted. The graph construction is done by iterating the tokens in pairs and constructing the graph, as described in Algorithm 1.

Algorithm 1 Create co-occurrence graphs

```
Data: token pairs
Result: A directed weighted graph
$graph$ ← new graph
for $first, second$ in $pairs$ do
    $first\ node \leftarrow add\_node(graph, first)$
    $second\ node \leftarrow add\_node(graph, second)$
    if exists edge $first \rightarrow second$ then
        weight of $first \rightarrow second$ ← weight of $first \rightarrow second + 1$
    else
        $add\_edge(graph, first, second, 1)$
    end
    if exists edge $second \rightarrow first$ then
        weight of $second \rightarrow first$ ← weight of $second \rightarrow first + 1$;
    end
end
return $graph$
```

The result is a directed weighted graph, `D`. An example of this process can be seen in Figure 3.1, where one can see the graph generated by the text

```
The woods are lovely dark and deep
But I have promises to keep
And miles to go before I sleep ,
And miles to go before I sleep .
```

which is an extract from the poem *Stopping by the woods in the snowy evening*, by Robert Frost.

Since directed weighted graphs can be cyclical, and some of the measures used can't be computed for cyclical graphs, 2 other representations are introduced:

An Undirected version where each edge becomes undirected. The weight of the edge `u-v` becomes the max of the weights `u->v` and `v->u`. This graph is called `UD`.

An Undirected version where each edge becomes undirected but only keep nodes `u`, `v` if both edges `u->v` and `v->u` exist; The weights are calculated the same way as described above. This graph is called `UDR`.

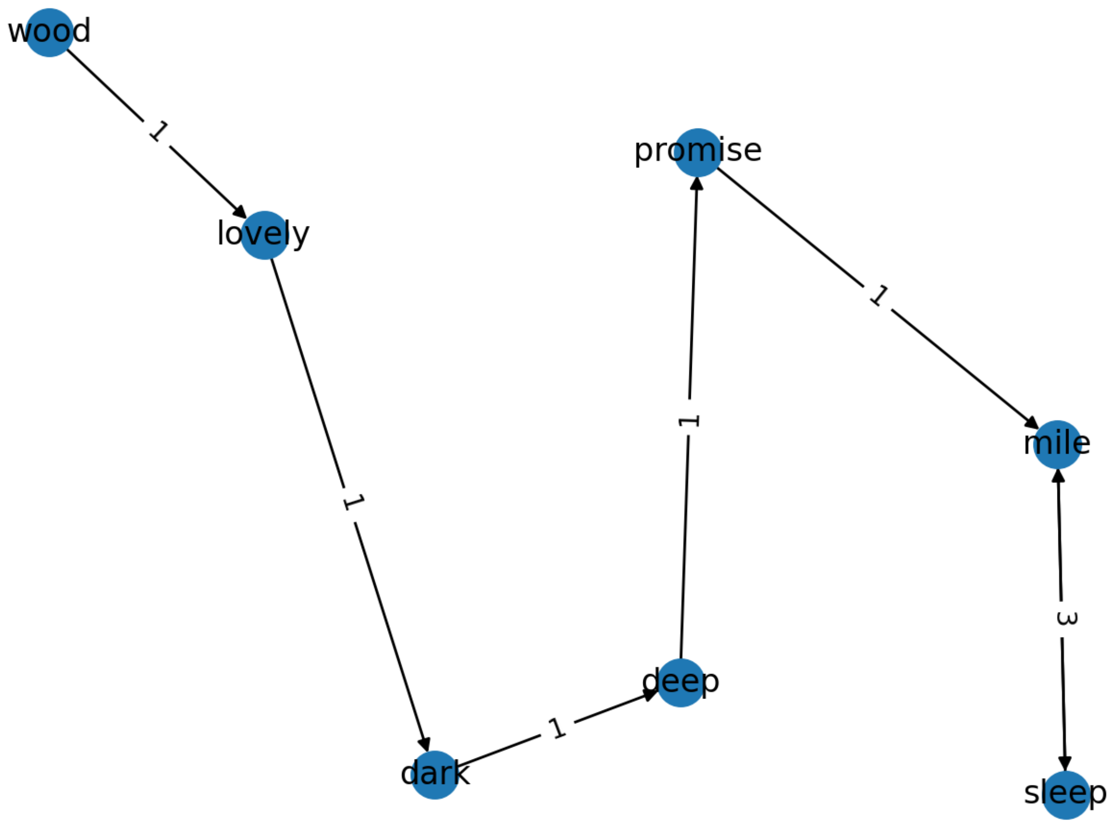

Figure 3.1: Example of a Co-Occurrence graph, in it's directed form.

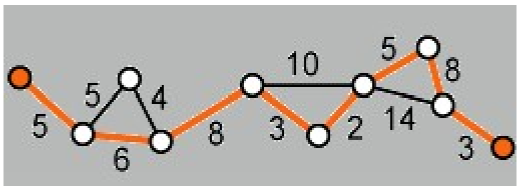

Figure 3.2: The diameter is displayed by the orange path. Adapted from `http://www.gitta.info/Accessibiliti/en/html/StructPropNetw_learningObject2.html`

3.4 Extract measures

For each partition, the graphs `D`, `UD` and `UDR` are used. To extract the measures, the networkx [HSS08] library (version 2.5) was used.

3.4.1 Average Clustering Coefficient

The Average Clustering Coefficient gives the average fraction of possible triangles for each node, for all nodes in graph `D`.

This corresponds to the function `networkx.algorithms.cluster.average_clustering`.

3.4.2 Diameter

The Diameter is the largest of all longest paths (eccentricity) between any two nodes in graph `UD`.

This corresponds to the function `networkx.algorithms.distance_measures.diameter`.

3.4.3 Radius

The Radius is the smallest of all longest paths (eccentricity) between any two nodes in graph `UD`.

This corresponds to the function `networkx.algorithms.distance_measures.radius`.

3.4.4 Cliques

The number of Cliques (complete sub-graphs) in graph `UDR`.

This corresponds to the function `networkx.algorithms.clique.graph_number_of_cliques`.

3.4.5 Average Load Centrality

The Average Load Centrality measures how many shortest paths, in average, pass through each node, considering weights, for all nodes in graph `UD`.

This corresponds to taking the average of the result from the function `networkx.algorithms.centrality.load_-centrality`.

3.4.6 Network Transitivity

The Network Transitivity measures the fraction of all connected triples that are triangles, in graph `D`.

This corresponds to the function `networkx.algorithms.cluster.transitivity`.

3.4.7 Average Betweenness Centrality

The Average Betweenness Centrality is the average of how many shortest paths pass through a node, for all nodes in graph `UD`.

This corresponds to taking the average of the result from the function `networkx.algorithms.centrality.betweenness_centrality`.

3.4.8 Average Shortest Path

The Average Shortest Path length is the average of the smallest number of edges between two nodes, for all nodes in graph `UD`.

This corresponds to the function `networkx.algorithms.shortest_paths.generic.average_short-est_path_length`.

3.4.9 Average Degree

The Average Degree (number of edges) of all nodes, in graph `UD`.

3.4.10 Number of Nodes

The total Number of Nodes, in graph `D`.

3.4.11 Number of Edges

The total Number of Edges, in graph `D`.

3.4.12 Average Intermittency

Intermittency expresses the periodicity of a word (to repeat). This measure is not calculated using a graph, but taking into account both, the partition and the text.

Start by calculating the set of spacings for all tokens in the text. A set of spacings represents how many tokens separate each consecutive occurrence of that token. To also take into account the token's first occurrence (t_0) and the number of tokens from the last occurrence to the last token of the text (t_f), a spacing $t_0 + t_f$ is added. For example in the text with no stop-words: `['graph', 'node', 'edge', 'edge', 'connect', 'node', 'node']`, the set of spacings for `node` is `{5-1=4, 6-5=1, 1+(6-6)=1}`; for `edge` it's `{3-2=1, 2+(6-3)=5}`.

Then, for each token `i` in the partition, calculate its intermittency using Equation 3.1

$$I = \sqrt{\frac{\overline{T}}{\overline{\overline{T^2}}} - 1} \tag{3.1}$$

where $\overline{T}$ denotes the average over all the spacings, and $\overline{T^2}$ denotes the average over all the spacings squared.

The intermittency is only considered for tokens that appear in the original text more than 4 times, as these are the more relevant ones [Ama15]. If no such tokens exist in the partition, the Average Intermittency is set to `0.0`.

3.5 Detailed Example

An example of the process is shown for the following extract, from the book "The Poems of Jonathan Swift, D.D., Volume 1", by Jonathan Swift.

```


Stella naturally expected to survive Swift , but it was not to be.She
died in the evening of the 28th January 1727—8; and on the same night he
began the affecting piece , "On the Death of Mrs. Johnson." (See "Prose
Works," vol.xi.)

With the death of Stella , Swift's real happiness ended , and he became
more and more possessed by the melancholy which too often accompanies the
broadest humour , and which , in his case , was constitutional.It was, no
doubt , to relieve it , that he resorted to the composition of the doggerel
verses , epigrams , riddles , and trifles exchanged betwixt himself and
Sheridan , which induced Orrery's remark that "Swift composing Riddles is
Titian painting draught—boards;" on which Delany observes that "a Riddle
may be as fine painting as any other in the world.It requires as strong
an imagination , as fine colouring , and as exact a proportion and keeping
as any other historical painting"; and he instances "Pethox the Great,"
and should also have alluded to the more learned example——"Louisa to
Strephon."

On Orrery's seventh Letter , Delany says that if some of the "coin is
base," it is the fine impression and polish which adds value to it , and
cites the saying of another nobleman , that "there is indeed some stuff
in it , but it is Swift's stuff."It has been said that Swift has never
taken a thought from any writer ancient or modern.This is not literally
true , but the instances are not many, and in my notes I have pointed out
the lines snatched from Milton , Denham, Butler——the last evidently a
great favourite .

It seems necessary to state shortly the causes of Swift not having
obtained higher preferment.Besides that Queen Anne would never be
```

```
reconciled to the author of the "Tale of a Tub"——the true purport of
which was so ill—understood by her——he made an irreconcilable enemy of
her friend , the Duchess of Somerset , by his lampoon entitled "The Windsor
Prophecy ."But Swift seldom allowed prudence to restrain his wit and
humour, and admits of himself that he "had too much satire in his vein";
and that "a genius in the reverend gown must ever keep its owner down";
and says further :

Humour and mirth had place in all he writ ;
He reconciled divinity and wit .

But that was what his enemies could not do.

Whatever the excellences and defects of the poems, Swift has erected , not
only by his works , but by his benevolence and his charities , a
_monumentum aere perennius ,_ and his writings in prose and verse
will continue to afford instruction and delight when the malevolence of
Jeffrey , the misrepresentations of Macaulay , and the sneers and false
statements of Thackeray shall have been forgotten .
```

After the first step - tokenization, stop-word removal and lemmatization - the text is significantly changed. The resulting text can be seen below:

```
Stella naturally expect survive Swift die evening 28th January 1727 8
   night begin affect piece Death Mrs. Johnson Prose Works vol.xi death
   Stella Swift real happiness end possess melancholy accompany broad
   humour case constitutional doubt relieve resort composition doggerel
   verse epigram riddle trifle exchange betwixt Sheridan induce Orrery
   remark swift composing Riddles titian paint draught board Delany
   observe Riddle fine painting world require strong imagination fine
   colouring exact proportion keep historical painting instance Pethox
   great allude learn example——"Louisa Strephon Orrery seventh Letter
   Delany say coin base fine impression polish add value cite saying
   nobleman stuff Swift stuff say Swift take thought writer ancient
   modern literally true instance note point line snatch Milton Denham
   Butler evidently great favourite necessary state shortly cause Swift
   have obtain high preferment Queen Anne reconcile author Tale tub"——the
    true purport ill understand irreconcilable enemy friend Duchess
   Somerset lampoon entitle Windsor Prophecy Swift seldom allow prudence
   restrain wit humour admit satire vein genius reverend gown owner say
   Humour mirth place writ reconcile divinity wit enemy excellence defect
    poem Swift erect work benevolence charity monumentum aere perennius
   writing prose verse continue afford instruction delight malevolence
   Jeffrey misrepresentation Macaulay sneer false statement Thackeray
   shall forget
```

For the next step - splitting the text - a size of 30 for each partition is employed. The resulting first partition can be observed below:

```
Stella naturally expect survive Swift die evening 28th January 1727 8
```

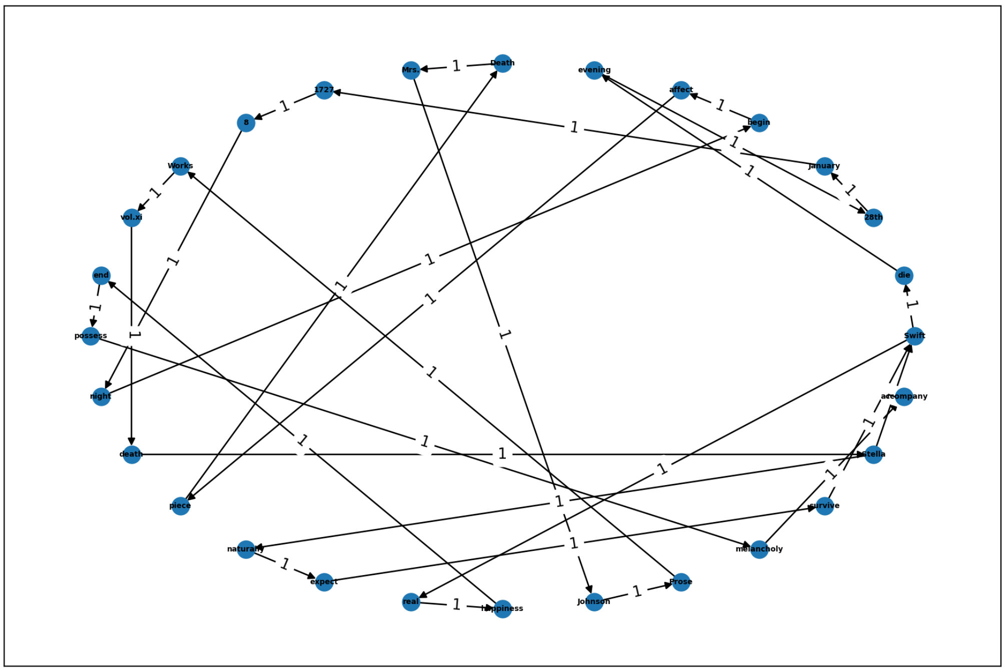

Figure 3.3: A Co-Occurrence graph created from the first partition.

```
night begin affect piece Death Mrs. Johnson Prose Works vol.xi death
Stella Swift real happiness end possess melancholy accompany
```

The result from the next step - creating the graph - can be observed in Figure 3.3. One can note that most edges have weight of 1, due to the small size of the extract. The results from the next step - extracting the measures - can be observed in Table 3.1. Once again, due to the small size of the extract, some values are 0.

Measure	Value
clustering coefficient	0
diameter	15
radius	9
cliques	0
load centralities	0.1917989417989418
transitivity	0
betweenness centrality	0.2495275888133031
shortest path	5.9867724867724865
degree	2.0714285714285716
nodes number	28
edges number	29
intermittency	6.315909526327415

Table 3.1: The various measures extracted from first partition's graph.

4 Corpora

The experiments are performed in three corpus. There are 2 corpus in English and 1 in Portuguese. All corpus have texts with an average number of tokens above 30000, as the approaches proposed in this thesis require a large enough text.

The first corpus - henceforth named `Gutenberg` - is a collection of books in English. The detailed information about each book is available in Appendix A.1. It has 80 books, each of its 8 authors with 10 books. This corpus was collected from Project Gutenberg, and, as such, has some metadata. This metadata is not relevant for Authorship Attribution, since it is not written by the author. To select only relevant text from each book, only the text limited by "*** START OF THIS PROJECT GUTENBERG EBOOK ***" and "*** END OF THIS PROJECT GUTENBERG EBOOK ***" was kept. While this removes most of the metadata, some transcriber notes are still present. The corpus is available at `https://data.mendeley.com/datasets/v964pnd26t/1`. In average, each book has 46941 tokens.

The next corpus - henceforth named `Gutenberg 6` - is a variation of the first corpus, `Gutenberg` where the books from Arthur Conan Doyle, Fyodor Dostoyevsky, Jack London, Jonathan Swift, Leo Tolstoy or Nathaniel Hawthorne are kept. The detailed information about each book is available in Appendix A.2. It has 60 books, each of its 6 authors with 10 books. In average, each book has 50109 tokens.

The last corpus - henceforth named `Portuguese` - is a collection of books in Portuguese. The detailed information about each book is available in Appendix A.3. It has 9 books, each of its 3 authors with 3 books. This corpus was collected from Project Gutenberg, but its metadata has been removed. The corpus is available in Mendeley at `https://data.mendeley.com/datasets/3ndyczks35/1`. In average, each book has 33706 tokens.

5 Authorship Attribution Co-Occurrence Network Architectures

One can construct multiple architectures using different features, classifiers, preprocessors, dimension reduction techniques and feature selection techniques. In this paper, the 3 main solutions tested are: *By Time-Series*, described in Section 5.1, *By Partition*, described in Section 5.2 and *By Document*, described in Section 5.3.

5.1 By Time-Series

In this architecture, each text is separated in various partitions, which are represented as graphs, and from which the measures are extracted. As such, each measure can be represented as a time series, by looking at all partitions. The 12 different measures are represented as 12 time series. From each time series, 4 moments are extracted - using Equation 5.1 - to be used as features with the classification algorithms. The training and prediction processes can be seen in Figure 5.1. On the left, one can see the steps taken when training with texts of known authorship; on the right, the steps taken when predicting the author of a text of unknown authorship.

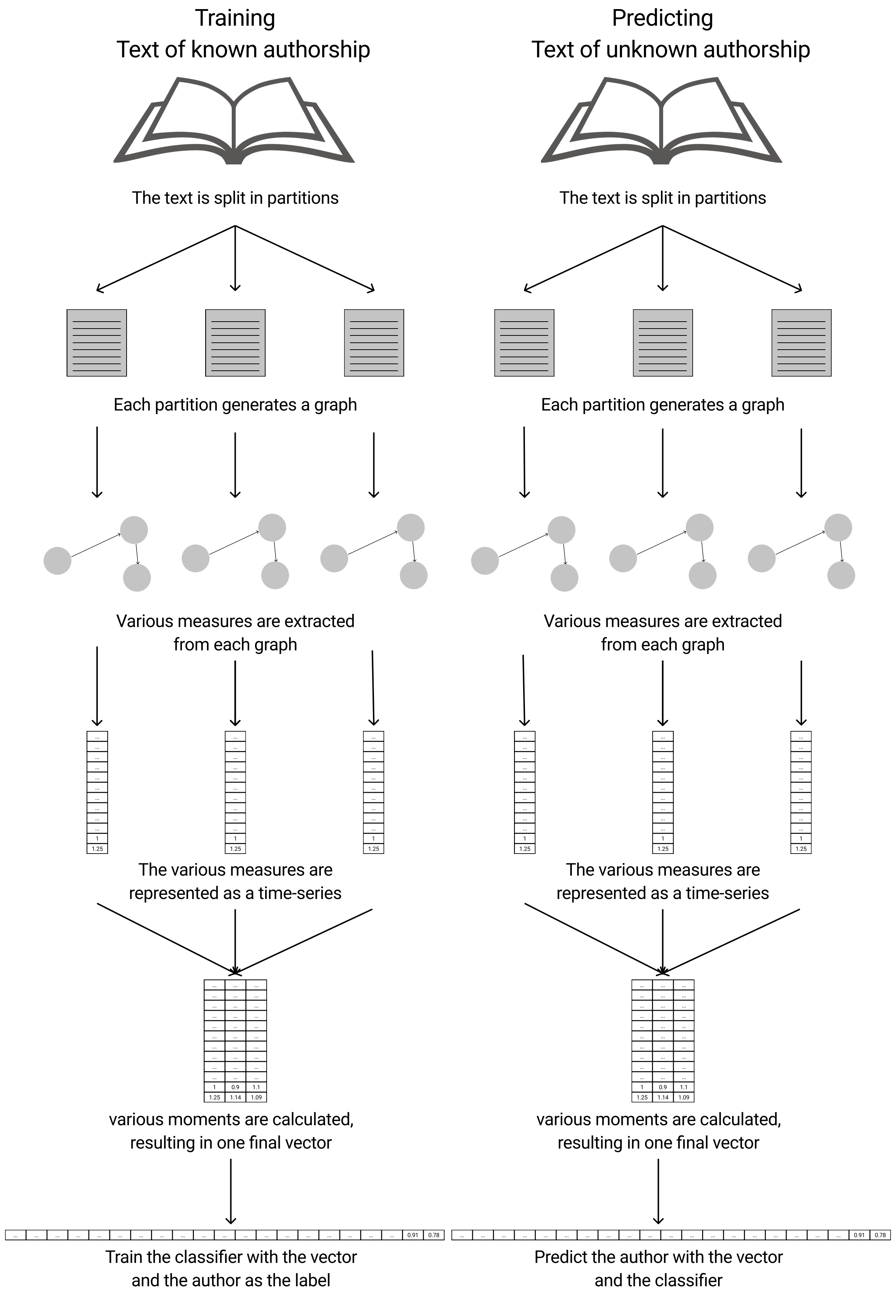
Training
Text of known authorship
Predicting
Text of unknown authorship
The text is split in partitions
The text is split in partitions
Each partition generates a graph
Each partition generates a graph
Various measures are extracted from each graph
Various measures are extracted from each graph
The various measures are represented as a time-series
The various measures are represented as a time-series
various moments are calculated, resulting in one final vector
various moments are calculated, resulting in one final vector
Train the classifier with the vector and the author as the label
Predict the author with the vector and the classifier

Figure 5.1: The By Time-Series architecture.

5.1.1 Features

Given that the time series can be considered stationary - as described in [AAO17] - and thus "allows one to compare estimated values for sample statistics from series of different lengths", the features for each measure can be calculated using Equation 5.1.

$$u_i = \sqrt[i]{\frac{1}{S-1}\sum_{j=0}^{S}(x_j - u_1)^i} \tag{5.1}$$

where u_1 denotes the average of the values in the series, S denotes the size of the series, x_j denotes the jth value of the series and $i \in [2, 3, 4]$.

This process yields 4 features for each measure, u_1, u_2, u_3 and u_4. Since there are 12 measures, each text is characterized by a 48 features vector.

5.2 By Partition

To train a classifier, one can look at each text's partitions and consider each partition a sample. Since each partition has 12 measures, these can be used as features and the author as the label.

To get the inferred author of a text, for each partition, its author is predicted. This yields as many author predictions for the text as the number of partitions. The final predicted author is considered to be the author that is the majority of the predictions. The training and prediction processes can be seen in Figure 5.2. On the left, one can see the steps taken when training with texts of known authorship; on the right, the steps taken when predicting the author of a text of unknown authorship.

5.3 By Document

Finally, one can look at the entire text as just one partition, being characterized by that partitions' *clustering coefficient, diameter, radius, cliques, transitivity, shortest path, degree, nodes number, edges number and intermittency* values. The *load centralities* and *betweenness centrality* values are not computed as they are very intensive to compute for larger documents. The training and prediction processes can be seen in Figure 5.3. On the left, one can see the steps taken when training with texts of known authorship; on the right, the steps taken when predicting the author of a text of unknown authorship.

5.4 Tools

In this thesis, components from sklearn [PVG+11] will be used. These can be divided in 4 categories: Preprocessing, Dimensionality Reduction, Feature Selection and Classifiers.

The components from the Preprocessing category are designed to treat the incoming feature values, changing them to fit the problem better; this could be standardizing the distribution, scaling to a range, among others. In this thesis, the following components are considered:

- Robust Scaler - scales the values but has special care regarding outliers.

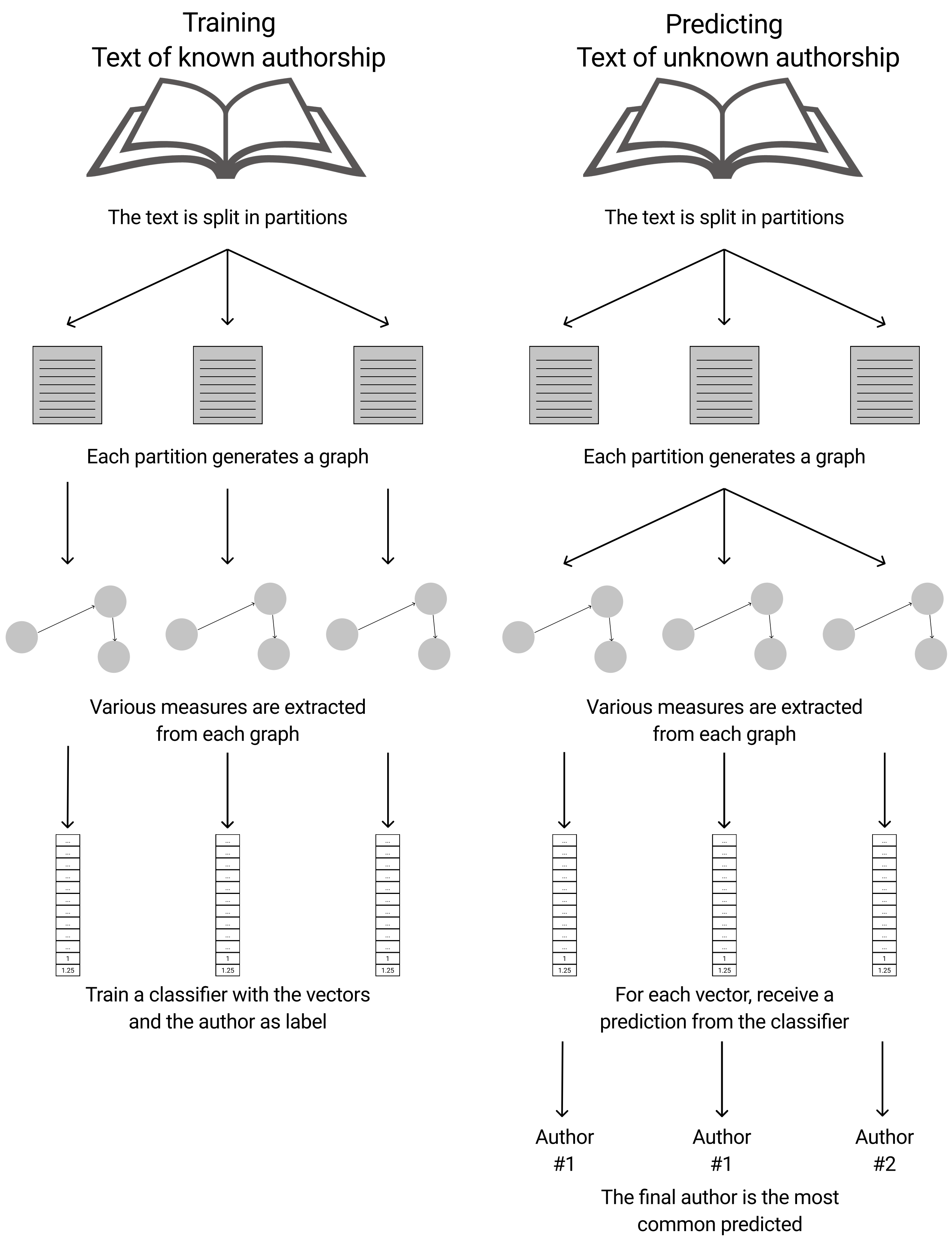
Training
Text of known authorship
Predicting
Text of unknown authorship
The text is split in partitions
The text is split in partitions
Each partition generates a graph
Each partition generates a graph
Various measures are extracted from each graph
Various measures are extracted from each graph
...
1
1.25
Train a classifier with the vectors and the author as label
For each vector, receive a prediction from the classifier
Author #1
Author #1
Author #2
The final author is the most common predicted

Figure 5.2: The by partition architecture.

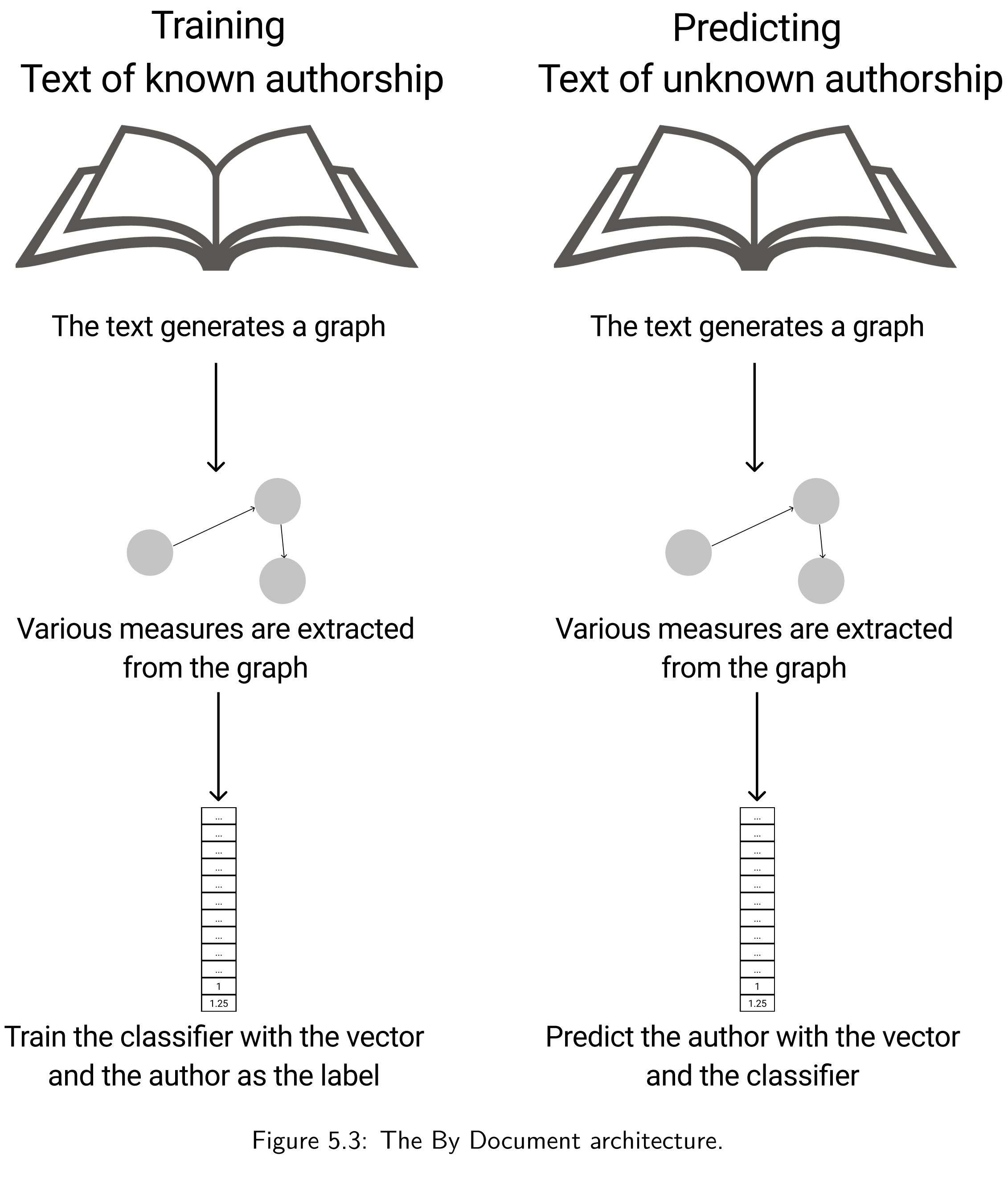
Training
Text of known authorship
Predicting
Text of unknown authorship
The text generates a graph
The text generates a graph
Various measures are extracted from the graph
Various measures are extracted from the graph
...
1
1.25
...
1
1.25
Train the classifier with the vector and the author as the label
Predict the author with the vector and the classifier

Figure 5.3: The By Document architecture.

- Quantile Transformer - transform the features so they follow a uniform distribution.
- Min Max Scaler - attempt to normalize the data within a given range.
- Max Abs Scaler - attempt to normalize the data within a given range.

The Dimensionality Reduction category has components that aim to reduce the dimension of the instances. Using components from this category in the *By Time-Series* architecture, one can reduce from vectors of 48 dimensions to 2. In this thesis, the following components are considered:

- Isomap - "seeks a lower-dimensional embedding which maintains geodesic distances between all points" [PVG+11]

From the Feature Selection category, no component is present in sklearn - at this time - that satisfies the necessities. Of all 12 features, some might be more appropriate than others; it follows then, that it's possible that using only a subset of all 12 features may lead to better results. This step will only be done in *By Partition* architecture where an exhaustive search of all subsets of features is doable. This is because the search space is more forgiving, since the samples only contain 12 features - resulting in $2^{12} - 1 = 4095$ possible combinations - as opposed to the 48 in the case of *By Time-Series*. This component will henceforth be referred to as *Select Best*.

Regarding the Classifiers category, the goal of its components is to learn from the samples and labels so it's later possible to predict what label a new sample belongs to. In this thesis, the following components are considered:

1. Decision Tree - is a non-parametric supervised learning method that learns some rules from the data, while modeling it as a tree.
2. Gaussian Naive Bayes - works by training probability functions for the data.
3. `K` Nearest Neighbours - places the samples in space and predicts what label has the most number of neighbours (from `K`) in the location of the new sample.

5.5 Exclude

It was observed that in the Gutenberg dataset some features in each partition are very different from the mean of that feature. A possible reason for this is the existence of "metadata" in some partitions (footnotes, chapter information...). To address this issue, Algorithm 2, to transform the partitions, is proposed. It reorders the time series values, while excluding `N` values. The algorithm can be visualized in Figure 5.4 where, from the initial 3 partitions with 4 measures each (degree, radius, cliques and load centralities), 2 new partitions are calculated. In the newly calculated time-series, only the values closest to the mean of each measure are kept. The highlighted values are discarded, as they are the farthest from their respective means. This component is applied before any others, changing the dataset.

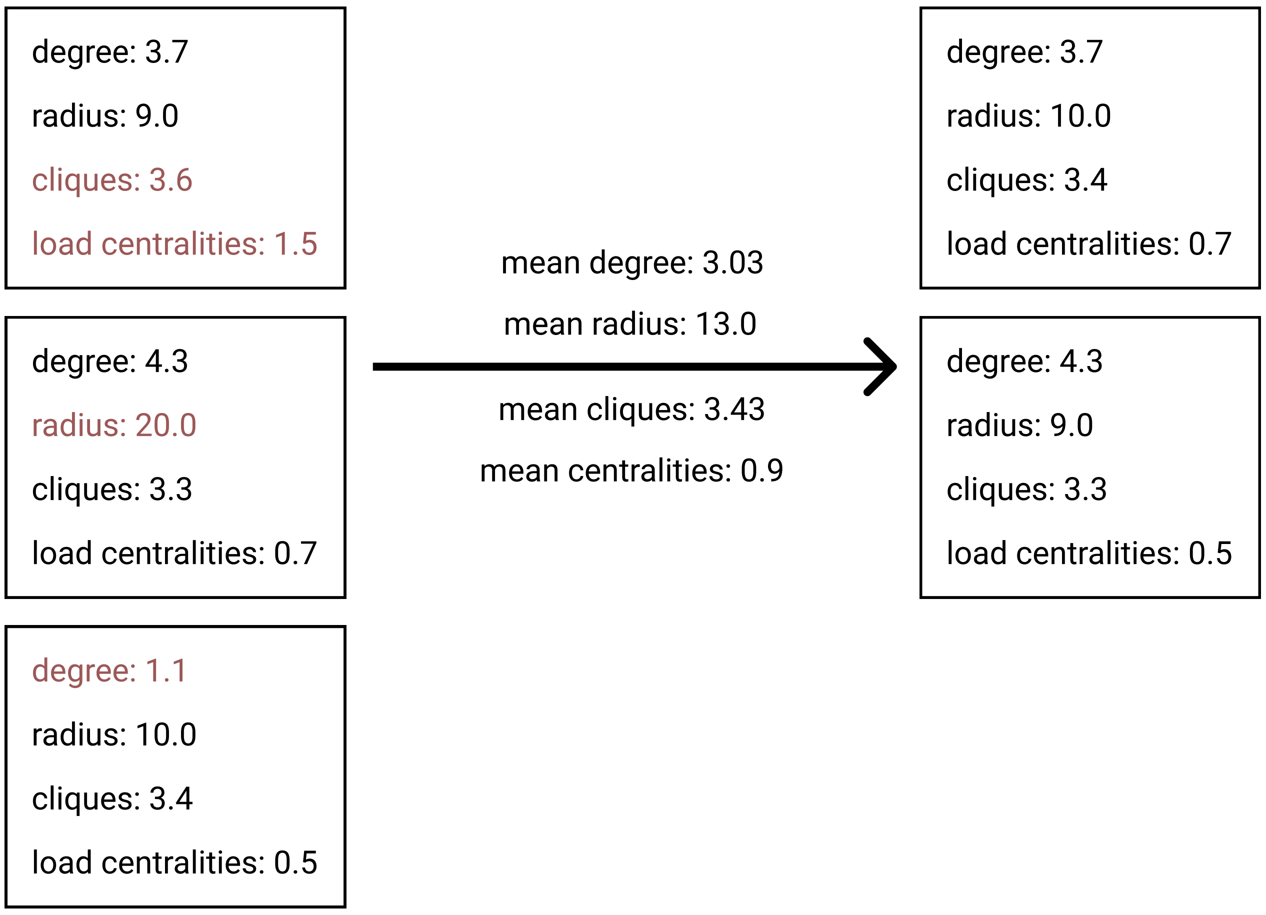

Figure 5.4: An example of the Exclude algorithm with exclude = 1

Algorithm 2 Exclude Algorithm

```
Data: old partitions, N, measures
Result: the new partitions
$new\ partitions \leftarrow partition[N]$
for $measure$ in $measures$ do
    $avg \leftarrow$ average of $measure$ in the series
    $diffs \leftarrow [\ ]$
    for $i$ =0 to $|partitions|$ do
        $diffs[i] \leftarrow (avg - partitions[i][measure])^2$
    end
    $argdiffs \leftarrow argsort(diffs)$
    $i \leftarrow 0$
    for $index$ in $argdiffs$ do
        $new\ partitions[i][measure] \leftarrow partitions[index][measure]$
        $i \leftarrow i + 1$
        if $i = N$ then
            break
        end
    end
end
return $new\ partitions$
```

5.6 Evaluation Measures

To evaluate the various architectures, one can consider the Authorship Attribution problem as a classification problem where each author is a class. The performance of such problems can be evaluated using the two standard scores: precision and recall. For a given class, precision and recall are defined as:

$$Precision = \frac{\#\text{True Positives}}{\#\text{True Positives} + \#\text{True Negatives}}$$

$$Recall = \frac{\#\text{True Positives}}{\#\text{True Positives} + \#\text{False Negatives}}$$

To get a general idea of the precision or recall of all classes, one can use micro or macro-averaging. With micro-averaging (accuracy) the score of each class is weighted by the number of instances, while the macro-average score is the mean of the scores of all classes (same weight).

To compare the results with [AAO17], the same technique was used to calculate the results: Stratified K-Fold. This allows one to use the same dataset to test and train. The dataset is split in folds, some being used for testing and others for training; the idea is to repeat this process multiple times, iterating over the dataset, yielding different folds each time. This can be observed in Figure 5.5 with imbalanced classes and 3 iterations.

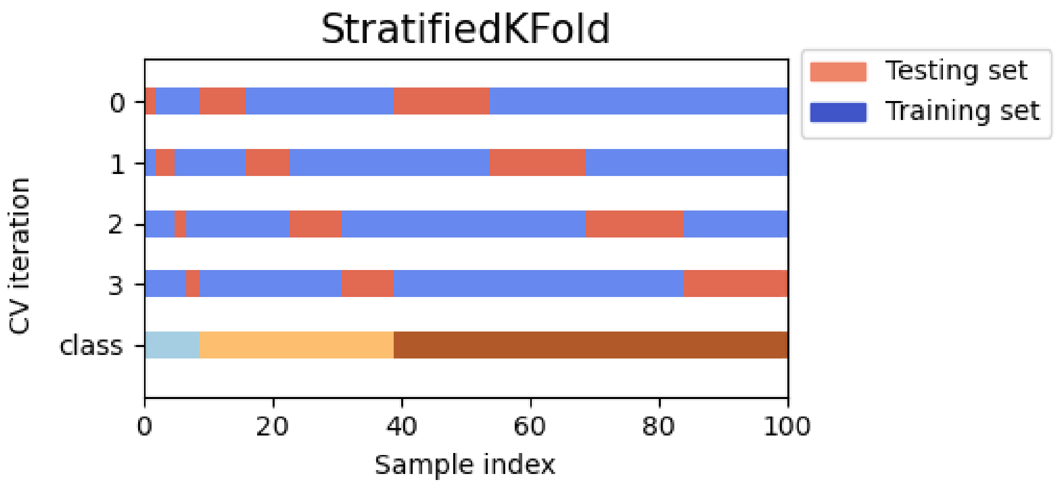

Figure 5.5: An example of Stratified k fold with 3 iterations and imbalanced classes. Adapted from `https://scikit-learn.org/stable/modules/cross_validation.html`

6 Experiments

As discussed in the previous chapter, one can construct multiple architectures by changing the components used. For this thesis, as described in Chapter 5, multiple architectures are tested: *By Time Series* - described in Section 5.1, *By Partition* - described in Section 5.2 and *By Document* - described in Section 5.3. The various components used for the various categories (Preprocessing, Dimensionality Reduction, Feature Selection, Classifiers) are described in Section 5.4. In the Preprocessing category, the components tested are Robust Scaler, Quantile Transformer, Min Max Scaler and Max Abs Scaler. In the Dimensionality Reduction category, only one component is tested: Isomap. For the Feature Selection category, the component Select Best - which selects the best feature set - is used in the By Partition architecture. For the Classifier category, the components tested are Decision Tree, Gaussian Naive Bayes and K Nearest Neighbours.

In addition, the proposed Exclude algorithm (described in Section 5.5) - used to remove and rearrange samples - is used in the *By Time Series* and *By Partition* architectures. In the former, this algorithm is used before calculating the moments.

For the three corpus: Gutenberg, Gutenberg 6 and Portuguese - presented in detail in Chapter 4 - the same experiments are performed. Most parameters are the default except any random state (when available) set to 42. The parameters whose values are not default - and are corpus dependent - can be seen in Table 6.1.

Parameter	Gutenberg	Gutenberg 6	Portuguese
Stratified	10-fold	10-fold	3-fold
Number of neighbours in K Nearest Neighbours	2	2	2
Number of neighbours in Isomap	10	10	5
Number of components in Isomap	10	10	10

Table 6.1: Unique parameters for each corpus. Both Gutenberg and Gutenberg 6 use the same values, while Portuguese uses different values in some components, since it has less texts.

Each architecture is tested with a different set of components, as one can see next.

6.1 By Time Series results

The first set of experiments utilizes the *By Time Series* architecture without `Exclude`, and uses components in a similar fashion to [AAO17]: a Preprocessor, Isomap and a Classifier. In addition, to measure the impact of using a Preprocessor and Isomap, tests are performed without these components. Next, the same experiment is performed but with `Exclude`, values ranging from 1 to 9. In the results only the best scoring `Excluded` are shown, as otherwise there are too many results to show. The tested configurations can be seen below:

- Exclude $\rightarrow$ Preprocessor $\rightarrow$ Isomap $\rightarrow$ Classifier
- Exclude $\rightarrow$ Preprocessor $\rightarrow$ Classifier
- Exclude $\rightarrow$ Classifier
- Preprocessor $\rightarrow$ Isomap $\rightarrow$ Classifier
- Preprocessor $\rightarrow$ Classifier
- Classifier

6.1.1 Gutenberg

Without Exclude

The best accuracy of 60% is achieved using a Gaussian Naive Bayes as the Classifier with Quantile Transformer as the Preprocessor but without the use of Isomap.

The best accuracy the K Nearest Neighbours Classifier achieves is 50% with a Min Max Scaler as the Preprocessor but without the use of Isomap.

The best accuracy the Decision Tree Classifier achieves is 49% with multiple configurations. The Preprocessor step seems to have no effect in this case. It is clear, however, that using Isomap hurts the accuracy.

The rest of the results can be seen in Table 6.2.

Steps	Decision Tree(%)	Gaussian Naive Bayes(%)	K Nearest Neighbors(%)
-	49.00	51.00	40.00
Max Abs Scaler	49.00	55.00	44.00
Max Abs Scaler, Isomap	31.00	42.00	33.00
Min Max Scaler	49.00	55.00	50.00
Min Max Scaler, Isomap	31.00	42.00	38.00
Quantile Transformer	49.00	**60.00**	41.00
Quantile Transformer, Isomap	38.00	42.00	38.00
Robust Scaler	49.00	55.00	44.00
Robust Scaler, Isomap	31.00	45.00	40.00

Table 6.2: Results of By Time Series, in Gutenberg, without the use of Excluded.

With Exclude

Even the best `Excluded` values don't usually have much effect; there's an average change in the order of the 1-5%, depending on the steps and Classifier.

The best scoring Classifier is still Gaussian Naive Bayes with 61%, when `Excluded = 4`. This is achieved with Quantile Transformer as a Preprocessor but without the use of Isomap.

The best accuracy the K Nearest Neighbours Classifier achieves is 51%, when `Excluded = 3` with a Min Max Scaler as the Preprocessor but without the use of Isomap.

The best accuracy the Decision Tree Classifier achieves is 51%, when `Excluded = 5` with Quantile Transformer as a Preprocessor but without the use of Isomap.

The rest of the results can be seen in Table 6.3.

6.1.2 Gutenberg 6

Without Exclude

In this corpus, the best accuracy - 70% - is achieved with Decision Tree as the Classifier. This is achieved with multiple configurations, regardless of the Preprocessor used, or even with none; in all cases, however, using Isomap results in worse accuracy.

The best accuracy the K Nearest Neighbours Classifier achieves is 55% with neither Preprocessor nor Isomap.

The best accuracy the Gaussian Naive Bayes achieves is 63% with Quantile Transformer as a Preprocessor but without the use of Isomap.

The rest of the results can be seen in Table 6.4.

With Exclude

The best `Excluded` values usually have a small change, depending on the steps and Classifier.

The best scoring Classifier is still Decision Tree with 72%, with `Excluded = 9`. This is achieved using

Steps	Decision Tree(%) Excluded	Gaussian Naive Bayes(%) Excluded	K Nearest Neighbors(%) Excluded
-	50.00 5	53.00 3	41.00 6
Max Abs Scaler	50.00 5	55.00 3	45.00 6
Max Abs Scaler, Isomap	40.00 9	46.00 8	38.00 5
Min Max Scaler	50.00 5	55.00 3	51.00 3
Min Max Scaler, Isomap	38.00 3	44.00 1	40.00 2
Quantile Transformer	51.00 5	**61.00** 4	42.00 4
Quantile Transformer, Isomap	45.00 8	51.00 2	41.00 9
Robust Scaler	50.00 5	55.00 3	45.00 7
Robust Scaler, Isomap	35.00 2	46.00 2	41.00 5

Table 6.3: Results of by Time Series in Gutenberg, using Exclude with values between 1-9.

Steps	Decision Tree(%)	Gaussian Naive Bayes(%)	K Nearest Neighbors(%)
-	**70.00**	57.00	55.00
Max Abs Scaler	**70.00**	57.00	52.00
Max Abs Scaler, Isomap	48.00	43.00	33.00
Min Max Scaler	**70.00**	57.00	53.00
Min Max Scaler, Isomap	37.00	53.00	45.00
Quantile Transformer	65.00	63.00	48.00
Quantile Transformer, Isomap	47.00	47.00	40.00
Robust Scaler	**70.00**	57.00	48.00
Robust Scaler, Isomap	47.00	53.00	47.00

Table 6.4: Results of By Time Series, in Gutenberg 6, without the use of Excluded.

Steps	Decision Tree(%) Excluded	Gaussian Naive Bayes(%) Excluded	K Nearest Neighbors(%) Excluded
-	**72.00** 9	57.00 1	55.00 1
Max Abs Scaler	**72.00** 9	57.00 1	52.00 1
Max Abs Scaler, Isomap	50.00 1	53.00 8	37.00 8
Min Max Scaler	**72.00** 9	57.00 1	57.00 4
Min Max Scaler, Isomap	45.00 2	63.00 5	47.00 6
Quantile Transformer	67.00 9	63.00 1	52.00 3
Quantile Transformer, Isomap	53.00 6	48.00 8	43.00 2
Robust Scaler	**72.00** 9	57.00 1	47.00 1
Robust Scaler, Isomap	48.00 2	53.00 8	50.00 3

Table 6.5: Results of by Time Series in Gutenberg 6, using Exclude with values between 1-9.

either a Robust Scaler, Min Max Scaler, Max Abs Scaler or no Preprocessor; in all cases, without the use of Isomap.

The best accuracy the K Nearest Neighbours Classifier achieves is 57%, with `Excluded = 4`. This is achieved using a Min Max Scaler as the Preprocessor but without the use of Isomap.

The best accuracy the Gaussian Naive Bayes Classifier achieves is 63%, with `Excluded = 5 or Excluded = 1`. This is achieved using two configurations. The first: a Quantile Transformer as a Preprocessor but without the use of Isomap. The second: Min Max Scaler as Preprocessor and Isomap.

The rest of the results can be seen in Table 6.5.

6.1.3 Portuguese

Without Exclude

The best accuracy of 89% is achieved using Gaussian Naive Bayes as the Classifier with Quantile Transformer as the Preprocessor but without the use of Isomap.

The best accuracy the K Nearest Neighbours Classifier achieves is 78%, with multiple configurations. In this case, the choice of Preprocessor is important: using Max Abs Scaler or Min Max Scaler yields the best results, regardless of usage of Isomap or not. Another configuration which also yields the 78% result is no Preprocessor but without the use of Isomap.

The best accuracy the Decision Tree Classifier achieves is 78%, with Min Max Scaler as the Preprocessor and Isomap.

Steps	Decision Tree(%)	Gaussian Naive Bayes(%)	K Nearest Neighbors(%)
-	67.00	78.00	78.00
Max Abs Scaler	67.00	56.00	78.00
Max Abs Scaler, Isomap	67.00	33.00	78.00
Min Max Scaler	67.00	56.00	78.00
Min Max Scaler, Isomap	78.00	33.00	78.00
Quantile Transformer	67.00	**89.00**	67.00
Quantile Transformer, Isomap	67.00	33.00	67.00
Robust Scaler	67.00	56.00	67.00
Robust Scaler, Isomap	67.00	33.00	67.00

Table 6.6: Results of By Time Series, in Portuguese, without the use of Excluded.

The rest of the results can be seen in Table 6.6.

With Exclude

Even the best `Excluded` values don't have any effect on the previous best scores.

The best accuracy of 89% is still achieved using Gaussian Naive Bayes as the Classifier with Quantile Transformer as the Preprocessor but without the use of Isomap.

The best accuracy the K Nearest Neighbours Classifier achieves is still 78%, with multiple configurations. In this case, the choice of Preprocessor is important: using Max Abs Scaler or Min Max Scaler yields the best results, regardless of usage of Isomap or not. Another configuration which also yields the 78% result is no Preprocessor but without the use of Isomap.

The best accuracy the Decision Tree Classifier still is 78% with Min Max Scaler as the Preprocessor and Isomap.

The rest of the results can be seen in Table 6.7.

6.2 By Partition results

The first set of experiments utilizes the *By Partition* architecture without `Exclude`. The components used are: a Preprocessor, Select Best and a Classifier. In addition, to measure the impact of using a Preprocessor and the Select Best Component, tests are performed without these components. Next, the same experiment is performed but with `Excluded = 1`. The tested configurations can be seen below:

- Exclude $\rightarrow$ Preprocessor $\rightarrow$ Select Best $\rightarrow$ Classifier
- Exclude $\rightarrow$ Preprocessor $\rightarrow$ Classifier
- Exclude $\rightarrow$ Classifier
- Preprocessor $\rightarrow$ Select Best $\rightarrow$ Classifier
- Preprocessor $\rightarrow$ Classifier
- Classifier

Steps	Decision Tree(%) Excluded	Gaussian Naive Bayes(%) Excluded	K Nearest Neighbors(%) Excluded
-	67.00 1	78.00 1	78.00 1
Max Abs Scaler	67.00 1	67.00 2	78.00 1
Max Abs Scaler, Isomap	67.00 1	44.00 2	78.00 1
Min Max Scaler	67.00 1	67.00 2	78.00 1
Min Max Scaler, Isomap	78.00 1	33.00 1	78.00 1
Quantile Transformer	67.00 1	**89.00** 1	67.00 1
Quantile Transformer, Isomap	67.00 1	33.00 1	67.00 1
Robust Scaler	67.00 1	67.00 2	67.00 1
Robust Scaler, Isomap	67.00 1	33.00 1	67.00 1

Table 6.7: Results of by Time Series in Portuguese, using Exclude with values between 1-9.

6.2.1 Gutenberg

Without Exclude

The best accuracy of 61% is achieved using a Decision Tree as the Classifier, with Robust Scaler as the Preprocessor and the Select Best component.

The best accuracy the K Nearest Neighbours Classifier achieves is 34% with many Preprocessors (Robust Scaler, Quantile Transformer or Max Abs Scaler), and the Select Best component.

The best accuracy the Gaussian Naive Bayes Classifier achieves is 45% with many Preprocessors (Robust Scaler, Max Abs Scaler or Min Max Scaler), and the Select Best component.

The rest of the results can be seen in Table 6.8.

With Exclude

The use of `Excluded = 1` has a positive effect, especially with K Nearest Neighbours.

The best scoring Classifier is still Decision Tree Classifier with 69%. This is achieved with Quantile Transformer as a Preprocessor and the Select Best component.

The best accuracy the K Nearest Neighbours Classifier achieves is 64% with Quantile Transformer as a Preprocessor and the Select Best component.

The best accuracy the Gaussian Naive Bayes Classifier achieves is 53% with Quantile Transformer as a Preprocessor and the Select Best Component.

Steps	Decision Tree(%)	Gaussian Naive Bayes(%)	K Nearest Neighbors(%)
-	48.00	39.00	26.00
Max Abs Scaler	50.00	39.00	25.00
Max Abs Scaler,Select Best	60.00	45.00	34.00
Min Max Scaler	49.00	39.00	26.00
Min Max Scaler,Select Best	59.00	45.00	33.00
Quantile Transformer	49.00	30.00	25.00
Quantile Transformer,Select Best	60.00	44.00	34.00
Robust Scaler	49.00	39.00	30.00
Robust Scaler,Select Best	**61.00**	45.00	34.00

Table 6.8: Results of by Partition, in Gutenberg, without the use of Excluded.

The rest of the results can be seen in Table 6.9.

6.2.2 Gutenberg 6

Without Exclude

The best accuracy of 65% is achieved using Decision Tree as the Classifier. This is achieved using either a Quantile Transformer or Robust Scaler as the Preprocessor and the Select Best component.

The best accuracy the K Nearest Neighbours Classifier achieves is 48% with a Max Abs Scaler as the Preprocessor, and the Select Best component.

The best accuracy the Gaussian Naive Bayes Classifier achieves is 60% with many Preprocessors (Max Abs Scaler, Min Max Scaler or Robust Scaler) and the Select Best component.

The rest of the results can be seen in Table 6.10.

With Exclude

The use of `Excluded = 1` has a positive effect, especially with K Nearest Neighbours.

The best scoring Classifier is now tied between Decision Tree and K Nearest Neighbours Classifiers, both having a score of 78%.

The Decision Tree achieves this score with Quantile Transformer as a Preprocessor and the Select Best component.

The best accuracy for the K Nearest Neighbours Classifier is achieved with Robust Scaler as a Preprocessor and the Select Best component.

The best accuracy the Gaussian Naive Bayes Classifier achieves is 62% with any Preprocessor and the Select Best component.

The rest of the results can be seen in Table 6.11.

Steps	Decision Tree(%) Excluded	Gaussian Naive Bayes(%) Excluded	K Nearest Neighbors(%) Excluded
-	55.00 1	42.00 1	48.00 1
Max Abs Scaler	54.00 1	42.00 1	48.00 1
Max Abs Scaler,Select Best	68.00 1	51.00 1	61.00 1
Min Max Scaler	55.00 1	42.00 1	42.00 1
Min Max Scaler,Select Best	68.00 1	51.00 1	60.00 1
Quantile Transformer	56.00 1	46.00 1	51.00 1
Quantile Transformer,Select Best	**69.00** 1	53.00 1	64.00 1
Robust Scaler	57.00 1	42.00 1	49.00 1
Robust Scaler,Select Best	68.00 1	51.00 1	60.00 1

Table 6.9: Results of by Partition in Gutenberg, using Excluded = 1.

Steps	Decision Tree(%)	Gaussian Naive Bayes(%)	K Nearest Neighbors(%)
-	57.00	52.00	35.00
Max Abs Scaler	55.00	52.00	42.00
Max Abs Scaler,Select Best	63.00	60.00	48.00
Min Max Scaler	55.00	52.00	42.00
Min Max Scaler,Select Best	63.00	60.00	45.00
Quantile Transformer	57.00	43.00	40.00
Quantile Transformer,Select Best	**65.00**	52.00	47.00
Robust Scaler	53.00	52.00	42.00
Robust Scaler,Select Best	**65.00**	60.00	47.00

Table 6.10: Results of by Partition, in Gutenberg 6, without the use of Excluded.

Steps	Decision Tree(%) Excluded	Gaussian Naive Bayes(%) Excluded	K Nearest Neighbors(%) Excluded
-	63.00 1	55.00 1	52.00 1
Max Abs Scaler	63.00 1	55.00 1	58.00 1
Max Abs Scaler,Select Best	77.00 1	62.00 1	77.00 1
Min Max Scaler	63.00 1	55.00 1	57.00 1
Min Max Scaler,Select Best	77.00 1	62.00 1	77.00 1
Quantile Transformer	63.00 1	58.00 1	52.00 1
Quantile Transformer,Select Best	**78.00** 1	62.00 1	73.00 1
Robust Scaler	63.00 1	55.00 1	58.00 1
Robust Scaler,Select Best	77.00 1	62.00 1	**78.00** 1

Table 6.11: Results of by Partition in Gutenberg 6, using Excluded = 1.

6.2.3 Portuguese

Without Exclude

The best accuracy of 100% is achieved using Decision Tree as the Classifier. This is achieved with any Preprocessor and the Select Best component.

The best accuracy the K Nearest Neighbours Classifier achieves is 89% with any Preprocessor except Max Abs Scaler, and the Select Best component.

The best accuracy the Gaussian Naive Bayes Classifier achieves is 89% with any Preprocessor, and the Select Best component.

The rest of the results can be seen in Table 6.12.

With Exclude

The use of `Excluded = 1` has a negative effect for some configurations, but a positive one for others.

The best scoring Classifier is now tied between Decision Tree and K Nearest Neighbours Classifiers with 100%.

The Decision Tree achieves this score with any Preprocessor and the Select Best component.

The best accuracy for the K Nearest Neighbours Classifier is achieved with any Preprocessor and the Select Best component.

Steps	Decision Tree(%)	Gaussian Naive Bayes(%)	K Nearest Neighbors(%)
-	78.00	67.00	67.00
Max Abs Scaler	78.00	67.00	67.00
Max Abs Scaler,Select Best	**100.00**	89.00	78.00
Min Max Scaler	78.00	67.00	67.00
Min Max Scaler,Select Best	**100.00**	89.00	89.00
Quantile Transformer	78.00	67.00	67.00
Quantile Transformer,Select Best	**100.00**	89.00	89.00
Robust Scaler	78.00	67.00	67.00
Robust Scaler,Select Best	**100.00**	89.00	89.00

Table 6.12: Results of by Partition, in Portuguese, without the use of Excluded.

The best accuracy the Gaussian Naive Bayes Classifier achieves is 89% with any Preprocessor and the Select Best Component.

The rest of the results can be seen in Table 6.13.

6.3 By Document results

The *By Document* architecture uses the components: Preprocessor, Isomap and Classifier. In addition, to measure the impact of using a Preprocessor and Isomap, tests are performed without these components. Because some measures scale heavily with graph size, they were not calculated. The calculated measures are: *clustering coefficient, diameter, radius, cliques, transitivity, shortest path, degree, nodes number, edges number* and *intermittency*. The measures that aren't calculated are: *load centralities* and *betweenness centrality*. The tested configurations can be seen below:

- Preprocessor $\rightarrow$ Isomap $\rightarrow$ Classifier
- Preprocessor $\rightarrow$ Classifier
- Classifier

6.3.1 Gutenberg

The best accuracy of 53% is achieved using K Nearest Neighbours as the Classifier. This score is achieved with two configurations: a Max Abs Scaler Preprocessor and Isomap; a Min Max Scaler Preprocessor but without the use of Isomap.

The best accuracy the Decision Tree Classifier achieves is 45% with Robust Scaler as the Preprocessor and Isomap.

The best accuracy the Gaussian Naive Bayes Classifier achieves is 48% with Robust Scaler as the Preprocessor and Isomap.

The rest of the results can be seen in Table 6.14.

Steps	Decision Tree(%) Excluded	Gaussian Naive Bayes(%) Excluded	K Nearest Neighbors(%) Excluded
-	67.00 1	56.00 1	78.00 1
Max Abs Scaler	67.00 1	56.00 1	78.00 1
Max Abs Scaler,Select Best	**100.00** 1	89.00 1	**100.00** 1
Min Max Scaler	67.00 1	56.00 1	78.00 1
Min Max Scaler,Select Best	**100.00** 1	89.00 1	**100.00** 1
Quantile Transformer	67.00 1	67.00 1	67.00 1
Quantile Transformer,Select Best	**100.00** 1	89.00 1	**100.00** 1
Robust Scaler	67.00 1	56.00 1	67.00 1
Robust Scaler,Select Best	**100.00** 1	89.00 1	**100.00** 1

Table 6.13: Results of by Partition in Portuguese, using Excluded = 1.

Steps	Decision Tree(%)	Gaussian Naive Bayes(%)	K Nearest Neighbors(%)
-	40.00	45.00	31.00
Max Abs Scaler	40.00	46.00	49.00
Max Abs Scaler, Isomap	39.00	45.00	**53.00**
Min Max Scaler	40.00	46.00	**53.00**
Min Max Scaler, Isomap	33.00	46.00	51.00
Quantile Transformer	41.00	45.00	46.00
Quantile Transformer, Isomap	41.00	45.00	41.00
Robust Scaler	40.00	46.00	45.00
Robust Scaler, Isomap	45.00	48.00	46.00

Table 6.14: Results of By Document in Gutenberg

Steps	Decision Tree(%)	Gaussian Naive Bayes(%)	K Nearest Neighbors(%)
-	52.00	52.00	32.00
Max Abs Scaler	53.00	52.00	62.00
Max Abs Scaler, Isomap	47.00	52.00	52.00
Min Max Scaler	52.00	52.00	60.00
Min Max Scaler, Isomap	52.00	48.00	55.00
Quantile Transformer	52.00	53.00	58.00
Quantile Transformer, Isomap	43.00	52.00	52.00
Robust Scaler	52.00	52.00	**63.00**
Robust Scaler, Isomap	50.00	52.00	52.00

Table 6.15: Results of By Document in Gutenberg 6

Steps	Decision Tree(%)	Gaussian Naive Bayes(%)	K Nearest Neighbors(%)
-	67.00	67.00	22.00
Max Abs Scaler	67.00	33.00	56.00
Max Abs Scaler, Isomap	**78.00**	33.00	56.00
Min Max Scaler	67.00	33.00	44.00
Min Max Scaler, Isomap	56.00	33.00	44.00
Quantile Transformer	67.00	56.00	44.00
Quantile Transformer, Isomap	56.00	33.00	44.00
Robust Scaler	67.00	33.00	56.00
Robust Scaler, Isomap	67.00	22.00	56.00

Table 6.16: Results of By Document in Portuguese

6.3.2 Gutenberg 6

The best accuracy of 63% is achieved using K Nearest Neighbours as the Classifier. This score is achieved with Robust Scaler as the Preprocessor but without the use of Isomap.

The best accuracy the Decision Tree Classifier achieves is 53% with Max Abs Scaler as the Preprocessor but without the use of Isomap.

The best accuracy the Gaussian Naive Bayes Classifier achieves is 53% with Quantile Transformer as the Preprocessor but without the use of Isomap.

The rest of the results can be seen in Table 6.15.

6.3.3 Portuguese

The best accuracy of 78% is achieved using Decision Tree as the Classifier. This score is achieved with Max Abs Scaler as the Preprocessor and Isomap.

The best accuracy the K Nearest Neighbours achieves is 56% with Robust Scaler (or Max Abs Scaler) as the Preprocessor; the use of Isomap doesn't influence these results.

The best accuracy the Gaussian Naive Bayes Classifier achieves is 67% with no Preprocessor or Isomap.

The rest of the results can be seen in Table 6.16.

7 Conclusions and Future Work

This thesis explores 3 different architectures to solve the problem of Authorship Attribution: *By Time Series*, *By Partition* and *By Document*. These architectures use graphs to represent the text in different ways. From theses graphs, measures are extracted and used differently in each architecture.

The *By Time Series* architecture gave unexpected results, they are very different from the ones presented in [AAO17], where the best score was 88.75%, with K Nearest Neighbours Classifier; this could be due to a different tokenizer (not specified), different implementations of the various components or how the metadata of the corpus is handled. In this architecture the use of the custom component Exclude generally improves the results, even if only in the 1-2% range.

For the same corpus - Gutenberg - and same components (at least in theory) the score achieved is around 40%. On the other hand, the best configuration in this thesis achieved a score of 61%. This is using Gaussian Naive Bayes and a custom component - Exclude - to get slightly better results. For the Gutenberg 6 corpus, a slimmed down version of the corpus from the point above, the results are better: 72% using Decision Tree. Still the same issue happens, K Nearest Neighbours has lower than expected results. On another corpus, Portuguese, the best configuration uses, once again Gaussian Naive Bayes to achieve a score of 89%. To note that this corpus has many less texts and authors.

The *By Partition* architecture is dominated by the Decision Tree classifier, as it always seems to yield the best results. The order of results is slightly above that of the *By Time Series* architecture. In general, in this architecture, the use of Exclude seems to improve the results. This architecture also introduces a new component - Select Best - which selects the best feature set from the available. In future work, it would be good to study the use of higher values of Exclude, as this component shows promise to a certain point. Another important aspect is how the author is decided from the various individual predictions; it would be good to explore other voting mechanisms.

For the Gutenberg corpus, the best configuration in this thesis achieved a score of 69%. This is using Gaussian Naive Bayes, the custom component Exclude and the custom component Select Best. For the Gutenberg 6 corpus, the results are better: 78% using Decision Tree. This is using the custom component Exclude and the custom component Select Best. On another corpus, Portuguese, the best configuration uses, once again, Decision Tree to achieve a score of 100%. This is achieved due to the Select Best component.

The last architecture, *By Document*, has the worst results. This is not unexpected, however, as this is the same conclusion as [AAO17]: by considering the whole text at once, small patterns are lost that wouldn't otherwise be, by considering each partition of text. This architecture doesn't use any custom components. In the future it would be good to study if the remaining features could improve the results.

For the Gutenberg corpus, the best configuration in this thesis achieved a score of 53%, using K Nearest Neighbours. For the Gutenberg 6 corpus, the results are better: 63% using K Nearest Neighbours. On another corpus, Portuguese, the best configuration uses, once again, Decision Tree to achieve a score of 78%.

Both components proposed in this thesis - Exclude and Select Best - improve the results in some way. As such, for the future, it might be worth exploring variations of these components. In regards to the Exclude component, instead of excluding the values furthest from the average, one could exclude the partitions whose values are always furthest from the average. In regards to the Select Best component, since it had such an positive effect in the results, it would be good to find an alternative for the *By Time Series* architecture, perhaps a hill climb adaptation.

Finally, the solution presented in this thesis could be used for Author Recommendation by using a text written by an author that is not in the trained authors.

A Corpus Documents

A.1 Gutenberg

Jonathan Swift

- Gulliver's Travels into Several Remote Nations of the World
- The Journal to Stella
- The Poems of Jonathan Swift, D.D., Volume 1
- The Poems of Jonathan Swift, D.D., Volume 2
- The Prose Works of Jonathan Swift, D.D. Volume 03
- The Prose Works of Jonathan Swift, D.D. Volume 04
- The Prose Works of Jonathan Swift, D.D. Volume 06

- The Prose Works of Jonathan Swift, D.D. Volume 07
- The Prose Works of Jonathan Swift, D.D. Volume 09
- The Prose Works of Jonathan Swift, D.D. Volume 10

Jack London

- Burning Daylight
- Martin Eden
- Michael, Brother of Jerry
- Smoke Bellew
- The Iron Heel
- The Jacket (The Star-Rover)
- The Little Lady of the Big House
- The Mutiny of the Elsinore
- The Sea-Wolf
- The Valley of the Moon

Arthur Conan Doyle

- A Study in Scarlet
- His Last Bow An Epilogue of Sherlock Holmes
- The Adventures of Sherlock Holmes
- The Gully of Bluemansdyke
- The Hound of the Baskervilles
- The Lost World
- The Memoirs of Sherlock Holmes
- The Return of Sherlock Holmes
- The Sign of the Four
- The Valley of Fear

Fyodor Dostoyevsky

- Crime and Punishment
- Notes from the Underground

- Poor Folk
- Short Stories
- The Brothers Karamazov
- The Gambler
- The Idiot
- The Possessed (The Devils)
- Uncle's Dream; and The Permanent Husband
- White Nights and Other Stories

Nathaniel Hawthorne

- Doctor Grimshawe's Secret
- Our Old Home A Series of English Sketches
- Passages from the English Notebooks, Complete
- Passages from the French and Italian Notebooks, Volume 1
- Passages from the French and Italian Notebooks, Volume 2
- Sketches and Studies
- The Marble Faun; Or, The Romance of Monte Beni - Volume 2
- The Scarlet Letter
- True Stories of History and Biography
- Twice Told Tales

Herman Melville

- Mardi and A Voyage Thither, Vol. I
- Mardi and A Voyage Thither, Vol. II
- Moby Dick; Or, The Whale
- Omoo Adventures in the South Seas
- Pierre; or The Ambiguities
- Redburn. His First Voyage
- The Confidence-Man His Masquerade
- The Piazza Tales
- Typee A Romance of the South Seas

- White Jacket; Or, The World on a Man-of-War

Leo Tolstoy

- Anna Karenina
- Redemption and two other plays
- Resurrection
- The Awakening
- The Cossacks A Tale of 1852
- The Kingdom of God Is Within You
- War and Peace
- What Shall We Do
- What to Do Thoughts Evoked by the Census of Moscow
- Youth

Bernard Shaw

- An Unsocial Socialist
- Caesar and Cleopatra
- Cashel Byron's Profession
- John Bull's Other Island
- Major Barbara
- Man and Superman A Comedy and a Philosophy
- Mrs. Warren's Profession
- On the Prospects of Christianity
- Pygmalion
- Treatise on Parents and Children

A.2 Gutenberg 6

Jonathan Swift

- Gulliver's Travels into Several Remote Nations of the World
- The Journal to Stella

- The Poems of Jonathan Swift, D.D., Volume 1
- The Poems of Jonathan Swift, D.D., Volume 2
- The Prose Works of Jonathan Swift, D.D. Volume 03
- The Prose Works of Jonathan Swift, D.D. Volume 04
- The Prose Works of Jonathan Swift, D.D. Volume 06
- The Prose Works of Jonathan Swift, D.D. Volume 07
- The Prose Works of Jonathan Swift, D.D. Volume 09
- The Prose Works of Jonathan Swift, D.D. Volume 10

Jack London

- Burning Daylight
- Martin Eden
- Michael, Brother of Jerry
- Smoke Bellew
- The Iron Heel
- The Jacket (The Star-Rover)
- The Little Lady of the Big House
- The Mutiny of the Elsinore
- The Sea-Wolf
- The Valley of the Moon

Arthur Conan Doyle

- A Study in Scarlet
- His Last Bow An Epilogue of Sherlock Holmes
- The Adventures of Sherlock Holmes
- The Gully of Bluemansdyke
- The Hound of the Baskervilles
- The Lost World
- The Memoirs of Sherlock Holmes
- The Return of Sherlock Holmes
- The Sign of the Four

- The Valley of Fear

Fyodor Dostoyevsky

- Crime and Punishment
- Notes from the Underground
- Poor Folk
- Short Stories
- The Brothers Karamazov
- The Gambler
- The Idiot
- The Possessed (The Devils)
- Uncle's Dream; and The Permanent Husband
- White Nights and Other Stories

Nathaniel Hawthorne

- Doctor Grimshawe's Secret
- Our Old Home A Series of English Sketches
- Passages from the English Notebooks, Complete
- Passages from the French and Italian Notebooks, Volume 1
- Passages from the French and Italian Notebooks, Volume 2
- Sketches and Studies
- The Marble Faun; Or, The Romance of Monte Beni - Volume 2
- The Scarlet Letter
- True Stories of History and Biography
- Twice Told Tales

Leo Tolstoy

- Anna Karenina
- Redemption and two other plays
- Resurrection
- The Awakening

- The Cossacks A Tale of 1852
- The Kingdom of God Is Within You
- War and Peace
- What Shall We Do
- What to Do Thoughts Evoked by the Census of Moscow
- Youth

A.3 Portuguese

António Lobo Antunes

- A Ordem Natural das Coisas
- As Naus
- Auto Dos Danados

José Saramago

- A Jangada De Pedra
- As Intermitencias da Morte
- Levantado do Chao

Mia Couto

- Jesusalem
- O Outro Pe da Sereia
- Vinte e Zinco

Bibliography

[AAO17] Camilo Akimushkin, Diego Raphael Amancio, and Osvaldo Novais Oliveira, Jr. Text authorship identified using the dynamics of word co-occurrence networks. *PLOS ONE*, 12(1):1–15, 01 2017.

[ABR19] Dr Anwar, Imran Bajwa, and Shabana Ramzan. Design and implementation of a machine learning-based authorship identification model. *Scientific Programming*, 2019:14, 01 2019.

[Ama15] Diego R Amancio. Authorship recognition via fluctuation analysis of network topology and word intermittency. *Journal of Statistical Mechanics: Theory and Experiment*, 2015(3):P03005, Mar 2015.

[BBU07] Ilker Bozkurt, O. Baghoglu, and Erkan Uyar. Authorship attribution. volume 1, pages 1 – 5, 12 2007.

[CMVPMR06] Rosa Coyotl-Morales, Luis Villaseñor-Pineda, Manuel Montes, and Paolo Rosso. Authorship attribution using word sequences. volume 4225, pages 844–853, 11 2006.

[GAPDSP18] Helena Gómez-Adorno, Juan-Pablo Posadas-Durán, Grigori Sidorov, and David Pinto. Document embeddings learned on various types of n-grams for cross-topic authorship attribution. *Computing*, 100(7):741–756, July 2018.

[GASP$^+$16] Helena Gomez Adorno, Grigori Sidorov, David Pinto, Darnes Vilariño, and Alexander Gelbukh. Automatic authorship detection using textual patterns extracted from integrated syntactic graphs. *Sensors*, 16:1374, 08 2016.

[HMVLB20] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python. 2020.

[HSS08] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[IR19] Rahul Radhakrishnan Iyer and Carolyn Penstein Rosé. A machine learning framework for authorship identification from texts. *ArXiv*, abs/1912.10204, 2019.

[Juo08] Patrick Juola. Authorship attribution. *Foundations and Trends® in Information Retrieval*, 1(3):233–334, 2008.

[MHM20] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020.

[PB18] Jagadeesh Patchala and Raj Bhatnagar. Authorship attribution by consensus among multiple features. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2766–2777, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[PS17] Nektaria Potha and Efstathios Stamatatos. An improved impostors method for authorship verification. In *CLEF*, pages 138–144, 08 2017.

[PVG$^+$11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[QHZ17] Chen Qian, Ting He, and Rao Zhang. Deep learning based authorship identification. 2017.

[Rho15] Dylan Rhodes. Author attribution with cnn's. 2015.

[Sei13] Shachar Seidman. Authorship verification using the impostors method notebook for pan at clef 2013. In *CLEF*, 2013.

[SSV18] Yunita Sari, Mark Stevenson, and Andreas Vlachos. Topic or style? exploring the most useful features for authorship attribution. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 343–353, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[Sta09] Efstathios Stamatatos. A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology*, 60(3):538–556, 2009.

[Sta17] Efstathios Stamatatos. Authorship attribution using text distortion. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1138–1149, Valencia, Spain, April 2017. Association for Computational Linguistics.