Dissertation

Intrinsic Plagiarism Detection and Author Analysis By Utilizing Grammar

Michael Tschuggnall

submitted to the Faculty of Mathematics, Computer Science and Physics of the University of Innsbruck

in partial fulfillment of the requirements for the degree of "Doctor of Philosophy"

Advisor: Univ.-Prof. Dr. Günther Specht

Innsbruck, 2014

Abstract

With the advent of the world wide web the number of freely available text documents has increased considerably in the last years. As one of the immediate results, it has become easier to find sources that serve as the basis for plagiarism. On the other side, it has become harder for detection tools to automatically expose plagiarism due to the huge amount of possible origins. Moreover, sources may even not be digitally available, resulting in an unsolvable problem for such tools, whereas experienced human readers might find suspicious passages based on an intuitive style analysis.

In this thesis, intrinsic plagiarism detection algorithms are proposed which operate on the suspicious document only and circumvent the problem of incorporating external data. The main idea is thereby to analyze the style of authors in terms of the grammar that is used to formulate sentences, and to expose significantly outstanding text fragments according to the syntax, which is represented by grammar trees. By using a similar style analysis, the idea has also been applied to the problem of automatically assigning authors to unseen text documents. Moreover, it is shown that grammar also serves as a distinguishing feature to profile an author, namely to predict his/her gender and age. Reusing all previous analyses and results, the idea has finally been adapted in order to be used to automatically detect different authorships in a collaboratively written document.

Zusammenfassung

Die Anzahl an frei verfügbaren Textdokumenten ist in den letzten Jahren aufgrund des enormen Aufschwungs des Internets erheblich gestiegen. Eine der Konsequenzen ist, dass Quellen für mögliche Plagiate leicht gefunden werden können, während es auf der anderen Seite für automatische Erkennungstools aufgrund der großen Datenmengen immer schwieriger wird, Plagiate zu erkennen. Zudem sind Quellen oft nicht in digitaler Form vorhanden, was für Tools, die auf Vergleiche mit bekannten Dokumenten basieren, ein unlösbares Problem darstellt. Andererseits können geübte menschliche Leser verdächtige Passagen oft über eine intuitive Stilanalyse ausfindig machen.

In dieser Arbeit werden verschiedene Algorithmen zur intrinsischen Plagiatserkennung entwickelt, welche ausschließlich das zu prüfende Dokument untersuchen und so das Problem umgehen, externe Daten heranziehen zu müssen. Dabei besteht die Grundidee darin, den Schreibstil von Autoren auf Basis der von ihnen verwendeten Grammatik zur Formulierung von Sätzen zu untersuchen, und diese Information zu nutzen, um syntaktisch auffällige Textfragmente zu identifizieren. Unter Verwendung einer ähnlichen Analyse wird diese Idee auch auf das Problem, Textdokumente automatisch Autoren zuzuordnen, angewendet. Darüber hinaus wird gezeigt, dass die verwendete Grammatik auch ein unterscheidbares Kriterium darstellt, um Informationen wie das Geschlecht und das Alter des Verfassers abzuschätzen. Schlussendlich werden die vorherigen Analysen und Resultate verwendet und so adaptiert, dass Anteile von verschiedene Autoren in einem gemeinschaftlich verfassten Text automatisch erkannt werden können.

Acknowledgements

"First of all, I have to say", that a comprehensive work like this thesis can obviously not be completed without the continuous help of many others. Because I am usually one of the kind that has some problems expressing gratitude - especially in a private environment - this is a perfect opportunity for me to thank those who guided, assisted, advised, motivated, mothered and also distracted me at the right moments in order to complete this thesis - and as a fact of which I'm proud of: **in time!**

Every thesis needs a relaxing but yet productive environment to be developed and written in: and this was provided by the DBIS group. At first I want to thank my advisor Günther Specht for giving me the opportunity to write the thesis, but also for always having an open ear for any emerging problems and providing the mentioned environment, leaving space for developing and following ideas. Moreover, I want to thank all DBIS members I work/ed with: Domi, Doris, Eva, Gabi, Martin, Michael, Niko, Peter, Robert, Seppi, Sylvia, Wolfi - thank you for always having time for helping me in any matter, discussing scientific and also non-scientific topics.

Undoubtedly the biggest thank goes to my wife Claudia for her loving care and never-ending support: not only for the time this thesis was developed and written, but basically for all decisions I made and for all life-changing directions I went in the last decade. You steadily supported me as much as possible while always keeping an eye on the maintenance of our relationship, which includes taking the necessary steps like suggesting a spontaneous mind-freeing short trip. I also want to thank my son Noah (you're exactly one year at this time) - especially for the "distraction"-part: one cannot think about a thesis improvement while changing diapers or playing guitar and singing children's songs. Consequently, you forced me to work even harder and more efficiently during office times - thanks!

I also want to thank my family: my parents, my brothers and my parents-in-law for providing me a 24/7 open and solid home, helping me in many matters and for generally having an open ear for anything. To my grandparents: thank you for the many nice and endless discussions, and also for cooking regularly since the beginning of my study: you waited long and patiently, but you finally have a "Doctor" within your family - congratulations!

A special thank goes to my uncle for continuously giving me interesting popular-scientific input and also for the time spent reading and commenting many of my publications.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt durch meine eigenhändige Unterschrift, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Stellen, die wörtlich oder inhaltlich den angegebenen Quellen entnommen wurden, sind als solche kenntlich gemacht. Die vorliegende Arbeit wurde bisher in gleicher oder ähnlicher Form noch nicht als Magister-/Master-/Diplomarbeit/Dissertation eingereicht.

Datum	Michael Tschuggnall

Contents

CHAPTER 1

Introduction

1.1. Motivation

With the advent of electronic data processing in combination with the global world wide web the amount of publicly available text documents is huge and increasing daily. Besides the existence of online libraries like Project Gutenberg [180] or Open Library [179] that offer free downloads of millions of e-books, textual content is also spread massively through social media applications. Here, users frequently use the numerous possibilities to compose and share text in various ways. Considering current statistics [171] estimating 70 billion pieces of content shared via Facebook or 190 million short messages posted on Twitter every day, the amount of shared textual information is huge.

Where the authors of text shared through social media like status posts or web blogs are usually known and easily classifiable, the usage and publishing of text becomes problematic as soon as copyright issues are involved. Especially in academia, recent events show that textual content is frequently copied, modified and claimed to be an author's own work without appropriate citation, despite the fact that it clearly isn't. Such cases of plagiarism can be detected with relatively few effort if text fragments are taken from easily available and popular sources like Wikipedia. In those situations simple algo-

rithms can be used which try to find plagiarism by basically just comparing a document against a large internal text document database. By utilizing approximate string matching algorithms like the Levenshtein Distance [104] or longest common subsequence algorithms [17], even copied and slightly modified text can be detected to a certain extent.

On the other side, an automatic detection becomes substantially more difficult when text is vastly rearranged or when the source document is not available. In latter cases an internal analysis of the document regarding the writing style is indispensable. For humans it is often easy to detect style shifts within a text block, but for computational algorithms it remains a hard problem. As an example, advisors of student works like seminar papers or bachelor theses in an academic area repeatedly find plagiarized sequences of sentences, because they seem *odd*.

Such human estimations are mostly based on the intuitive detection of changes in the writing style, including measures like the usage or richness of vocabulary, the (average) length of sentences or the complexity of the grammar used. The computer-based analysis of such style shifts considering many stylistic characterstics in order to expose plagiarism in text documents is usually called *intrinsic plagiarism detection* in scientific communities. This thesis contributes to this research field and introduces different algorithms based on a novel style feature that is able to significantly distinguish between the writing styles of different authors, achieving promising results. In particular, the grammar syntax of writers is analyzed and utilized in the fields of intrinsic plagiarism detection, authorship attribution, author profiling and decomposition of multi-author documents.

1.2. Research Objectives

The main idea of this thesis is based on the assumption that different authors have different writing styles in terms of the grammar they use. Therefore the sentences of documents are analyzed by their grammar, i.e., by inspecting plain POS tags or full parse trees. The information gained is then consequently applied to different research fields, whereby the basic question for each category is whether a grammar analysis using the algorithms presented in this thesis is sufficient to (a) represent a standalone approach or to (b) enhance existing state-of-the-art algorithms.

Intrinsic Plagiarism Detection

According to its definition, the main goal of intrinsic plagiarism detection is to find plagiarism in text documents by inspecting a suspicious document only. In contrast to external detection algorithms which make use of large databases or even internet search engines for extensive comparisons, intrinsic approaches are supposed to detect plagiarism by applying style analysis and finding irregular patterns.

The research question discussed in this thesis is as follows: Can solely grammar analysis be used to identify plagiarized passages in a suspicious document?

Authorship Attribution

The objective of traditional authorship attribution is to assign known authors to previously unseen documents. Usually, several text documents for each candidate author are known, from which the algorithms have to learn in order to be able to make correct predictions. If the restriction is given to assign one of the candidates to be the author of the document in question, the problem is called *closed-class* attribution. On the other hand, if additionally a "non-of-them" answer is allowed, this (more difficult) problem is usually referred to as *open-class*.

In this thesis the following question is evaluated: Can solely grammar analysis be used to learn from text samples and to correctly closed-class predict authorships of unseen documents?

Automatic Author Profiling

In contrast to the traditional authorship attribution problem, the task of automated author profiling is not to assign authorships to documents, but to predict meta information about the author of an unseen document. Such meta information includes gender, age or the geographic origin of the author, but also psychological classifications.

This thesis investigates on the question whether the grammar of authors can be used to reliably determine their gender and age.

Multi-Author Decomposition

Finally, the discrimination of authorships of a multi-author document is a task which is closely related to intrinsic plagiarism detection, and thus tries to separate text passages that are written by different authors. The main difference is that - in distinction to plagiarism detection - several authors may

have collaborated on a document, and that the amount of contribution may be equally distributed per author. Consequently, the assumption that a *main* author exists cannot be used, and state-of-the-art clustering techniques have to be utilized.

Using a similar grammar analysis like in the other subproblems, the following question is evaluated: Is the grammar of authors sufficient in order to be used as input for modern clustering algorithms, so that authorships can be descriminated in a document and correct author clusters can be built?

1.3. Published Work

Throughout the PhD studies several works have been published in international, peer-reviewed scientific conference proceedings. Each publication describes a part of this thesis and is used as a basis for the respective chapter.

In particular, the following papers have been published:

First-Author Conference Papers

- M. Tschuggnall and G. Specht. *Plag-Inn: Intrinsic Plagiarism Detection Using Grammar Trees.* In Proceedings of the 17th International Conference on Application of Natural Language to Information Systems (NLDB), Groningen, The Netherlands, June 2012, volume 7337 of LNCS, Springer, pages 284–289. [185]

- M. Tschuggnall and G. Specht. *Detecting Plagiarism in Text Documents Through Grammar-Analysis of Authors.* In Proceedings of the 15. GI-Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW), Magdeburg, Germany, March 2013, volume 214 of LNI, pages 241–259. [187]

- M. Tschuggnall and G. Specht. *Countering Plagiarism by Exposing Irregularities in Authors' Grammar.* In Proceedings of the European Intelligence and Security Informatics Conference (EISIC), Uppsala, Sweden, August 2013, IEEE, pages 15–22. [186]

- M. Tschuggnall and G. Specht. *Using Grammar-Profiles to Intrinsically Expose Plagiarism in Text Documents.* In Proceedings of the 18th International Conference on Application of Natural Language to Information Systems (NLDB), Salford, UK, June 2013, volume 7934 of LNCS, Springer, pages 297–302. [188]

- M. Tschuggnall and G. Specht. *Enhancing Authorship Attribution by Utilizing Syntax Tree Profiles.* In Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL), Gothenburg, Sweden, April 2014, volume 2: Short Papers, Association for Computational Linguistics, pages 195–199. [190]

- M. Tschuggnall and G. Specht. *What Grammar Tells About Gender and Age of Authors.* In Proceedings of the 4th International Conference on Advances in Information Mining and Management (IMMM), Paris, France, July 2014, pages 30–35. [191]

Book Contributions

- M. Tschuggnall. *Plag-Inn: Uncovering Plagiarism by Examining Author's Grammar Syntax.* In M. Barden and A. Ostermann, editors, Scientific Computing@uibk. Innsbruck University Press, 2013. [184]

Workshop Contributions

- M. Tschuggnall and G. Specht. *Automatic Decomposition of Multi-Author Documents Using Grammar Analysis.* In Proceedings of the 26th GI-Workshop on Grundlagen von Datenbanken (GvD), Bozen, Italy, October 2014. [189]

Other Contributions

The following publication is a result of the author's Master thesis in the field of recommender systems:

- W. Gassler, E. Zangerle, M. Tschuggnall, and G. Specht. SnoopyDB: *Narrowing the Gap between Structured and Unstructured Information using Recommendations.* In Proceedings of the 21st ACM Conference on Hypertext and Hypermedia (HT), Toronto, Ontario, Canada, June 2010, pages 271-272. [51]

1.4. Thesis Outline

The remainder of this thesis is structured as follows:

As one of the main contributions of this work, Chapter 2 discusses the utilization of grammar structures to intrinsically detect plagiarism in text documents. After showing the basics used for the grammar analysis in Section 2.2 and Section 2.3, respectively, the elementary *Plag-Inn* algorithm (Section 2.4)

as well as the variants *POS-Plag-Inn* (Section 2.5) and *PQ-Plag-Inn* (Section 2.6) are explained and evaluated in detail.

The application of the grammar analysis to the authorship attribution problem is shown in Chapter 3, whereas the application to the automatic profiling of gender and age is discussed in Chapter 4. Finally, Chapter 5 explains the automatic clustering of text paragraphs by using the grammar style of authors.

Related work of all research objectives, i.e., intrinsic plagiarism detection, authorship attribution, author profiling and clustering is described in Chapter 6. Finally, Chapter 7 concludes the contributions of this thesis and further discusses future work.

CHAPTER 2

Intrinsic Plagiarism Detection

2.1. Introduction

Today more and more text documents are made publicly available through large text collections or literary databases. As recent events show, the detection of plagiarism in such systems becomes considerably more important as it is very easy for a plagiarist to find an appropriate text fragment that can be copied, where on the other side it becomes increasingly harder to correctly identify plagiarized sections due to the huge amount of possible sources. In this thesis novel approaches to detect plagiarism in text documents are presented, that circumvent large data comparisons by performing intrinsic data analysis, i.e., analysis of grammar syntax.

The two main approaches for identifying plagiarism in text documents are known as *external* and *intrinsic* algorithms [142], which are illustrated in Figure 6.2. External algorithms compare a suspicious document against a given, unrestricted set of documents obtained from multiple databases created from sources like open libraries, freely available published academic papers or the world wide web in general, often by incorporating search engines. Basically, the suspicious document is split into several segments, whereby every segment is then compared against every possible document in the data set. Often applied

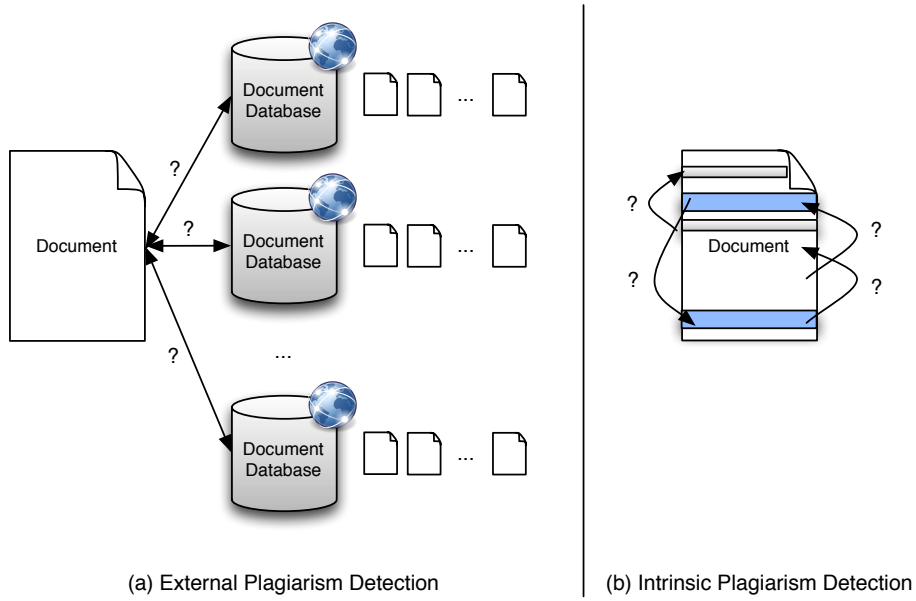


Figure 2.1.: Difference Between External and Intrinsic Plagiarism Detection.

techniques used in external approaches include n-grams [131], word-n-grams [14] comparisons or standard IR techniques like common subsequences [61]. Moreover, machine learning techniques [15] are also heavily utilized, whereby the main scientific contribution is to present new features or to intelligently select existing ones.

On the other side, intrinsic approaches are allowed to inspect the suspicious document only and have to comprise the writing style of an author in some way. The challenging task is thereby to find irregular text sequences within the document based on several measures. Among others, features like the frequency of words from predefined word-classes (vocabulary) [132], complexity analysis [157] or n-grams [169, 87] as well are used to find plagiarized sections.

Although the majority of external algorithms perform significantly better than intrinsic algorithms by using the advantage of a huge data set gained from the Internet, intrinsic methods are useful when such a data set is not available. For example, in scientific documents that use information mainly from books which are not digitally available, a proof of authenticity is nearly impossible for a computer system to make. Moreover, authors may modify the source text in such a way that even advanced, fault-tolerant text comparison algorithms cannot detect similarities. In addition, intrinsic approaches can be used as a preceding technique to help reducing the set of source documents for CPU- and/or memory-intensive external procedures.

As a reference for all evaluation results shown in this chapter and to strengthen the difficulty of the problem, the detection rates (F-scores) of state-of-the-art intrinsic plagiarism systems should be considered, which range from about 8% up to 32% only, depending on the test data. Like it is shown later in this chapter, these detection rates could be met and even outperformed by the algorithms developed in this thesis. On the other hand and as stated before, external algorithms perform significantly better and achieve detection rates of up to about 80%. A more detailed summary on the algorithms and performances of related work is given in Section 6.1.

The rest of this chapter is organized as follows: At first, Section 2.2 gives an overview of the grammar syntax used by authors, and subsequently Section 2.3 recaps the concept of pq-grams and pq-gram indices, as they are used extensively throughout this thesis to analyze grammar. The intrinsic plagiarism detection algorithm *Plag-Inn* using the latter by structurally comparing all sentences of a document is described in Section 2.4. The *POS-Plag-Inn* algorithm shown in Section 2.5 is also based on a sentence-by-sentence comparison, but uses POS tags only, which are compared by utilizing dynamic programming algorithms. Finally, in Section 2.6 the *PQ-PlagInn* algorithm is explained, which also investigates on structural differences of grammar trees by comparing pq-gram profiles of sentence windows.

2.2. The Grammar Syntax of Authors

Every natural language is based on a set of vocabulary and a set of grammar rules that allow its users to build sentences, whereby both numbers differ for each language. For example, the Oxford English Dictionary lists about 170,000 distinct words that are currently used [133], whereas the German Duden estimates the German vocabulary to contain 300,000 - 500,000 distinct words [19]. More importantly, the number of actually used words is of interest when formulating sentences. Here, a recent study[1] incorporating more than two million native English speakers found out that the average vocabulary size of an adult ranges between 20,000 and 35,000 words.

2.2.1. Grammar Rules

As a result of the evolution of a language like English for several thousand years [60], it provides numerous valid possibilities to transmit a single message. First, a sentence can be reformulated by exchanging the vocabulary without changing the syntax. For example, the sentences

[1] Test Your Vocab - How Many Words Do You Know?, http://testyourvocab.com

(i) *He is feeling sick because his back is aching.*

and

(ii) *He is being ill as his rear is hurting.*

deliver the same meaning, but the latter only uses 44% of the original vocabulary.

Second, an at least equally powerful way to reconstruct a sentence is given by the set of rules the grammar of a natural language defines. For example, a simplified English grammar could look like follows[2]:

```
sentence → nounphrase verbphrase
nounphrase → nounexpression | determiner nounexpression
nounexpression → noun | adjective nounexpression
verbphrase → verb | verb nounphrase
```

In combination with the lexical (terminal) rules

```
determiner → a | the
noun → cat | dog
verb → chases
adjective → big | brown | lazy | white
```

the sentence

(iii) *The big white cat chases a lazy brown dog.*

can be built by systematically applying the given rewriting rules:

```
sentence

    → nounphrase verbphrase
    → determiner nounexpression verbphrase
```

[2] example taken from http://www.amzi.com/AdventureInProlog/a15nlang.php, visited March 2014

→ *The* nounexpression verbphrase
→ *The* adjective nounexpression verbphrase
→ *The big* nounexpression verbphrase
→ *The big* adjective nounexpression verbphrase
→ *The big white* noun verbphrase
→ *The big white cat* verbphrase
→ *The big white cat* verb nounphrase
→ *The big white cat chases* nounphrase
→ *The big white cat chases* determiner nounexpression
→ *The big white cat chases a* nounexpression
→ *The big white cat chases a* adjective nounexpression
→ *The big white cat chases a lazy* nounexpression
→ *The big white cat chases a lazy* adjective nounexpression
→ *The big white cat chases a lazy brown* nounexpression
→ *The big white cat chases a lazy brown* noun
→ *The big white cat chases a lazy brown dog*

Being still a research topic, scientist discuss whether - and if yes, where - the grammar of natural languages can be placed in the four-level Chomsky hierarchy [31]: (Type 0) recursively enumerable, (Type 1) context sensitive, (Type 2) context free or (Type 3) regular grammars. While the authors in [99] claim that natural languages are of Type 3, recent research concludes that they can't be fitted into any of the Chomsky types [76].

2.2.2. Parse Trees

A more readable option to visualize the grammar construction of a sentence is by using grammar trees. Figure 2.2 shows the syntax tree for sentence (iii) according to the previously defined grammar. In natural language processing (NLP) applications [32] such trees are usually referred to as *(full) parse trees* or *syntax trees*, and the nodes are normally labeled with part-of-speech (POS) tags which refer to Penn Treebank tags [110]. An excerpt of important tags including examples[3] is shown in Table 2.1, whereas the complete list of Penn Treebank tags can be seen in the Appendix in Section A.1.

Obviously a natural language like English consists of much more grammar rules than presented in the example earlier, which are recognized by modern parsers. Thus, by utilizing a state-of-the-art parser like the Stanford Parser [90], the correct parse tree using Penn Treebank tags is shown in Figure 2.3.

[3]examples taken from http://www.clips.ua.ac.be/pages/mbsp-tags, visited March 2014

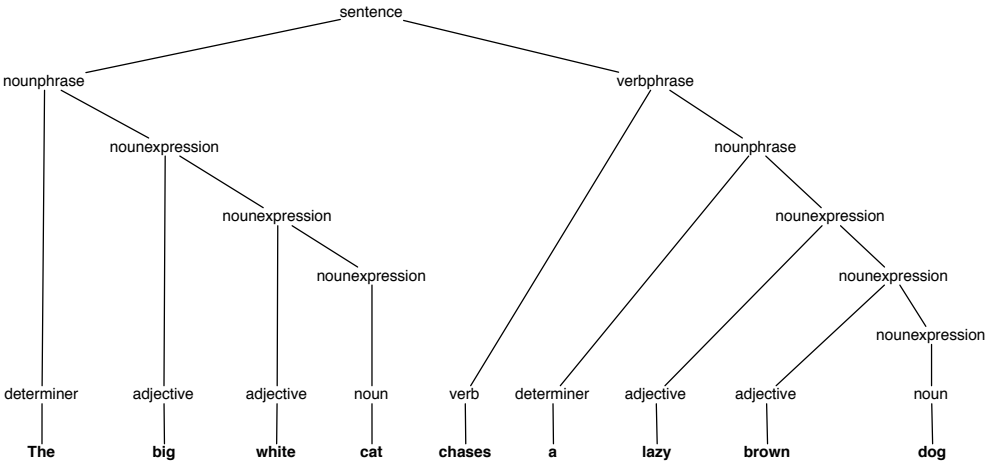


Figure 2.2.: Grammar Tree of Sentence (iii).

Tag	**Description**	**Example**
CC	conjunction, coordinating	and, or, but
DT	determiner	the, a, these
IN	conjunction, subordinating or preposition	of, on, before, unless
JJ	adjective	nice, easy
JJS	adjective, superlative	nicest, easiest
NP	noun phrase	the strange bird
NN	noun, singular or mass	tiger, chair, laughter
PRP	pronoun, personal	me, you, it
RB	adverb	extremely, loudly, hard
RP	adverb, particle	about, off, up
VB	verb, base form	think
VBZ	verb, 3rd person singular present	she thinks
VP	verb phrase	was looking
WP	wh-pronoun, personal	what, who, whom

Table 2.1.: Excerpt of Penn Treebank Tags.

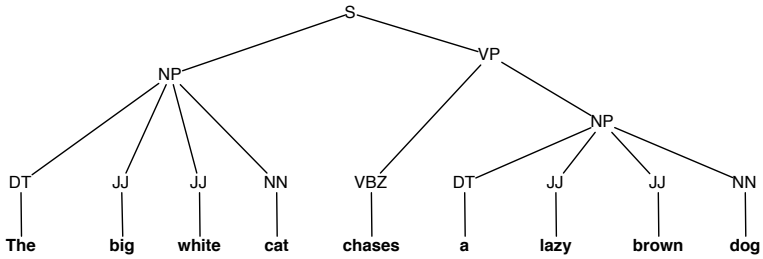


Figure 2.3.: Correct Parse Tree of Sentence (iii) using POS tags.

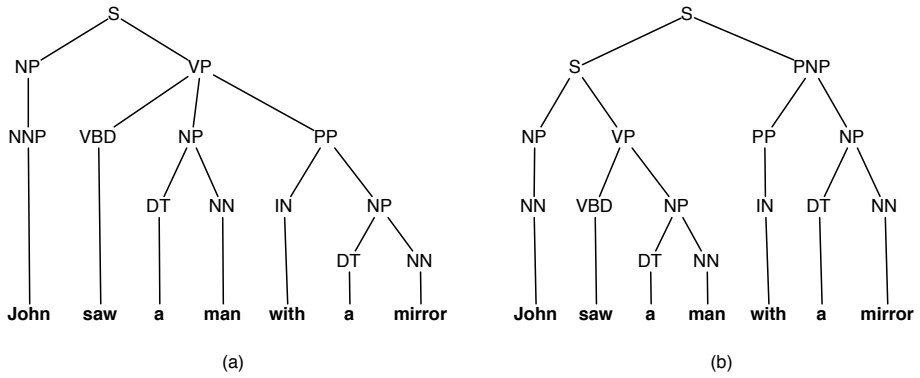


Figure 2.4.: Ambiguous Parse Trees for Sentence (iv).

2.2.3. Ambiguity

In the case of the example stated earlier, the parse tree for sentence (iii) is distinct as there is only one possible derivation. On the other hand there exist many sentences which have more than one correct parse trees. In particular this occurs when a sentence has an ambiguous meaning. For example, the sentence

(iv) *John saw a man with a mirror.*

can be read in two ways: (a) John saw a man, and the man holds/has a mirror. (b) John saw a man, and he saw him by using a mirror. Accordingly, the two possible parse trees are shown in Figure 2.4. Another interesting example has been found in [162], where a morpho-syntactical analysis of written old Hebrew revealed that most of the sentences in the Old Testament have various possible parse trees, which makes the interpretation very interesting for linguists and theologists.

Basically, ambiguities can be differentiated into global and local ambiguities [123], respectively, where "global ambiguity impacts the whole sentence", and "local ambiguity is limited to one or more pieces of a sentence. In case of the example shown in Figure 2.4, where multiple parse trees for a sentence exist, the type is called *structural ambiguity*, i.e. where different interpretations of a sentence can be made by varying the syntax. Additionally, a *word sense ambiguity* occurs when one or more of the terminal nodes of a parse tree, i.e. words, can be understood in different ways. An example for this type would be the word *cards* in the sentence "*She has cards in her pocket*", which could be seen like "credit cards" or "playing cards" [123].

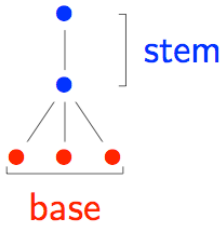


Figure 2.5.: Structure of a pq-gram Consisting of Stem $p = 2$ and Base $q = 3$.

Although ambiguities are important to consider, especially for linguists, the work described in this thesis is neglecting multiple parse trees for a single sentence. Instead, the most probable grammar tree that is estimated by the parser is chosen as a representative structure.

2.3. Preliminaries: pq-grams

2.3.1. The pq-gram index

Similar to n-grams which represent subparts of given length n of a string, Augsten et al. proposed pq-grams which extract substructures of an ordered, labeled tree [12, 69]. The size of a pq-gram is determined by a stem (p) and a base (q) like it is shown in Figure 2.5. Thereby p defines how much nodes are included vertically, and q defines the number of nodes to be considered horizontally.

For example, a valid pq-gram with $p = 2$ and $q = 3$ starting from the root of tree T_a illustrated in Figure 2.6 would be the subtree `(A)-(B)-(D,E,F)`, which can be serialized as `[A-B-D-E-F]`.

The pq-gram index then consists of all possible pq-grams[4] of a tree. In order to obtain all pq-grams, the base is shifted left and right additionally: If then less than p nodes exist horizontally, the corresponding place in the pq-gram is filled with `*`, indicating a missing node. Applying this idea to tree T_a, also the following pq-grams - resulting from horizontal shifts - have to be considered:

[4]For simplicity reasons, the term 'pq-gram' denotes the serialization of a pq-gram in the following.

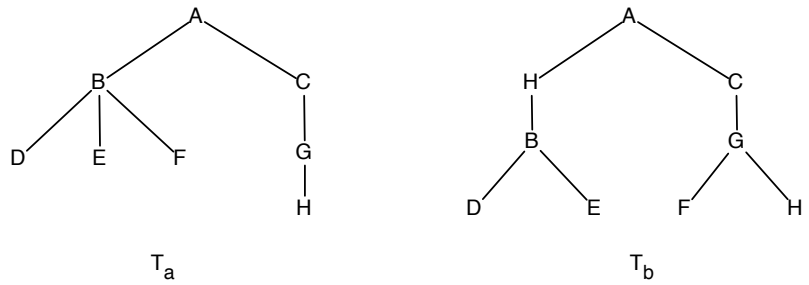


Figure 2.6.: Two Examples of Ordered, Labeled Trees.

- `A-B-*-*-D` (base shifted left by two)
- `A-B-*-D-E` (base shifted left by one)
- `A-B-E-F-*` (base shifted right by one)
- `A-B-F-*-*` (base shifted right by two)

Additionally, if the height of a node is less than p+1, i.e. a node has no children to perform horizontal shifts on, the corresponding missing children are also treated as regular missing nodes, resulting in q gap nodes (`*`). Consequently, all leaves have the pq-gram pattern `[`*parent-leaf*`-*-*-*-*]`. Finally, also an imaginary node connected to the root has to be calculated. Thus, also the pq-gram `[*-A-*-B-C]` is valid.

The pq-gram index is then a bag of all valid pq-grams of a tree[5], which includes all possible valid extractions for each starting node. Because the index is a bag, all multiple occurrences of the same pq-grams are also present multiple times in the index. The size of a pq-gram index is $\mathcal{O}(n)$ for a tree with n nodes [12].

As an example, the complete pq-gram index $\mathcal{I}$ of tree T_a using $p = 2$ and $q = 3$ is as follows:

$\mathcal{I}(T_a) = \{$

```
*A**B, *A*BC, *ABC*, *AC**,          (root node is *)
AB**D, AB*DE, ABDEF, ABEF*, ABF**,   (root node is A)
AC**G, AC*G*, ACG**,                 (root node is A)
BD***,BE***,BF***,                   (root node is B)
```

[5]Originally the serialization of each pq-gram is hashed and not stored as string label

`CG**H, CG*H*, CGH**,` *(root node is* `C`*)*
`GH***` *(root node is* `G`*)*

}

2.3.2. The pq-gram distance

An often used concept to compare trees is the tree edit distance [20], which calculates the minimum cost to transform a tree into another, different tree. Thereby the following edit operations are allowed: (1.) insertion (2.) deletion and (3.) renaming. For each operation a cost has to be defined, and the calculation is either based on a *unit cost model* (no differences between leaves and non-leaves) or a *fanout model* (changes on leaves have small costs, but non-leaf changes cost proportional to the node fanout). A major disadvantage of the tree edit distance is that its computation is very costly: its complexities are $\mathcal{O}(n^3)$ for runtime and $\mathcal{O}(n^2)$ for needed space, respectively [20]. Additionally an appropriate cost model has to be found in order to be sensitive to structure changes.

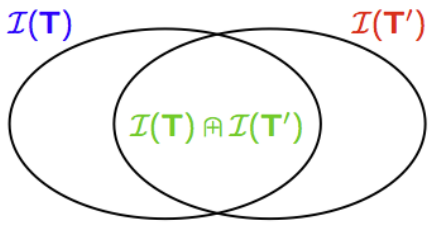


Figure 2.7.: Visualization of the Components of the pq-gram Distance[6].

As a distance between trees is needed massively for algorithms described in this thesis, the more efficient pq-gram distance is used, which has a runtime complexity of $\mathcal{O}(n \cdot log(n))$ [12]. Moreover, it is implicitly sensitive to structure changes, which is important for the algorithms. More precisely, the pq-gram distance is a lower bound of the fanout weighted tree edit distance and is formally defined as

$$dist^{pq}(T_1, T_2) = |\mathcal{I}(T_1) \uplus \mathcal{I}(T_2)| - 2 \cdot |\mathcal{I}(T_1) \uplus \mathcal{I}(T_2)|$$

whereby $\uplus$ and $\uplus$ correspond to the union and intersection of bags (multi-sets), i.e., multiple occurrences of an item are also added or subtracted multiple

[6] reused from presentation slides of a talk about pq-grams by Prof. Augsten in Innsbruck, Austria, 2.5.2011

times, respectively. A visualization of the components of the distance is shown in Figure 2.7. As an example, the pq-gram distance between trees T_a and T_b can be calculated as follows:

$$\mathcal{I}(T_a) = \{\texttt{*A**B}, \texttt{*A*BC}, \texttt{*ABC*}, \texttt{*AC**}, \texttt{AB**D}, \texttt{AB*DE}, \texttt{ABDEF}, \texttt{ABEF*}, \texttt{ABF**}, \texttt{AC**G}, \texttt{AC*G*}, \texttt{ACG**}, \texttt{BD***}, \texttt{BE***}, \texttt{BF***}, \texttt{CG**H}, \texttt{CG*H*}, \texttt{CGH**}, \texttt{GH***}\}$$

$$\mathcal{I}(T_b) = \{\texttt{*A**H}, \texttt{*A*HC}, \texttt{*AHC*}, \texttt{*AC**}, \texttt{AH**B}, \texttt{AH*B*}, \texttt{AHB**}, \texttt{AC**G}, \texttt{AC*G*}, \texttt{ACG**}, \texttt{HB**D}, \texttt{HB*DE}, \texttt{HBDE*}, \texttt{HBE**}, \texttt{CG**F}, \texttt{CG*FH}, \texttt{CGFH*}, \texttt{CGH**}, \texttt{BD***}, \texttt{BE***}, \texttt{GF***}, \texttt{GH***}\}$$

$\lvert\mathcal{I}(T_a)\rvert = 19$	number of pq-grams in $\mathcal{I}(T_a)$
$\lvert\mathcal{I}(T_b)\rvert = 22$	number of pq-grams in $\mathcal{I}(T_b)$
$\lvert\mathcal{I}(T_a) \uplus \mathcal{I}(T_b)\rvert = 19 + 22$	multi-set union of the two indices
$\lvert\mathcal{I}(T_a) \upharpoonright \mathcal{I}(T_b)\rvert = 7$	multi-set intersection of the two indices (the number of pq-grams occurring in both indices)

Finally, the pq-gram distance between the trees T_a and T_b is:

$$dist^{pq}(T_a, T_b) = \lvert\mathcal{I}(T_a) \uplus \mathcal{I}(T_b)\rvert - 2 \cdot \lvert\mathcal{I}(T_a) \upharpoonright \mathcal{I}(T_b)\rvert = 19 + 22 - 2 \cdot 7 = 27$$

2.4. The Plag-Inn Algorithm[7]

2.4.1. Algorithm

The *Plag-Inn algorithm*[8] represents the basis for all plagiarism detection algorithms presented in this thesis. Being an intrinsic detection approach, it attempts to expose plagiarism within a text document based on stylistic changes. Based on the assumption that different authors use different grammar rules to build their sentences, it compares the grammar of each sentence against the grammar of each other sentence and tries to find suspicious ones.

For example, the sentence[9]

(S_1) *The strongest rain ever recorded in India shut down the financial hub of Mumbai, officials said today.*

could also be formulated as

(S_2) *Today, officials said that the strongest Indian rain which was ever recorded forced Mumbai's financial hub to shut down.*

which is semantically equivalent but differs significantly according to its syntax. The grammar trees produced by the two sentences are shown in Figure 2.8. It can be seen that there is a significant difference in the building structure of each sentence. The main idea of the approach is to quantify those differences and to find outstanding sentences or paragraphs which are assumed to have a different author and thus may be plagiarized. In order to analyze a sentence, pq-grams and pq-gram distances are used.

Given a text document to analyze, the "suspicious" document, the Plag-Inn algorithm consists of five basic steps:

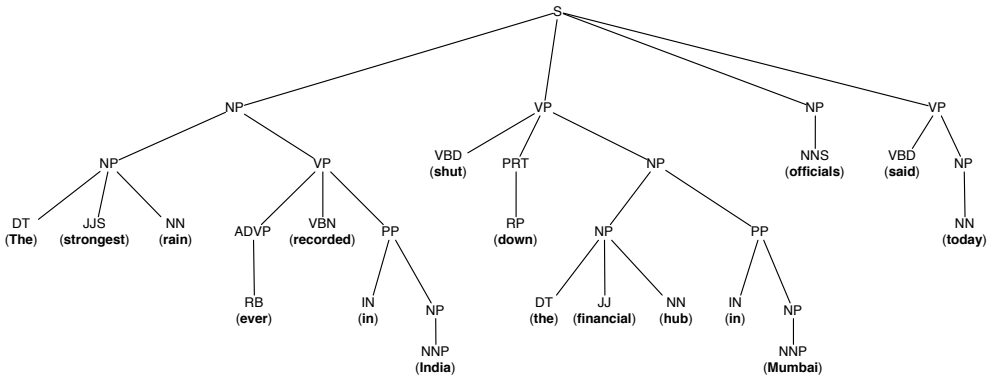
1. Split the document into single sentences.

2. Compute full parse trees for each sentence.

[7] This section is based on and contentual partly reused from the paper: M. Tschuggnall and G. Specht. *Detecting Plagiarism in Text Documents through Grammar-Analysis of Authors.* In Proceedings of the 15. GI-Fachtagung Datenbanksysteme für Business, Technologie und Weg (BTW), Magdeburg, Germany, March 2013, volume 214 of LNI, pages 241–259. [187]

[8] Plag-Inn stands for *In*trinsic *Plag*iarism Detection *Inn*sbruck

[9] example taken and modified from the Stanford Parser website [181]



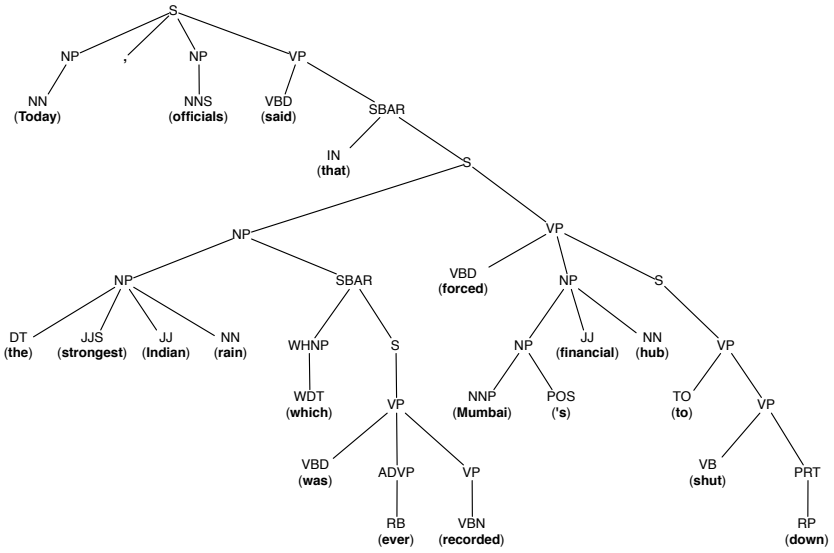


Figure 2.8.: Grammar Trees Resulting From Sentence (S_1) and (S_2).

3. Calculate the pq-gram distance between every distinct pair of sentences and store the result into a distance matrix.

4. Fit the distances into a Gaussian normal distribution and mark significantly outstanding sentences as plagiarized.

5. Refine the final prediction by grouping/ungrouping text passages and selecting/deselecting individual sentences.

In the following each step is explained in detail.

Splitting the text into single sentences

At first the document is preprocessed by eliminating unnecessary whitespaces or non-parsable characters. For example, many data sets often are based on novels and articles of various authors, whereby frequently OCR text recognition is used due to the lack of digital data. Additionally, such documents contain problem sources like chapter numbers and titles or incorrectly parsed picture frames that result in non-alphanumeric characters.

The cleaning step is not crucial to the algorithm, but supports the subsequent task of splitting the document into single sentences, which is done using Sentence Boundary Detection (SBD) algorithms [175]. The simple sounding task of recognizing sentence ends has been a research problem for many years, as the intuitive splitting by fullstops is not sufficient, like the following example[10] demonstrates:

"Prof. Dr. Pierre Vinken, a 61 year old U.S. citizen, will join the board as a nonexecutive director on Nov. 29. Mr. Vinken is chairman of Elsevier N.V., the Dutch publishing group."

Many approaches have achieved accuracies of over 98% already before the 20th century, e.g. [151], whereby recent algorithms even report an error rate of less than 0.25%, e.g. [56]. As the correct splitting of sentences is essential to all approaches throughout this thesis, state-of-the-art SBD algorithms have to be utilized. Currently, the open source tool *OpenNLP*[11] is used to meet this requirement.

[10] example taken and modified from http://blog.dpdearing.com/2011/05/opennlp-1-5-0-basics-sentence-detection-and-tokenizing/, visited March 2014

[11] Apache OpenNLP, http://incubator.apache.org/opennlp, visited March 2014

Computing full parse trees

Using the *Stanford Parser* [90], a syntax tree is computed for each sentence. Like it is described in Section 2.2, each node of the tree is labelled with a Penn Treebank tag, which for example correspond to word-level classifiers like nouns (NN) or phrase-level classifiers like verb phrases (VP). Since the Plag-Inn algorithm investigates only the grammatical structure, the actual words (vocabulary) of a sentence are irrelevant. Consequently, the terminals of each tree, i.e., the words, are dismissed.

Calculating pq-gram distances

Having a grammar tree for all n sentences of the text document, the difference between every distinct pair of sentences is stored into a triangular distance matrix D_n:

$$D_n = \begin{pmatrix} d_{1,1} & d_{1,2} & d_{1,3} & \cdots & d_{1,n} \\ d_{1,2} & d_{2,2} & d_{2,3} & \cdots & d_{2,n} \\ d_{1,3} & d_{2,3} & d_{3,3} & \cdots & d_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{1,n} & d_{2,n} & d_{3,n} & \cdots & d_{n,n} \end{pmatrix} = \begin{pmatrix} 0 & d_{1,2} & d_{1,3} & \cdots & d_{1,n} \\ * & 0 & d_{2,3} & \cdots & d_{2,n} \\ * & * & 0 & \cdots & d_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ * & * & * & \cdots & 0 \end{pmatrix}$$

Each $d_{i,j}$ entry corresponds to the distance of the grammar trees between sentence i and j, whereby $d_{i,j} = d_{j,i}$. The distance itself is calculated by computing the pq-gram distance of the respective pq-gram indices of the sentences. As the distance between sentence i and j is the same as the distance between sentence j and i, the resulting distance matrix is triangular. Hence, filling D_n requires only $\binom{n}{2} = \frac{n(n-1)}{2}$ distance calculations, where n corresponds to the total number of sentences of the document. Nevertheless, by performing experiments it could be observed that the calculation and storage of pq-gram distances make up only a small proportion of the global execution time, and that the major effort is needed of the syntactic parsing of sentences.

The assumption followed by this approach is that individual (groups of) sentences have significantly higher distances to all other sentences in the document, and that such sentences can be exposed to be plagiarized. As an example, the matrix

$$D_6 = \begin{pmatrix} 0 & 4 & \mathbf{17} & 6 & 3 & 4 \\ 4 & 0 & \mathbf{21} & 5 & 7 & 2 \\ \mathbf{17} & \mathbf{21} & \mathbf{0} & \mathbf{14} & \mathbf{15} & \mathbf{16} \\ 6 & 5 & \mathbf{14} & 0 & 4 & 5 \\ 4 & 7 & \mathbf{15} & 4 & 0 & 6 \\ 4 & 2 & \mathbf{16} & 5 & 6 & 0 \end{pmatrix}$$

indicates that sentence 3 may have been plagiarized as its distances are significantly higher. A visualization of a distance matrix of a document consisting of 1500 sentences (D_{1500}) and containing sentences with significantly higher distances is depicted in Figure 2.9, whereby the triangular character of the matrix is ignored in this case for better visibility. The z-axis represents the pq-gram distance between the sentences on the x- and y-axis, and it can be seen that there are significant differences in the style of sentences around number 100 and 800, respectively. In contrast, a 3D plot of a smaller, 200 sentences long document which contains no suspicious sections is illustrated in Figure 2.10.

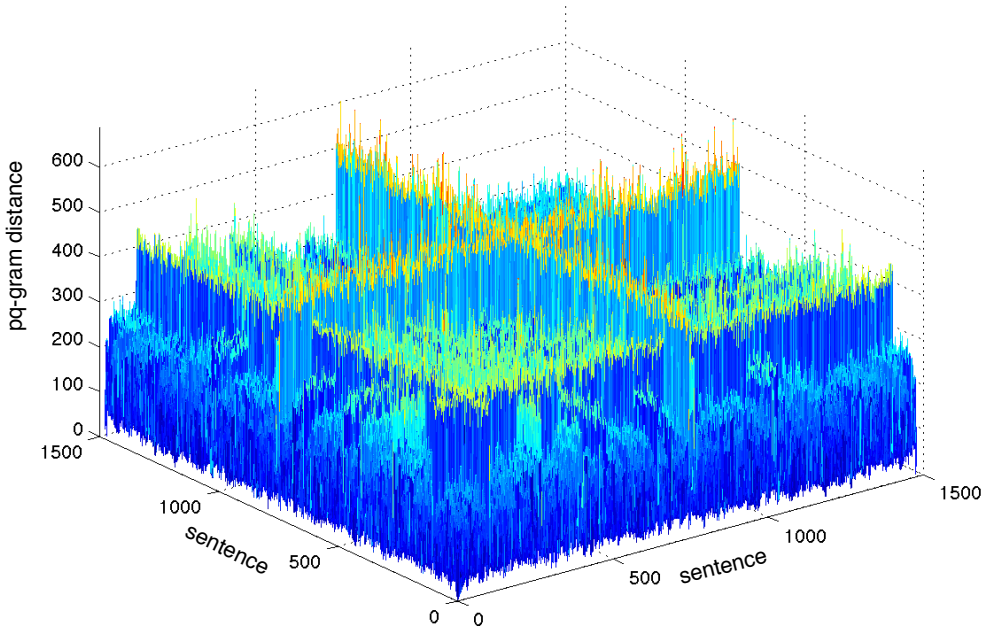


Figure 2.9.: Plag-Inn: Distance Matrix of a Large Document With 1500 Sentences Containing Suspicious Sections.

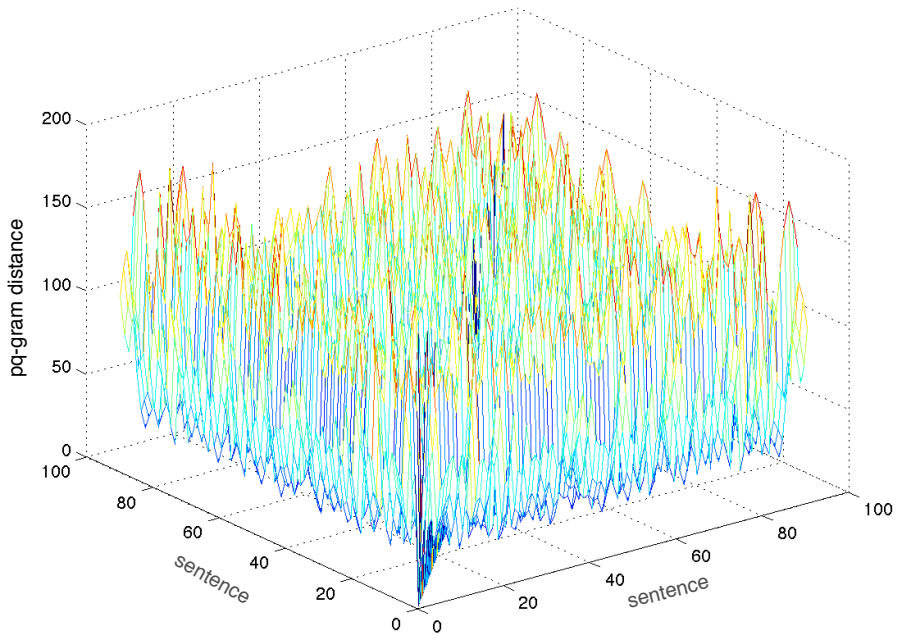


Figure 2.10.: Plag-Inn: Distance Matrix of a Short Document With 100 Sentences Containing no Suspicious Sections.

Calculating average distances, Gaussian normal distribution and suspicious sentences

Significant differences which are already visible to a human eye in the distance matrix plot are now examined through statistical methods. To find significantly outstanding sentences, i.e., sentences that might have been plagiarized, the median distance for each row in D_n is calculated. The resulting vector

$$\bar{d} = (\bar{d}_1, \bar{d}_2, \bar{d}_3, \cdots, \bar{d}_n)$$

is then fitted to a Gaussian normal distribution, which estimates the mean value μ and the standard deviation σ. The two Gaussian values can thereby be interpreted as a common variation of how the author of the document builds his sentences grammatically.

Finally, all sentences that have a higher distance than a predefined threshold δ_{susp} are marked as suspicious. The definition and optimization of δ_{susp} (where $\delta_{susp} \gg \mu+\sigma$) is shown in Section 2.4.3. Figure 2.11 depicts the mean distances resulting from averaging the distances for each sentence in the distance matrix

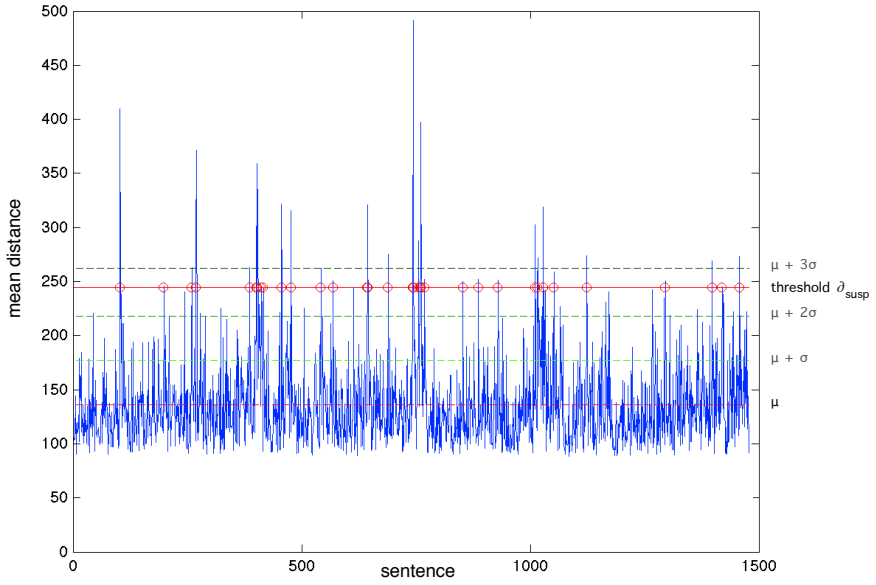


Figure 2.11.: Plag-Inn: Average Distances Including the Gaussian-Fit Values μ and σ.

of the example shown in Figure 2.9. After fitting the data to a Gaussian normal distribution, the resulting mean μ and standard deviation σ are marked in the plot. The threshold δ_{susp} that splits ordinary from suspicious sentences can also be seen, and all sentences exceeding this threshold are marked.

Making the final prediction

The last step of the algorithm is to smooth the results coming from the mean distances and the Gaussian fit algorithm. At first, suspicious sentences that are close together with respect to their occurrence in the document are grouped into paragraphs. Secondly, standalone suspicious sentences might be dropped because personal experiences showed that it is unlikely in many cases that just one sentence has been plagiarized. The algorithm incorporating these ideas is explained in detail in the following section.

2.4.2. Sentence Selection Algorithm

The result of steps 1-4 of the Plag-Inn algorithm is a vector $\bar{d}$ of size n which holds the average pq-gram distances of each sentence to all other sentences. After fitting this vector to a Gaussian normal distribution, all sentences having

a higher average distance than a predefined threshold δ_{susp} are marked as suspicious. This step is already the first step as can be seen in Algorithm 1.

The main objective of the further procedure of the sentence-selection algorithm is to group together sentences into plagiarized paragraphs and to eliminate standalone suspicious sentences. As the Plag-Inn algorithm is based on the grammatical structure of sentences, short instances like *"I like tennis."* or *"At what time?"* carry too less (grammar) information and are most often not marked as suspicious as their building structure is too simple. Nevertheless, such sentences may be part of a plagiarized section and should therefore be detected. For example, if eight sentences in a row have found to be suspicious except one in the middle, it is intuitively very likely that it should be marked as suspicious as well.

To group together sentences, the procedure shown in Algorithm 1 traverses all sentences in sequential order. If it finds a sentence that is marked as suspicious, it first creates a new plagiarized section and adds this sentence. As long as ongoing suspicious sentences are found they are added to this section. When a sentence is not suspicious, the global idea is to use a lookahead variable ($curLookahead$) to step over non-suspicious sentences and to check if there is a suspicious sentence within the lookahead-range. If a sentence is then found to be suspicious within a predefined maximum ($maxLookahead$), this sentence and all non-suspicious sentences in between are added to the plagiarized section, and the lookahead variable is reset. Otherwise if this maximum is exceeded and no suspicious sentences are found, the current section is closed and added to the final result.

After all sentences are traversed, plagiarized sections in the final set R that contain only one sentence are checked to be filtered out. Intuitively this step makes sense as it can be assumed that authors do not copy only one sentence in a large paragraph, but most likely more than one. Within the evaluation of the algorithm described in Section 2.4.4 it could additionally be observed that single-sentence sections of plagiarism are often the result of wrongly parsed sentences coming from noisy data. To ensure that these sentences are filtered out, but *strongly plagiarized* single sentences remain in the result set, another threshold is introduced. Accordingly, δ_{single} defines the average distance threshold that has to be exceeded by sections that contain only one sentence in order to remain in the result set R.

As the optimization of parameters (Section 2.4.3) showed, the best results can be achieved when choosing $\delta_{single} > \delta_{susp}$, which strengthens the intuitive assumption that a single-sentence section has to be *really* different. On the other hand, genetic optimization algorithms also generated parameter config-

Algorithm 1 Sentence Selection Algorithm

input:

d_i	mean distance of sentence i
δ_{susp}	suspicious sentence threshold
δ_{single}	single suspicious sentence threshold
$maxLookahead$	maximum lookahead for combining non-susp. sent.
$filterSingles$	indicates whether single susp. sent. should be filtered

variables:

$susp_i$	indicates whether sentence i is suspicious or not
R	final set of suspicious sections
S	set of sentences belonging to a suspicious section
T	temporary set of sentences
$curLookahead$	used lookaheads

set $susp_i \leftarrow false$ for all i, $R \leftarrow \varnothing, S \leftarrow \varnothing, T \leftarrow \varnothing$, $curLookahead \leftarrow 0$
for i from 1 to n **do**
    if $d_i > \delta_{susp}$ **then** $susp_i \leftarrow true$
end for
for i from 1 to number of sentences **do** ▷ traverse sentences
    if $susp_i = true$ **then**
        if $T \neq \varnothing$ **then**
            $S \leftarrow S \cup T$ ▷ add all non-suspicious sentences in between
            $T \leftarrow \varnothing$
        end if
        $S \leftarrow S \cup \{i\}$, $curLookahead \leftarrow 0$
    else if $S \neq \varnothing$ **then**
        $curLookahead \leftarrow curLookahead + 1$
        if $curLookahead \leq maxLookahead$ **then**
            $T \leftarrow T \cup \{i\}$ ▷ add non-susp. sentence i to temporary set T
        else
            $R \leftarrow R \cup \{S\}$ ▷ finish section S and add it to the final set R
            $S \leftarrow \varnothing, T \leftarrow \varnothing$
        end if
    end if
end for
if $filterSingles = true$ **then** ▷ filter single-sentence sections
    for all plagiarized sections S of R **do**
        if $|S| = 1$ **then**
            $i \leftarrow$ the (only) element of S
            if $d_i < \delta_{single}$ **then** $R \leftarrow R \setminus \{S\}$
        end if
    end for
end if

urations that recommend to not filter single-sentence sections at all, but leave them in the final prediction set of plagiarized sentences.

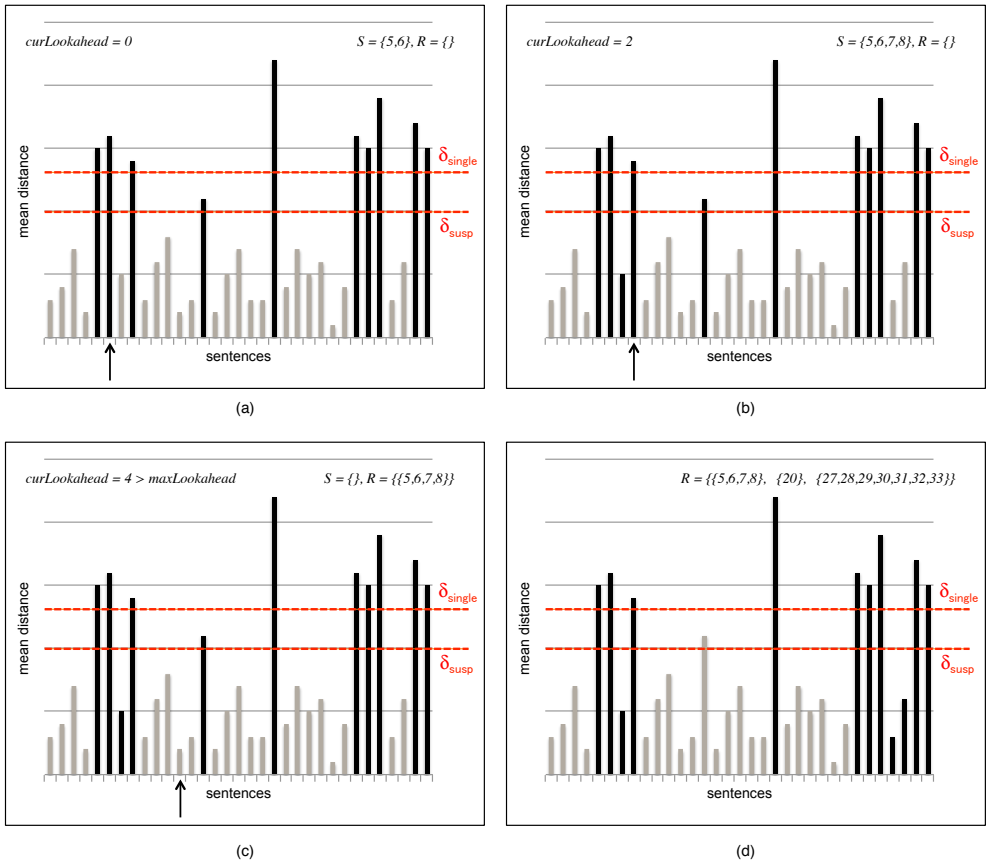


Figure 2.12.: Example of the Sentence-Selection Algorithm.

An example of how the algorithm (using $maxLookhead = 3$) works can be seen in Figure 2.12. Diagram (a) shows all sentences, where all instances with a higher mean distance than δ_{susp} have been previously marked as suspicious. When reaching suspicious sentence 5, it is added to a newly created section S. After adding sentence 6 to S, the lookahead variable is incremented as sentence 7 is not suspicious. Reaching sentence 8 which is suspicious again, both sentences are added to S as can be seen in Diagram (b). This procedure continues until the maximum lookahead is reached, which can be seen in Diagram (c). Because the next sentence is not suspicious (see marker), S is closed and added to the result set R. Finally, Diagram (d) shows the final result set after eliminating single-sentence sections. As can be seen, sentence 14 has been filtered out as its mean difference is less than δ_{single}, whereas sentence 20 remains in the final result R.

A comprehensive example showing every step of the algorithm can be seen in Section A.3 of the appendix.

2.4.3. Optimization

Test Set

To evaluate and optimize the Plag-Inn algorithm including the sentence-selection algorithm, the PAN 2010 test corpus (PAN-PC-10), [147] has been used. It originally contains more than 27,000 English documents, and thereof approximately 4,700 documents which are specifically targeted for intrinsic plagiarism detection. The documents consist of a various number of sentences, starting from short texts from e.g. fifty sentences up to novel-length documents of about 7,000 sentences. About 50% of the documents contain plagiarism, varying the amount of plagiarized sections per document, while the other 50% are left originally and contain no plagiarized paragraphs.

Most of the plagiarism cases are built by copying text fragments from other documents and subsequently inserting them in the suspicious document, while manual obfuscation of the inserted text is done additionally in some cases. Also, some plagiarism cases have been built by copying and subsequently translating from other source-languages like Spanish or German. Finally, for every document there exists a corresponding annotation file which can be consulted for an extensive evaluation.

Detailed statistics about the PAN-PC-10 corpus are presented in Table 2.2.

Plagiarism per Document		
hardly	(5%-20%)	45%
medium	(20%-50%)	15%
much	(50%-80%)	25%
entirely	(> 80%)	15%

Document Length		
short	(1-10 pp.)	50%
medium	(10-100 pp.)	35%
long	(100-1000 pp.)	15%

(a) Document Statistics

Obfuscation	
none	40%
artificial	
- low obfuscation	20%
- low obfuscation	20%
simulated	6%
translated({de,es} to en)	14%

Case Length		
short	(50-150 words)	34%
medium	(300-500 words)	33%
long	(3000-5000 words)	33%

(b) Plagiarism Case Statistics

Table 2.2.: Statistics of the PAN-PC-10 Corpus.

As depicted in the previous section, the sentence-selection algorithm relies on various input variables:

- δ'_{susp}: suspicious sentence threshold. Every sentence that has a higher mean distance is marked as suspicious.

- δ'_{single}: single suspicious sentence threshold. Every sentence in a single-sentence plagiarized section that is below this threshold is unmarked in the final step of the algorithm.

- $maxLookahead$: maximum lookahead. Defines the maximum value of checking if there is a suspicious sentence occurring after non-suspicious sentences that can be included into the current plagiarized section.

- $filterSingles$: boolean switch that indicates whether sections containing only one sentence should be filtered out. If $filterSingles = true$, the single suspicious sentence threshold δ_{single} is used to determine whether a section should be dropped or not.

Thereby, the values for the thresholds δ'_{susp} and δ'_{single}, respectively, represent the inverse probability range of the Gaussian curve that include sentences with a mean distance that is not marked as suspicious. For example, $\delta'_{susp} = 0.9973$ would imply that $\delta_{susp} = \mu + 3\sigma$, meaning that all sentences having a higher average distance than $\mu + 3\sigma$ are marked as suspicious. In other words, one would find 99.73% of the values of the Gaussian normal distribution within a range of $\mu \pm 3\sigma$. In Figure 2.11, δ_{susp} resides between $\mu + 2\sigma$ and $\mu + 3\sigma$.

In the following the optimization techniques are described which should help finding the parameter configuration that produces the best result. To achieve this, predefined configurations have been tested as well as genetic algorithms have been utilized. Additionally, the latter have been applied to find optimal configurations on two distinct document subsets that have been split by the number of sentences (see Section 2.4.3).

All configurations have been evaluated using the common IR-measures *recall*, *precision* and the resulting harmonic mean *F-measure*. In this case the recall value represents the percentage of plagiarized sections found, and the precision value corresponds to the percentage of correct matches, respectively. In order to compare this approach to others, the algorithm defined by the PAN workshop [147] has been used to calculate the according values[12].

[12]Note that the PAN algorithms of calculating recall and precision, respectively, are based on plagiarized sections rather than plagiarized characters, meaning that if an algorithm detects 100% of a long section but fails to detect a second short section, the F-measure can never exceed 50%. Calculating the F-measure character-based throughout the Plag-Inn evaluation resulted in an increase of about 5% in all cases.

Predefined Configurations

As a first attempt, 450 predefined configurations have been created by building all[13] permutations of the values in Table 2.3.

Parameter	Range
δ'_{susp}	$[0.994, 0.995, \ldots, 0.999]$
δ'_{single}	$[0.9980, 0.9985, \ldots, 0.9995]$
$maxLookahead$	$[2, 3, \ldots, 16]$
$filterSingles$	[`yes`, `no`]

Table 2.3.: Configuration Ranges for Predefined Parameter Optimization.

The five best evaluation results using the predefined configurations are shown in Table 2.4. It can be seen that all of the configurations make use of the single-sentence filtering, using almost the same threshold of δ'_{single} = 0.9995. Surprisingly, the maximum lookahead with values from 13 to 16 are quite high. Transformed to the problem definition and the sentence-selection algorithm, this means that the best results are achieved when sentences can be grouped together in a plagiarized section while stepping over up to 16 non-suspicious sentences.

As it is shown in the following section, genetic algorithms produced a better parameter configuration using a much lower maximum lookahead.

δ'_{susp}	$maxLook.$	$filterS.$	δ'_{single}	**Recall**	**Precision**	**F**
0.995	16	yes	0.9995	0.159	0.150	**0.155**
0.994	16	yes	0.9995	0.161	0.148	**0.155**
0.996	16	yes	0.9995	0.154	0.151	**0.153**
0.997	15	yes	0.9995	0.150	0.152	**0.152**
0.995	13	yes	0.9990	0.147	0,147	**0.147**

Table 2.4.: Best Evaluation Results using Predefined Configuration Parameters.

Genetic Optimization Algorithms

Since the evaluation of the documents of the PAN corpus is computationally intensive, a better way than just evaluating fixed configurations is to use genetic algorithms [58] to find optimal parameter assignments.

[13] In configurations where single-sentence sections are not filtered, i.e. $filterSingles = no$, permutations originating from the values of δ'_{single} have been ignored.

Genetic optimization algorithms emulate biological evolutions, implementing the principle of the *"Survival of the fittest"*[14]. In this sense genetic algorithms are based on *chromosomes* which consist of several *genes*. A gene can thereby be seen as a parameter, whereas a chromosome represents a set of genes, i.e., the parameter configuration. Basically, a genetic algorithm consists of the following steps:

1. Create a ground-population of chromosomes of size p.

2. Randomly assign values to the genes of each chromosome.

3. Evaluate the *fitness* of each chromosome, i.e., evaluate all documents of the corpus using the parameter configuration resulting from the individual genes of the chromosome.

4. Keep the fittest 50% of the chromosomes and alter their genes, i.e., alter the parameter assignments so that the population size is p again. Thereby the algorithms recognize whether a change in any direction lead to a fitter gene and takes it into account when altering the genes [58].

5. If the predefined number of evolutions e is reached, then stop the algorithm, otherwise repeat from step 3.

With the use of genetic optimization algorithms significantly more parameter configurations could be evaluated against the test corpus. Using the JGAP-library[15], which implements genetic programming algorithms, the parameters of the sentence-selection algorithm have been optimized. As the algorithm needs a high amount of computational effort and to avoid overfitting, random subsets of 1,000 to 2,000 documents have been used to evaluate each chromosome, whereby these subsets have been randomized and renewed for each evolution. As can be seen in Table 2.5, the results outperform the best predefined configuration with an F-measure of about 23%.

What can be seen in addition is that the best configuration gained from using a population size of $p = 400$ recommends to not filter out single-sentence plagiarized sections, but to rather keep them.

Genetic Optimization Algorithms On Document Subsets

By a manual inspection of the individual results for each document of the test corpus it could be seen that in some configurations the algorithm produced

[14]The phrase was originally stated by the British philosopher Herbert Spencer.

[15]http://jgap.sourceforge.net, visited October 2012

p	δ'_{susp}	$maxLook.$	$filterS.$	δ'_{single}	**Recall**	**Precision**	**F**
400	0.999	4	no	-	0.211	0.257	**0.232**
200	0.999	13	yes	0.99998	0.213	0.209	**0.211**

Table 2.5.: Parameter Optimization Using Genetic Algorithms.

very good results on short documents, while on the other hand it produced poor results on longer, novel-length documents. Additionally when using other configurations, the F-measure results of longer documents were significantly better.

To verify the assumption that different length documents should be treated differently, the test corpus has been split by the number of sentences in a document. For example, when using 150 as splitting number, the subsets $S_{<150}$ and $S_{\geq 150}$ have been created, containing all documents that have less than 150 sentences and containing all documents that have more than or exactly 150 sentences, respectively. Then, for each of the two subsets, the optimal parameter configuration has been evaluated using genetic algorithms as it is described in Section 2.4.3. Like before, a random number of documents from 1,000 to 2,000 has been used to evaluate a chromosome. An abstract schema of the document subsets strategy using a splitting number of 150 is sketched in Figure 2.13.

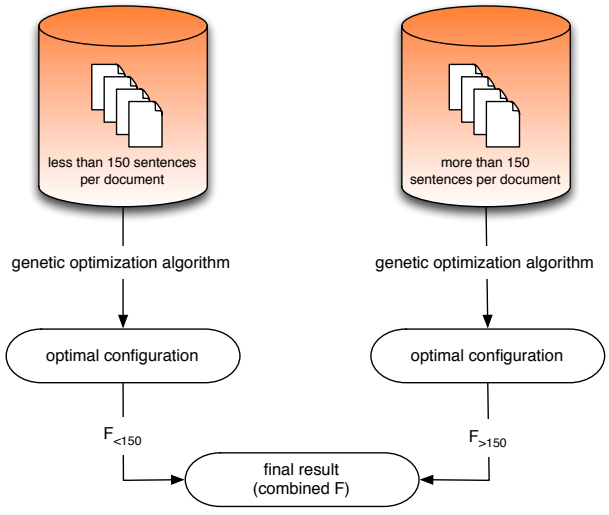


Figure 2.13.: Abstract Strategy of Optimizing Parameters for Document Subsets.

Table 2.6 shows the best configurations produced by genetic algorithms using the sentence-split document subsets. For the dividing number 100, 150 and 200 sentences per document have been chosen as this seemed to be the optimal range found by manual inspection (as discussed in Section 2.4.5 this could be improved in future work).

With an F-measure of about 50% on the short-documents subset and 21% on the long-documents subset the sentence-split evaluation worked best with a splitting number of 100 and a resulting overall F-measure of 35.7%. All configurations significantly outperform the genetic algorithm optimization described earlier, in the best case by over 12%. Moreover, what can be seen in all configurations is that the short-documents subsets could be optimized significantly better, resulting in F-values of about 45% to 50%. As discussed later, this already indicates that the algorithm is well suited for short documents.

subset	δ_{susp}	$maxLook.$	$filterS.$	δ_{single}	**Recall**	**Precision**	**F**
$S_{\geq 100}$	0.998	6	yes	0.9887	0.205	0.216	**0.210**
$S_{<100}$	0.999	1	no	-	0.501	0.508	**0.504**
					0.353	0.362	**0.357**
$S_{\geq 150}$	0.963	9	yes	0.9993	0.118	0.109	**0.113**
$S_{<150}$	0.999	4	no	-	0.494	0.478	**0.486**
					0.306	0.294	**0.300**
$S_{\geq 200}$	0.963	10	yes	0.9998	0.108	0.115	**0.111**
$S_{<200}$	0.999	2	yes	0.9999	0.441	0.457	**0.449**
					0.275	0.286	**0.280**

Table 2.6.: Parameter Optimization Using Genetic Algorithms on Document Subsets Split by the Number of Sentences.

2.4.4. Evaluation

The following section shows an extensive evaluation of the Plag-Inn approach using the sentence selection algorithm. All results are based on the optimal configuration gained from using the genetic optimization algorithms over the whole PAN-PC-10 test corpus as described in Section 2.4.3. Although the genetic algorithms optimized for the document subsets split by text length produced significant better results, it is more realistic to use just one parameter configuration in order to avoid overfitted results (manual inspection showed that small documents contained significantly less plagiarism cases in the test corpus).

Figure 2.14 shows the overall evaluation result with an F-measure of 23% As shown, using the best parameter configuration on document subsets even 35%

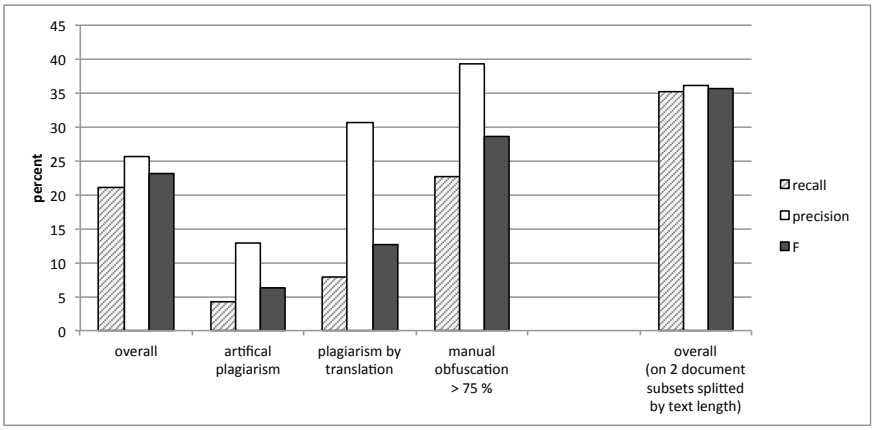


Figure 2.14.: Plag-Inn: Overall Evaluation Results.

can be achieved. It can be seen that plagiarism cases created by translation could be detected better than those created artificially[16]. This result is as expected because the grammar of another language is always different than the target language (English), and translation - which is done by automatic algorithms in most cases - produces changes in grammar that can be detected more easily.

The fourth column shows the results for documents where at least 75% of the plagiarism cases have been built by copying and subsequently doing manual obfuscation by hand. The F-measure for those cases almost reaches 30%, which indicates that the approach works very well for *real* plagiarism cases, i.e., cases that were created by humans rather than by computer algorithms. Finally, for comparison the results gained from the best parameter configurations on two document subsets split by the number of sentences contained is shown in the last column. As stated before, the best result of about 35% could be achieved by using a splitting number of 100.

In all results the precision is higher than the recall, meaning that the approach is better in correctly interpreting suspicious passages than in finding them. In other words, if the algorithm finds something, the accuracy is high.

The Plag-Inn approach achieves significantly different results on documents that contain plagiarism and on documents that do not contain plagiarism, as it is shown in Figure 2.15. Thereby, *clean* documents are processed very well and with balanced values for recall, precision and the according F-value of

[16] *artificially* means that a source fragment has been copied and modified by computer algorithms.

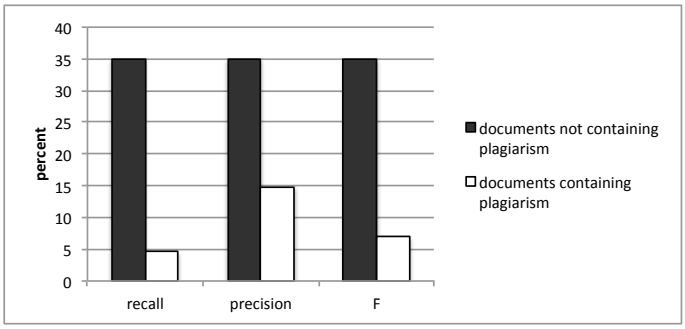


Figure 2.15.: Plag-Inn: Evaluation Results for Documents Containing and Not Containing Plagiarism Cases.

about 35%, respectively. On the other side, documents containing plagiarized sections are obviously more difficult to detect for the algorithm, while the precision is consistently higher than the recall as experienced before.

Evaluations show that the shorter documents get, the better the algorithm works. Like it is illustrated in Figure 2.16, an F-measure of over 40% could be achieved on documents containing less than 300 sentences. Moreover as stated before, the algorithm performs steadily better when documents get shorter, reaching an F-measure of nearly 70% on very short documents containing less than 50 sentences. As the algorithm uses grammatical inconsistencies to find plagiarized sections, it might be that the variety of sentence syntaxes is too high in long documents, such that the algorithm fails frequently and produces the overall result of only 23%.

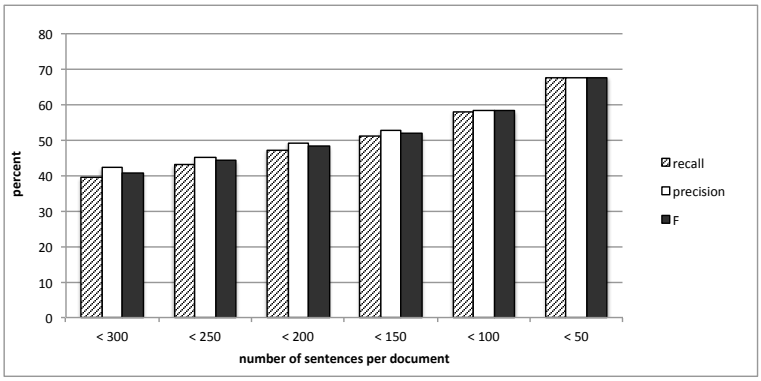


Figure 2.16.: Plag-Inn: Evaluation Results for Short Documents.

What could be found in addition is that the approach is sensitive to the number of plagiarized sections per document as it is shown in Figure 2.17. Here,

all documents that contain plagiarism have been inspected concerning the concrete number of plagiarism cases per document. It can be seen that the more sections of a document have been plagiarized, the better the results get. In other words, the more an author steals in his work, the more likely it is that it is detected by the algorithm.

Finally, the best option to improve the approach in future work is to reduce the number of false-positives, which is depicted in Figure 2.18. Diagram (a) shows the number of false-positives and false-negatives, respectively, where over 35% of all test corpus documents have wrongly been marked as plagiarized. On the horizontal axis the number of detected plagiarized sections for false-positives, and the number of not detected sections for false-negatives are shown. For about half of the wrongly predicted false-positives, the algorithm detected less than 5 plagiarism cases, and for about 10% of the documents the algorithm detected only one plagiarized section. This means that improving the algorithm in a way such that the false-positives that contain only one suspicious passage could be reduced or eliminated, this would lead to a significantly better overall result.

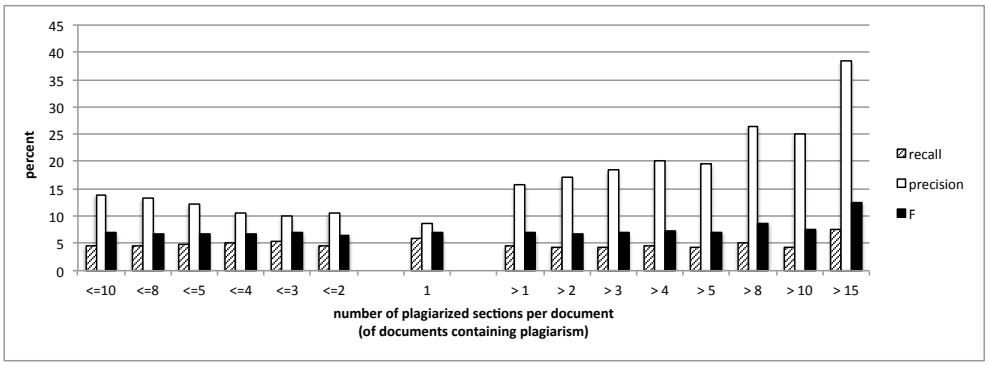
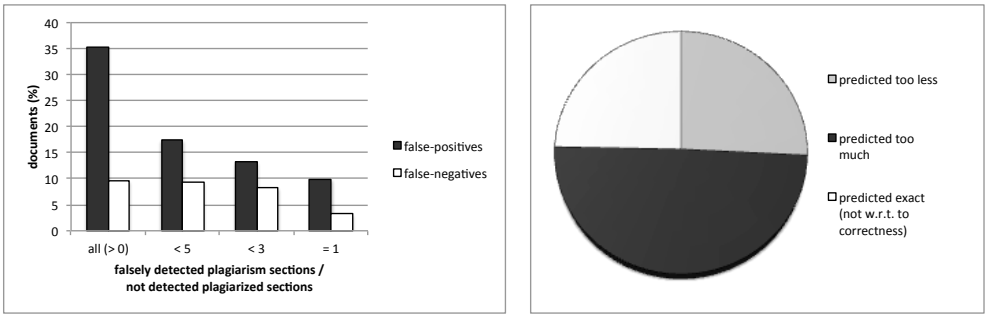


Figure 2.17.: Plag-Inn: Evaluation Results Correlated to Number of Plagiarized Sections per Document.

In diagram (b) the percentage of predictions is shown. According to the high number of false-positives, the algorithm predicted too much for about half of the documents, i.e., it detected plagiarized sections where there originally were less or even none. For the other half, too less or the exact number of sections have been detected. It has to be stressed that the amount of exact predictions in terms of plagiarized sections does not necessarily correspond to the number of *correct* detections. For example, the algorithm might have found four plagiarized sections in a document that contained exactly four plagiarized sections, but it might be the case that only two of them are correct.

Nevertheless, manual inspections of the results have shown that the majority of exact predictions really correspond to the correct sections.



(a) (b)

Figure 2.18.: Plag-Inn: False-Positives, False-Negatives and General Prediction Statistics.

2.4.5. Conclusion

In this Chapter the intrinsic plagiarism detection approach *Plag-Inn* is described: it aims to find plagiarism in text documents by inspecting the suspicious document only. The main idea is to analyze grammatical structures that authors use to build sentences by calculating pq-grams and pq-gram distances between sentences, in order to find inconsistencies in the syntax. After irregularities have been found by using Gaussian normal distribution fitting, an algorithm that selects and combines suspicious sentences is presented.

Furthermore, various parameters of the algorithm have been optimized by using predefined configurations and genetic algorithms. Using the best parameter setting, the algorithm achieves an F-measure of about 23%. By additionally using different settings for short and long documents, an overall F-measure of about 35% could be achieved, which is a rather high value for intrinsic plagiarism detection systems. In addition, an F-score of over 50% could be gained for a subset of short documents. Thereby the splitting number for the two document subsets has intuitively been chosen, and it should be considered to find the optimal value algorithmically in future work.

Extensive evaluations showed that the approach works very well on short documents containing less than 300 sentences, and that the more authors plagiarize, the more likely it is for the algorithm to detect the according sections. For documents consisting of less than 50 sentences, an F-measure of nearly 70% could be reached. Nevertheless, a drawback of the approach is that it predicts too much in many cases, i.e., it detects plagiarism where there is none, lead-

ing to a high number of false-positives. Future work should concentrate on reducing this number to improve the algorithm significantly.

On the other side, the number of false-negatives is low, implying that the approach is well-suited for ensuring that a document is not plagiarized. Evaluations showed that if the algorithm states that a document is plagiarism-free, it is right in over 90% of the cases.

2.5. The POS-PlagInn Algorithm[17]

2.5.1. Algorithm

The basic principle of the POS-Plag-Inn algorithm has been inherited from the Plag-Inn algorithm described earlier, i.e. like the original approach it attempts to detect irregularities in the grammar syntax of sentences in order to expose possible plagiarism in text documents. The major difference is the analysis of the grammar, which is made by utilizing POS tags only rather than inspecting full parse trees. Also, the comparison of sentences is made by using dynamic programming algorithms instead of pq-gram distances.

Similarly to the original approach, the POS-Plag-Inn algorithm consists of the following steps:

1. Cleaning the document and splitting it into single sentences.

2. Calculating POS tags for each extracted sentence and computing POS tag sequences reflecting the building structure.

3. Comparing each POS tag sequence with each other sequence and storing the difference into a distance matrix.

4. Calculating the average distance for each sentence and utilizing a Gaussian normal distribution function.

5. Using the sentence selection algorithm and marking sentences/text sections as suspicious if they differ *significantly*.

POS tag extraction

After cleaning and splitting the document into single sentences (see Section 2.4.1), POS tags are assigned to each word of a sentence using the *Stanford POS Tagger* [183]. For example, sentence S_1 of the previous section is tagged with the following tags:

The/DT *strongest*/JJS *rain*/NN *ever*/RB *recorded*/VBN *in*/IN *India*/NNP *shut*/VBD *down*/RP *the*/DT *financial*/JJ *hub*/DT *in*/IN *Mumbai*/NNP , *officials*/NNS *said*/VBD *today*/NN .

[17] This section is based on and contentual partly reused from the paper: M. Tschuggnall and G. Specht. *Countering Plagiarism by Exposing Irregularities in Authors' Grammar.* In Proceedings of the European Intelligence and Security Informatics Conference (EISIC), Uppsala, Sweden, August 2013, IEEE, pages 15–22. [186]

Although the information extracted by performing POS tagging only is more shallow than constructing the full parse tree, results discussed in Section 2.5.3 indicate that it is sufficient to find inconsistencies in text building structures. Moreover it is computationally significantly easier, especially with longer and more complex sentences.

In order to reflect the structure information only, the concrete word information is neglected. Thus, for each sentence a POS tag sequence like the following is created and stored:

```
DT-JJS-NN-RB-VBN-IN-NNP-VBD-RP-DT-JJ-DT-IN-NNP-NNS-VBD-NN
```

Distance calculation

Following the original Plag-Inn approach, each sentence is compared against all other sentences by calculating a distance and storing it in a distance matrix:

$$D_n = \begin{pmatrix} 0 & d_{1,2} & \cdots & d_{1,n} \\ * & 0 & \cdots & d_{2,n} \\ * & * & \cdots & d_{3,n} \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \cdots & 0 \end{pmatrix}$$

The key concept used by this algorithm variant is the computation of $d_{i,j}$, i.e., the distance between the POS tag sequences of sentences i and j. Here, the distances are calculated using modified sequence alignment algorithms that are frequently applied in the field of genetics. More concretely, the well-known algorithms *Global Alignment* (Needleman-Wunsch) [124] and *Local Alignment* (Smith-Waterman) [161] have been adapted to align POS sequences.

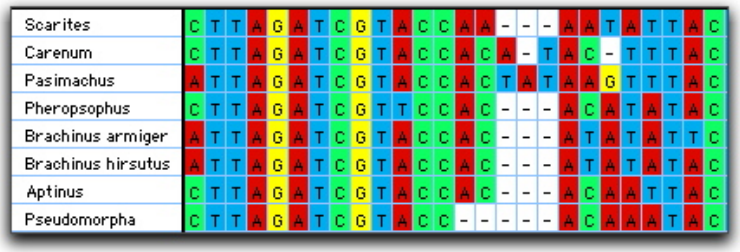


Figure 2.19.: Small Fragment of DNA Sequence Alignment of Different Species.

The original idea of the algorithms is to ideally align two DNA sequences (see Figure 2.19[18]) using cost models that incorporate gene (mis)matches, insertions and deletions, resulting in an overall score of how similar the sequences are. Thereby global alignment solutions align the whole sequences, whereas the local alignment algorithm searches for a local maximum of similarity. For example, using 2 for a match score and -1 for insertion, deletion or mismatch costs, respectively, the similarity value of the DNA-strings[19] `CTCTAGCATT` and `GTGCAC` could result in a score of 2 for the global alignment:

```
C T C T A G C A T T
      |   | | |
- - G T - G C A - C
```

and a score of 7 for the local alignment:

```
T A G C A
|   | | |
T - G C A
```

In case of the plagiarism detection algorithm the alignment methods have been modified to be able to align POS tags. Thereby each tag is treated as a single instance: for example, aligning the two sentences

(1) *This*/DT *is*/VBZ *a*/DT *simple*/JJ *sentence*/NN

(2) *This*/DT *is*/VBZ *not*/RB *the*/DT *most*/RBS *complex*/JJ *construction*/NN

using the same scoring scheme would result in the following global alignment with score 8:

```
DT VBZ -  DT  -  JJ NN
|   |     |      |  |
DT VBZ RB DT RBS JJ NN
```

[18]picture taken from http://www.sequence-alignment.com, visited April 2014

[19]The characters represent the nucleotides contained in a DNA string: Adenine (A), Guanine (G), Cytosine (C), Thymine (T)

In order to not only incorporate the maximum local similarity calculated by the local alignment algorithm, but to reflect the difference of whole sentences, the global alignment algorithm has been chosen as a first attempt. Additionally global alignment scores implicitly also incorporate differences in sentence lengths, a feature which has also been used in previous approaches. Nevertheless, manual analysis of the evaluation of some random documents indicates that there is no significant difference in overall results when using global or local alignment.

As the POS-PlagInn algorithm tries to find inconsistencies rather than similarities between sentence structures, the scoring scheme is inverted, i.e., using a negative value for matches and positive values for mismatches, insertions or deletions, respectively. The weights for all further optimizations and evaluations have preliminary been defined to be -1 for matches, 1 for insertions/deletions and 1 for mismatches. With the inverted scheme, the distance matrix D_n as stated above is calculated by aligning each pair of sentences i and j (i.e. their POS tag sequences) and storing their distance $d_{i,j}$.

An example of a visualized distance matrix of a document containing about 2000 sentences is illustrated in Figure 2.20. It can be seen that there are significant differences in the style of the text around sentence number 900. Additionally it is important to note that these distances are significantly different with respect to *all* other sentences, i.e., they represent not just local peaks.

Predicting Plagiarized Sentences

Having computed all distances between sentences, the further procedure is similar to the Plag-Inn algorithm. Using the distance matrix, the average distance with respect to all other sentences is calculated for each sentence, and the average distance vector is then fitted into a Gaussian normal distribution, resulting in estimations of the mean μ and standard deviation σ.

By using the thresholds δ_{susp} (every sentence with higher mean distance is marked as plagiarized) and δ_{single} (the mean distance of single sentences have to be above this threshold in order to be marked as plagiarized) and applying the sentence selection algorithm as described in Sections 2.4.1 and 2.4.2, respectively, the final prediction of possible plagiarism is made.

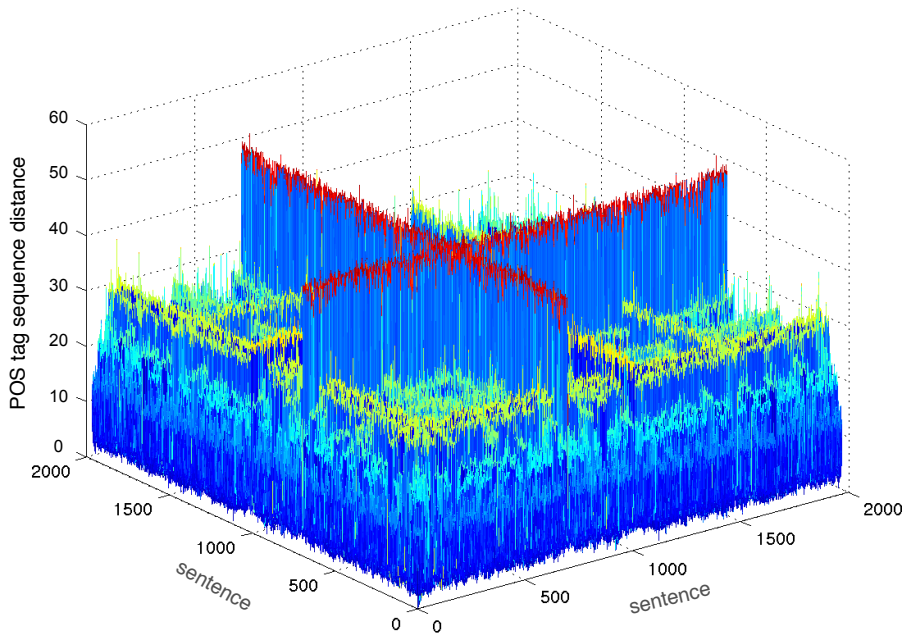


Figure 2.20.: POS-Plag-Inn: Distance Matrix of a Sample Document Consisting of about 2000 Sentences with Global Peaks.[20]

2.5.2. Optimization

The algorithm variant has been optimized similarly to the Plag-Inn algorithm and evaluated on the same data set (PAN-PC-10). Recapitulating Section 2.4.3, the following parameters have been optimized:

- δ'_{susp}: suspicious sentence threshold
- δ'_{single}: single suspicious sentence threshold
- $maxLookahead$: maximum lookahead
- $filterSingles$: boolean switch; should single sentences be filtered using δ'_{single} or not?

As with the Plag-Inn algorithm, genetic algorithms have been used to optimize parameter settings. As a consequence of the improved results of the

[20] © 2013 IEEE

p	δ_{susp}	l_{max}	$filter$	δ_{single}	**Recall**	**Precision**	**F**
200	0.99981	3	yes	0.99998	0.282	0.329	**0.304**
400	0.99998	2	yes	0.99997	0.261	0.319	**0.287**

Table 2.7.: Parameter Optimization Using Genetic Algorithms.[21]

original algorithm by using document subsets of long and short documents, this technique has also been evaluated.

Genetic Optimization Algorithms

The best configuration reaching an F-measure of about 30% can be seen in Table 2.7, where p and l_{max} correspond to population sizes and the maximum lookahead used, respectively. Surprisingly, what can be seen as well is that the best configuration produced by a population size of 400 recommends to filter single-sentence plagiarism sections with a *lower* threshold than multiple-sentence sections. Although a better configuration could be found, this result still represents a local maximum which indicates that authors might as well plagiarize single sentences rather than longer text fragments only.

Genetic Optimization Algorithms On Document Subsets

As done with the original approach, the data set has been split into large and short documents in order to find optimal parameter configurations for the distinct subsets. Table 2.8 shows the best configurations produced by genetic algorithms using the sentence-split document subsets with dividing numbers of 100, 150 and 200 sentences per document, respectively.

With an F-measure of about 52% for documents having less than 100 sentences, and about 22% for all longer documents, respectively, a total F-value of 37% could be achieved with the best parameter configuration. Thereby the algorithm performed significantly better on documents having less sentences on all three evaluation parts, which indicates that it works very good on about scientific-paper-length sizes and has drawbacks on novel-length sizes. A first intuitive conclusion may be that the longer documents get, the more variety on sentence constructions it contains, leading to blurring syntax.

Like in the previous Plag-Inn algorithm, the dividing numbers of 100, 150 and 200 have been chosen manually and are not guaranteed to be optimal. In this sense future work should also incorporate genetic algorithms to find the optimal dividing threshold for the POS-Plag-Inn algorithm.

[21] © 2013 IEEE

subset	δ_{susp}	$maxLook.$	$filterS.$	δ_{single}	**Recall**	**Prec**	**F**
$S_{\geq 100}$	0.99996	3	yes	0.99999	0.188	0.271	**0.222**
$S_{<100}$	0.99999	8	yes	0.99999	0.520	0.524	**0.522**
					0.354	*0.398*	***0.374***
$S_{\geq 150}$	0.96639	10	yes	0.99987	0.158	0.090	**0.115**
$S_{<150}$	0.99975	1	yes	0.99999	0.504	0.527	**0.515**
					0.331	*0.309*	***0.319***
$S_{\geq 200}$	0.99996	9	no	-	0.182	0.250	**0.211**
$S_{<200}$	0.99999	1	yes	0.99992	0.475	0.495	**0.485**
					0.329	*0.373*	***0.349***

Table 2.8.: Parameter Optimization Using Genetic Algorithms on Document Subsets Split by the Number of Sentences.[22]

2.5.3. Evaluation

Reusing the document sets of the PAN 2010 workshop (PAN-PC-10) as described in Section 2.4.3, the algorithm has been trained and optimized with the training set and subsequently evaluated against the test corpus consisting of about 4000 English documents.

Table 2.9 summarizes the evaluation results gained from using the optimization techniques described earlier. With an F-score of 52% the best result could be achieved by applying the parameter setting optimized for documents having less than 100 sentences. While this score has been found by evaluating the corresponding document subset only (< 100 sentences), an F-score of 30% could be reached by using a single parameter configuration over the whole test set.

Furthermore the PAN workshop metric *granularity* has been measured. It describes the "grouping" quality of correctly annotated plagiarized sections, meaning that a continuous plagiarized section should also be annotated as *one* section rather than multiple sections. For example, a granularity value of 2 would denote that the algorithm in average predicted 2 sections instead of one. Hence, a granularity value of 1.0 is optimal, which could be achieved in most of cases as can be seen in Table 2.9.

Regardless of the optimization splitting number of sentences to be contained in a document, the algorithm works best for the document subsets having fewer sentences. While the subsets with documents having more sentences achieve only an F-score of about 10-20%, their counterpart reside with F-scores of

[22] © 2013 IEEE

document subset	**Granularity**	**Recall**	**Precision**	**F**
less than 100 sentences	1.0	0.520	0.524	**0.522**
less than 150 sentences	1.0	0.504	0.527	**0.515**
less than 200 sentences	1.0	0.475	0.495	**0.485**
all documents	1.05	0.282	0.329	**0.304**

Table 2.9.: Best Evaluation Results using the PAN-PC-10 data set.[23]

about 50%. By combining the individual results for split subsets, F-scores of about 35% could be reached. Concretely, the variant using the splitting number of 100 sentences per document worked best with an F-score of 37%. This is also the global optimum in comparison to the evaluation using just a single parameter setting, which achieves an F-score of about 30%. Generally, recall and precision values are balanced in all settings, tending to have a higher precision in most cases. An overview of the global results is depicted in Figure 2.21.

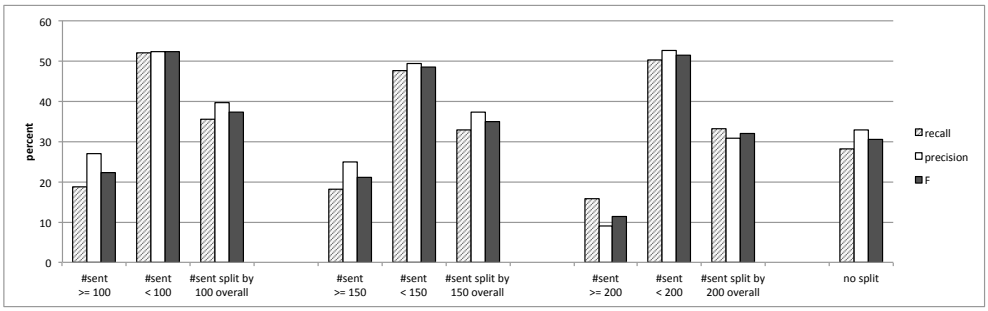


Figure 2.21.: POS-Plag-Inn: Global Evaluation Results.

Similarly to the original approach, the POS tag variant achieves different results when separating documents that contain and not contain plagiarism cases. As shown in Figure 2.23, the algorithm processes plagiarism-free documents at a significantly higher accuracy than documents which contain plagiarism.

In order to avoid overfitted results adjusted to the training set, the variant using just a single parameter configuration over the whole test subset has been used for all further analyses. Therefore, because previous optimizations already showed that the algorithm is sensitive to the number of sentences per document, the single parameter configuration has been evaluated according to the number of sentences. The results illustrated in Figure 2.23 indicate

[23] © 2013 IEEE

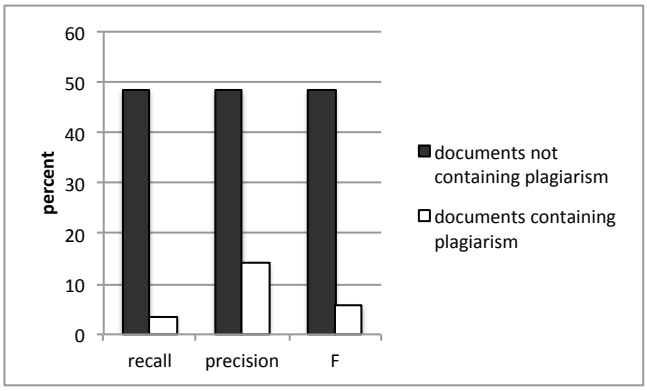


Figure 2.22.: POS-Plag-Inn: Evaluation Results for Documents Containing and Not Containing Plagiarism Cases.

that the less sentences a document has, the better the algorithm operates, i.e., predicts plagiarism correctly. For documents consisting of less than 50 sentences, an F-score of over 65% could be reached. On the other side, a possible explanation for the rather poor results concerning longer, novel-length documents might be that structural syntactic differences are blurred on longer documents. In other words, if an author writes long documents, s/he may also use different sentence construction sets throughout the document in an unconscious or even conscious manner that compensate differences in style.

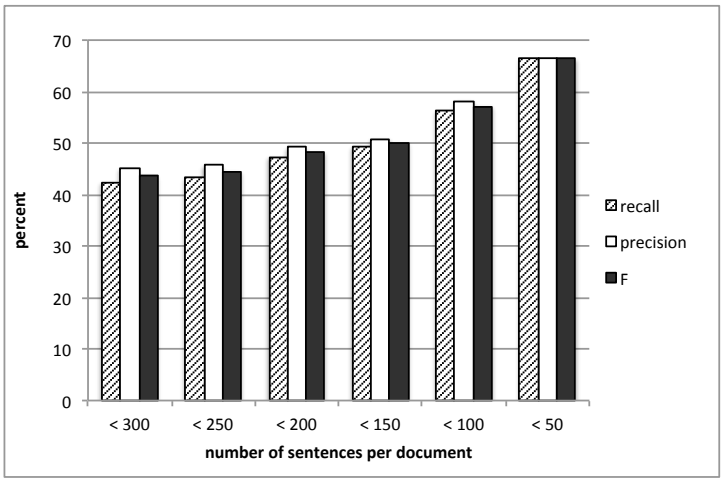


Figure 2.23.: POS-Plag-Inn: Evaluation Results By Number of Sentences Per Document.[24]

[24] © 2013 IEEE

To investigate the results for documents containing plagiarism, the number of false-positives and false-negatives, respectively, have been evaluated. The results depicted in Figure 2.24 show that the number of false-positives is rather high, i.e., the algorithm often predicts plagiarism where there actually is none. Hence, the algorithm should also be improved in the future by eliminating false-negatives.

Diagram (a) in Figure 2.24 illustrates the number of falsely detected plagiarized sections on plagiarism-free documents, and the number of not detected plagiarized sections on documents that contain plagiarism, where the number of false-positives is significantly higher. According to previous observations the rate of false-positives has been compared with the number of sentences contained in a document. The result shown in diagram (b) make clear that the main set of false-positives result from long documents, i.e., the shorter the document is, the less the probability of falsely predicted plagiarism is. Finally Figure 2.25 illustrates that the algorithm predicts too much plagiarism cases in general. For example, if a document contains three plagiarized sections, it is often the case that significantly more than three sections are predicted, which results in a decrease of the F-score for documents containing plagiarism.

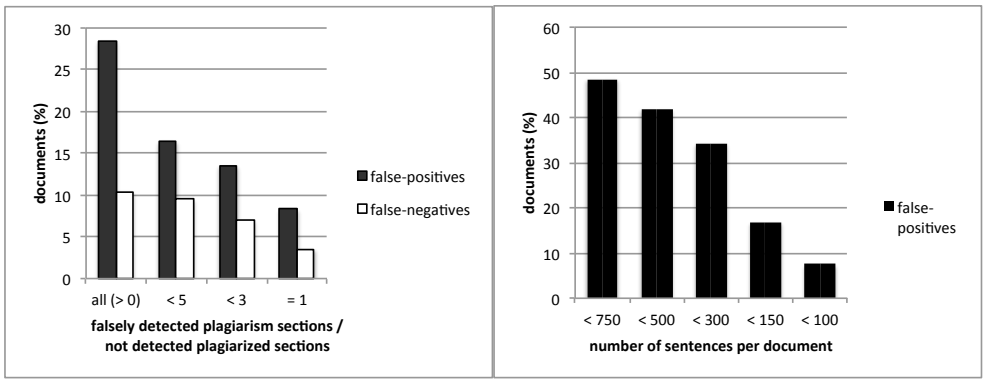


Figure 2.24.: POS-Plag-Inn: False-Positives and False-Negatives.[25]

The number of documents where the algorithm predicted too few and the number of documents where it predicted the exact amount of plagiarism cases are balanced equally. Also for this evaluation, *exact* predictions do not necessarily correspond to *correct* predictions (see Section 2.4.4).

Compared to the original algorithm this variant performs slightly better, especially on short documents. This is a surprising outcome as the POS-Plag-Inn

[25] © 2013 IEEE

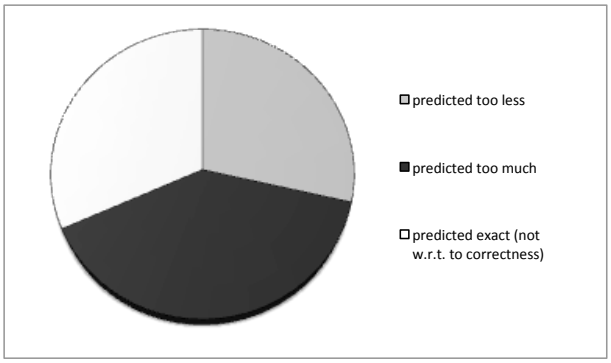


Figure 2.25.: POS-Plag-Inn: Plagiarism Prediction Distribution.[26]

algorithm doesn't reflect the building structure at all but only analyzes the usage of POS tag sequences. A possible explanation might be that the distance metrics produce different results, especially on longer sentences. For example, if an author uses both long and short sentences, the pq-gram distance grows at a significantly higher pace than the global alignment algorithm, as the difference between grammar trees also grows significantly with long sentences. On the other hand, the global alignment distance grows at most linearly with the number of words (using the proposed scoring scheme), and can therefore better smooth distances of long and short sentences.

Nevertheless, the assumption that the analysis of grammar structures may produce better results - according to the fact that trees contain more information than pure POS tags - is evaluated and approved by a third variant of the Plag-Inn algorithm, which is described in the following section.

[26] © 2013 IEEE

2.6. The PQ-PlagInn Algorithm[27]

2.6.1. Algorithm

The intention of the PQ-Plag-Inn variant is to improve the Plag-Inn algorithm by getting rid of sentence-by-sentence comparisons and using grammar tree profiles of whole text fragments instead. Thereby frequencies of pq-grams occurring in predefined sentence windows are compared against the frequencies of pq-grams occurring in the whole document. By applying this idea, full parse trees can be utilized without facing the previously mentioned drawback of the pq-gram distance when comparing grammar trees of significantly different complexity (i.e. long and short sentences).

Summarizing, the PQ-Plag-Inn algorithm consists of the following steps:

1. Cleaning the document and splitting it into single sentences.
2. Computing full parse trees for each sentence.
3. Creating the document profile.
4. Traversing the document using sliding windows and comparing each window profile against the document profile.
5. Utilizing a Gaussian normal distribution and applying the sentence selection algorithm.

The first two steps are equal to the original algorithm described in Section 2.4.1, i.e., the document is cleaned, split into single sentences by using the Stanford Parser, and then a grammar tree is calculated for each sentence. Also in this variant, the terminals of the parse trees are ignored as purely the grammar of authors is investigated. As the main difference in this variant, a pq-gram profile of the whole document is calculated subsequently, which is then compared to profiles computed of sentences in sliding windows. Figure 2.26 depicts the basic sliding window technique, and each of the steps is explained in more detail in the following.

[27] This section is based on and contentual partly reused from the paper: M. Tschuggnall and G. Specht. *Using Grammar-Profiles to Intrinsically Expose Plagiarism in Text Documents.* In Proceedings of the 18th International Conference on Application of Natural Language to Information Systems (NLDB), Salford, UK, June 2013, volume 7934 of LNCS, Springer, pages 297–302. [188]

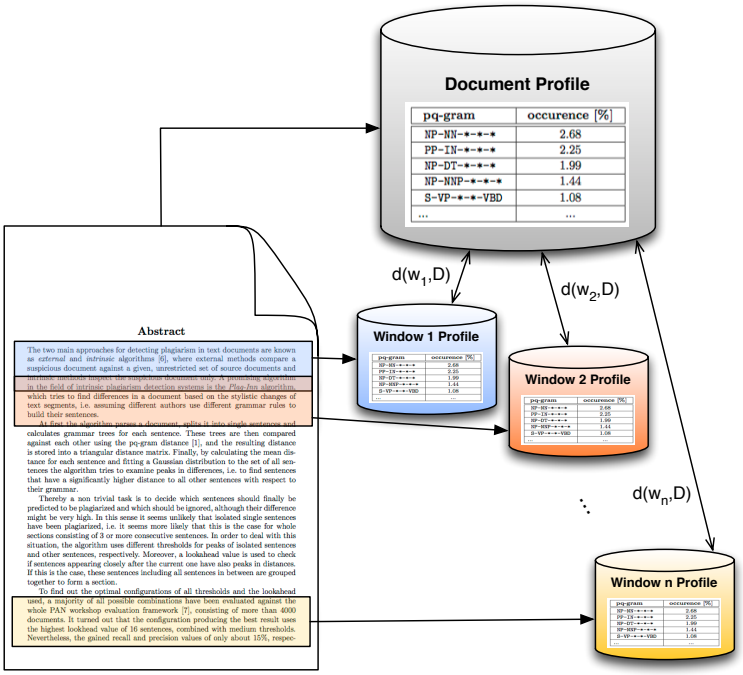


Figure 2.26.: Overview of the PQ-Plag-Inn Algorithm.

Creating the document profile

Having computed a grammar tree for every sentence, the pq-gram index of each tree is calculated and stored. Subsequently, the pq-gram profile of the whole document is calculated by combining all pq-gram indices of all sentences. In this step the number of occurrences is counted for each pq-gram and then normalized by the document length, i.e., normalized by the total number of distinct pq-grams.

As an example, the five mostly used pq-grams (using $p = 2$ and $q = 3$) of a selected document are shown in Table 2.10. The *pq-gram profile* then consists of the complete table of pq-grams and their occurrences in the given document, indicating the *favors* or the *style* of syntax construction used by the (main) author.

Comparison using sliding windows

The basic idea is now to utilize sliding windows and calculate the distance for each window compared to the pq-gram profile. A window has a predefined length l which defines how many sentences should be contained, and the window step s defines the starting point of the windows (i.e. the starting point of window i is equal to the starting point of window $i - 1 + s$).

pq-gram	occurrence [%]
NP-NN-*-*-*	2.68
PP-IN-*-*-*	2.25
NP-DT-*-*-*	1.99
NP-NNP-*-*-*	1.44
S-VP-*-*-VBD	1.08

Table 2.10.: Example of the Five Mostly Used pq-grams of a Sample Document.

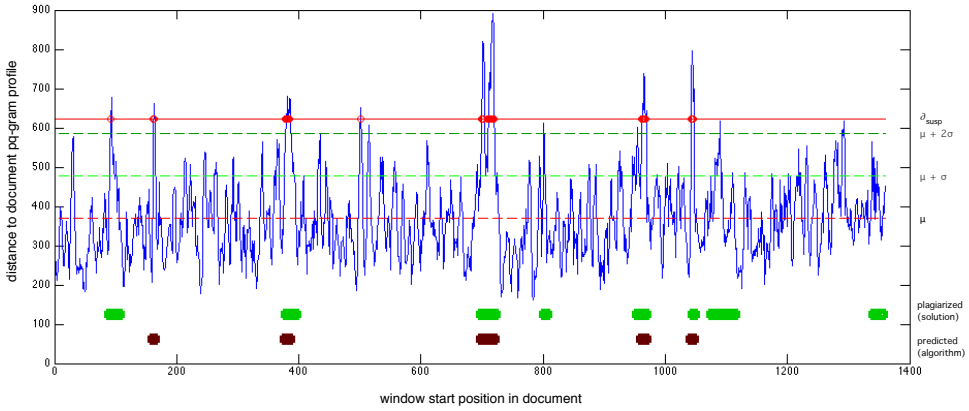


Figure 2.27.: PQ-Plag-Inn: Distances of pq-gram Occurrences of Sliding Windows Compared to the Document Profile.

Then for each window the pq-gram profile $P(w)$ is calculated and compared to the pq-gram profile of the whole document. For calculating the distance, the measure proposed in [169] has been used, as it is well suited for comparing short text fragments (the window w) with large text fragments (the document D):

$$d(w, D) = \sum_{p \in P(w)} \left(\frac{2(f_w(p) - f_D(p))}{f_w(p) + f_D(p)} \right)^2$$

Thereby $f_w(p)$ and $f_D(p)$ denote the normalized frequencies of occurrences of pq-gram p in the window w and the document D, respectively. An example of the sliding window distances of a whole document is illustrated in Figure 2.27. It shows the distance for each sliding window starting position, using a window length of $l = 5$ and a window step of $s = 1$ in this case.

As it can already be seen visually, some sliding windows differ significantly more than others, which may be the case because they contain plagiarized

sentences. The final decision of whether sentences should be predicted as plagiarized or not is made like it is done in all previous algorithm variants, i.e., by fitting a Gaussian normal distribution function and estimating the mean μ and standard deviation σ. By using the same thresholds δ_{susp} and applying the previously presented sentence selection algorithm, the final prediction of plagiarism is made. Note that because of the fact that sliding windows of (at least two) sentences are used in this approach, the threshold δ_{single} is not needed. On the other hand, blocks of sentences may still be grouped together, and initially unmarked sentences may be marked as suspicious if they reside in between two suspicious blocks.

The bottom of the diagram shown in Figure 2.27 depicts an example of the final prediction of the algorithm together with the correct solution obtained from the corresponding test set annotations of the document.

2.6.2. Evaluation

Test Set and parameters

Like the previous variants and for comparability reasons, the PQ-Plag-Inn algorithm has been evaluated using the PAN-PC-10 data set and optimized by using predefined parameter configurations as well as genetic optimization algorithms. Concretely, the parameters shown in Table 2.11 have been used.

Parameter	Description	Range
δ'_{susp}	threshold for sliding windows	$[0.994, 0.995, \ldots, 0.999]$
$maxLookahead$	max. lookahead (sent. select.)	$[2, 3, \ldots, 16]$
l	window length (in sentences)	$[2, 3, \cdots, 10]$
s	window step (in sentences)	$[1, 2]$

Table 2.11.: Configuration Ranges for the Evaluation.

Results

Table 2.12 shows the best evaluation results for each window length and step using the pq-gram configuration $p = 2$ and $q = 3$. It can be seen that this approach performs better than the previous variants, approving the intuitive assumption that the utilization of full parse trees is more expressive than using pure POS tags without structure information.

Given the window length l, the algorithm worked better using a window step of $s = 2$ most of the times. The maximum lookahead used for the sentence selection algorithm resides between 1 and 5, i.e., has a value range like it is expected. Overall, the best result could be produced by using a window length of $l = 4$, a window step of $s = 2$ and a maximum lookahead of 2:

l	s	δ_{susp}	$maxLook.$	**Recall**	**Precision**	**F**
2	1	0.99364	3	0.322	0.362	**0.341**
2	2	0.95848	4	0.301	0.423	**0.352**
3	1	0.99876	2	0.278	0.387	**0.324**
3	2	0.99585	2	0.298	0.423	**0.349**
4	1	0.98954	3	0.267	0.352	**0.303**
4	2	0.99492	2	0.417	0.424	**0.420**
5	1	0.99538	4	0.304	0.374	**0.335**
5	2	0.99609	2	0.358	0.412	**0.383**
6	1	0.97443	4	0.277	0.387	**0.323**
6	2	0.99369	5	0.371	0.411	**0.390**
7	1	0.99207	1	0.359	0.366	**0.362**
7	2	0.99342	2	0.325	0.394	**0.356**
8	1	0.99078	4	0.415	0.379	**0.396**
8	2	0.99914	2	0.410	0.421	**0.415**
9	1	0.99188	2	0.429	0.382	**0.404**
9	2	0.98850	3	0.297	0.418	**0.347**
10	1	0.99795	5	0.378	0.383	**0.381**
10	2	0.99548	4	0.426	0.412	**0.419**

Table 2.12.: Best Evaluation Results of the PQ-Plag-Inn Algorithm.

this configuration could achieve an F-measure of 42%, outperforming all other Plag-Inn variants. A visualization of the results is shown in Figure 2.28.

Also for the PQ-Plag-Inn variant a more detailed evaluation has been conducted to measure its performance. Basically, the results are similar to the previous algorithm variants, i.e., as a main characteristic it performs significantly better on documents that do not contain plagiarism. Moreover, it also works better with decreasing length of the texts as can be seen in Figure 2.29, but the differences between longer and shorter texts is not that high as in other variants, which can be intrepreted in a way that by using profiles the Plag-Inn algorithms becomes more stable.

On the other side and what is illustrated in Figure 2.30, the plagiarism detection distribution differs significantly from the other variants. Where previously the algorithms predicted too much for the majority of documents, this problem could be substantially diminished. As can be seen, for most of the documents the exact amount of plagiarized section has been predicted, i.e., combined with other analyses, this results from predicting non-plagiarized documents correctly. Summarzing, this improvement in the distribution pie is most likely also responsible for the good F-measure achieved by this algorithm variant.

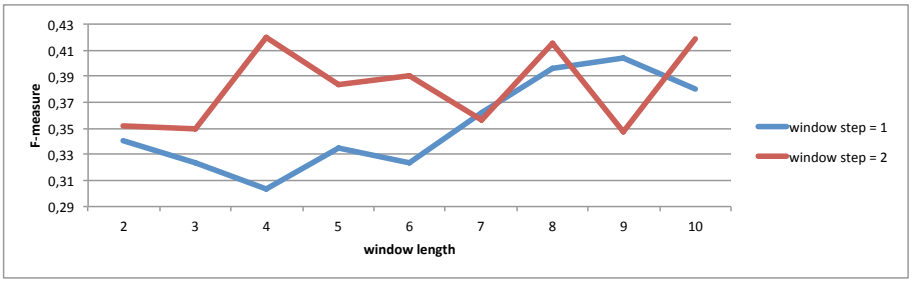


Figure 2.28.: PQ-Plag-Inn: Evaluation Results Using Different Window Lengths and Steps.

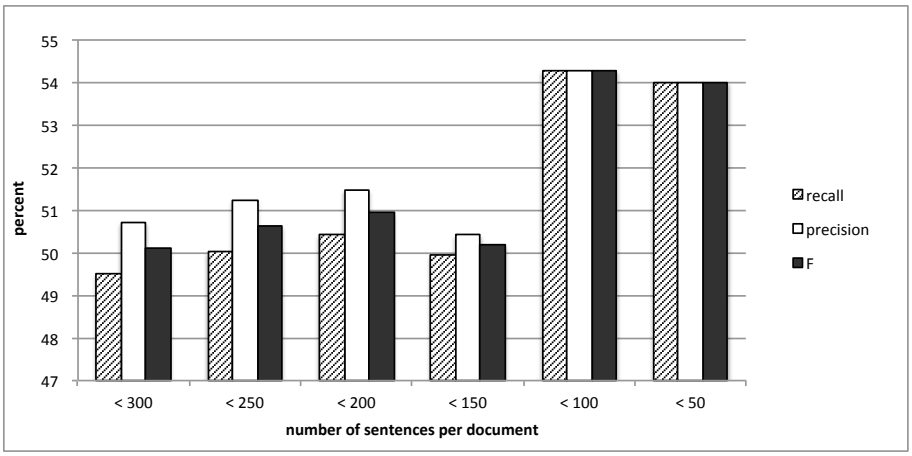


Figure 2.29.: PQ-Plag-Inn: Evaluation Results By Number of Sentences Per Document.

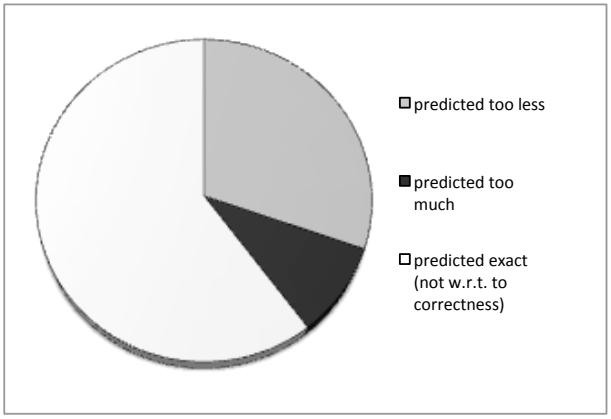


Figure 2.30.: PQ-Plag-Inn: Plagiarism Detection Distribution.

2.7. Conclusion and Future Work

In this chapter, the Plag-Inn algorithm including variants has been described, which is an intrinsic plagiarism detection approach based on a novel style feature. The main idea is to analyze the grammar of authors and to find irregular sentence constructions in order to expose plagiarism. The algorithm thereby uses a full parse tree for every sentence, calculates the corresponding pq-gram index and compares the latter with the index of all other sentences by using the pq-gram distance, to quantify the differences of grammar trees. By fitting a Gaussian normal distribution to the distances and applying a final sentence selection procedure, a prediction of plagiarism is made.

Similarly, the POS-Plag-Inn algorithm is also based on sentences, but analyzes only pure POS tag sequences. The distances between these sequences are then calculated by using dynamic programming algorithms, which are used as a basis for the subsequent steps equally to the original algorithm.

Finally, the major difference between the original algorithm and the third PQ-Plag-Inn variant is that the distance calculation is not based on a sentence-by-sentence comparison, but on profiles of sliding windows. Thereby a profile consists of an ordered list of all occurring pq-grams including their normalized frequencies.

Comparison of the different variants

All three variants of the approach have been evaluated using a state-of-the-art test data set, whereby parameters have been optimized using different methods. Figure 2.31 visualizes a comparison of the performance of the three approaches over the whole test data set (i.e. the figure is showing the results using the best configuration for all the documents rather than optimizations of large/short document subsets). As can be seen, the PQ-Plag-Inn variant performs best with an accuracy of over 40%. Compared to other state-of-the-art algorithms, which achieve at most 30-33% on the same data set, this result is very promising.[28] Considering the good performance of the PQ-Plag-Inn algorithm, the methods used in this variant, i.e., pq-gram profiles, represent the basis also for the authorship attribution, profiling and multi-author decomposition approaches described in the following chapters.

[28] Nevertheless, it has to be stated for fairness reasons that the other approaches on this data set were conducted during a workshop competition, i.e., they had no opportunity to optimize parameters. Although the Plag-Inn algorithms were also optimized for a different training set, they could be tested and refined with the actual data set.

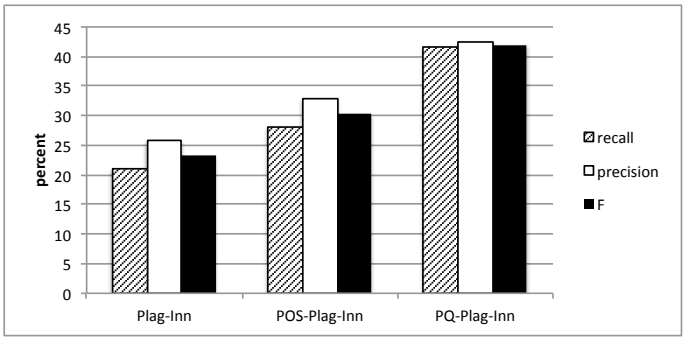


Figure 2.31.: Comparison of the Performances of the Plag-Inn Algorithm Variants.

Future Work

As the algorithms presented in this chapter are based on the grammatical structure, the syntax rules that allow authors to create sentences is of high importance. Consequently, the assumption that the more complex a language is with respect to the grammar rules, the better plagiarism can be detected, should be evaluated. For example, natural languages like German or French give authors more possibilities to formulate text, and thus the style may different more significantly and it should be *easier* for algorithms to expose plagiarism. This should be verified/falsified by doing respective experiments.

Moreover, the algorithms should make use of previous research in the field of intrinsic plagiarism detection. It is to be assumed that by combining existing techniques like n-grams, word-n-grams, sentence lengths or machine learning algorithms with the Plag-Inn algorithm, the accuracy can be improved. Especially approaches that analyze the vocabulary (richness) of the document should be incorporated, as currently the usage of concrete words is completely ignored.

Finally, experiments using other state-of-the-art test sets should be performed to solidify the performance of the algorithms.

CHAPTER 3

Authorship Attribution[1]

3.1. Introduction

The increasing amount of documents available from sources like publicly available literary databases often raises the question of verifying disputed authorships or assigning authors to unlabeled text fragments. As the original problem was initiated already in the midst of the 20$^{\text{th}}$ century by Mosteller and Wallace, who tried to find the correct authorships of *The Federalist Papers* [119] (for details see Sections 3.5.1 and 6.2), authorship attribution is still a major research topic. Especially with latest events in politics and academia, the verification of authorships becomes increasingly important.

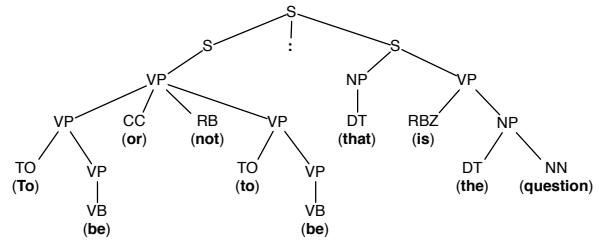
Similar to the intrinsic plagiarism algorithms presented in Chapter 2, which aim to find text fragments not written but claimed to be written by an author, the problem of traditional authorship attribution is defined as follows: Given

[1]This chapter is based on and contentual partly reused from the paper: M. Tschuggnall and G. Specht. *Enhancing authorship attribution by utilizing syntax tree profiles.* In Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL), Gothenburg, Sweden, April 2014, volume 2: Short Papers, pages 195–199. Association for Computational Linguistics. [190]

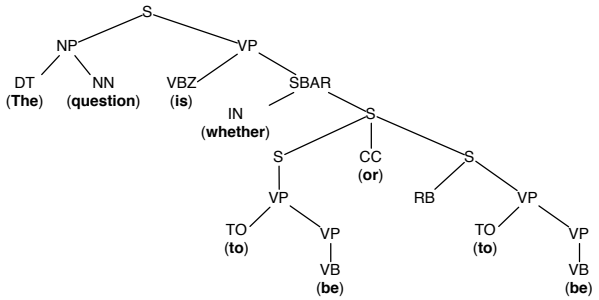
several authors with text samples for each of them, the question is to label a previously unseen document with the correct author. In contrast to the latter *closed-class* problem, a more difficult task is addressed in the *open-class* problem, where additionally a "non-of-them"-answer is allowed. The difficulties of the latter result from the problem that dynamic thresholds have to defined or algorithmically determined, that decide whether the text is written by one of the candidates or not. In other words, defining globally valid thresholds is not sufficient, because they depend on the (previously unknown) set of candidates. An even harder definition of the open-class problem, which is sometimes used and which implies outside-world knowledge, can be summarized in the following question: *"Here is a document, tell me who wrote it."* [80]

Basically the two main approaches are called *profile-based* and *instance-based* [168], whereby the former concatenate all available text samples per author and calculate a single profile according to a predefined set of features, and the latter process individual instances per text one-at-a-time in order to build an attribution model. The majority of algorithms are profile-based, but also instance-based approaches are meaningful, especially in cases when an author writes in different genres that cannot be compared or concatenated (e.g. Goethe wrote lyric, prosa or epic poetry but also scientific texts on the theory of colors). Both approaches frequently apply classification algorithms like support vector machines (e.g. [167, 154]), maximum entropy machine learners (e.g. [178]) or neural networks (e.g. [194]) to learn from samples and to predict the final answer.

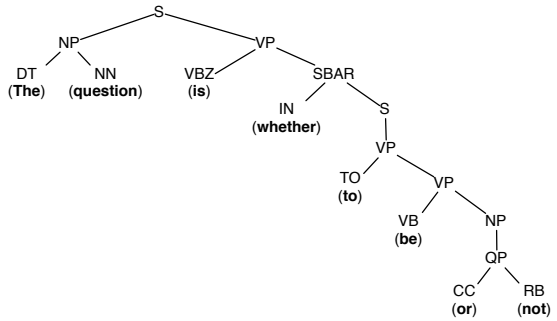
In this thesis a novel profile-based approach for the traditional, closed-class authorship attribution task is presented. It recapitulates the ideas that are implemented in the previously shown intrinsic plagiarism detection algorithms and therefore incorporates the assumption that different authors have different writing styles - in terms of the grammar structure - and that this style is used mostly unconsciously. As depicted earlier, due to the fact that an author has many different choices of how to formulate a sentence using the existing grammar rules of a natural language, the assumption is that an analysis of the sentence construction is sufficient to distinguish between individual authors. For example, the famous Shakespeare quote *"To be, or not to be: that is the question."* could have been also formulated differently like it is shown in Figure 3.1. The main idea of this approach is to quantify those differences by calculating grammar profiles for each candidate author as well as for the unlabeled document, and to assign one of the candidates as the author of the unlabeled document by comparing the profiles. To quantify the differences between profiles, multiple metrics have been implemented and evaluated.



"To be, or not to be: that is the question." (S_3)



"The question is whether to be or not to be." (S_4)



"The question is whether to be or not." (S_5)

Figure 3.1.: Grammar Trees of Variations of a Shakespeare Quote.

The rest of this chapter is organized as follows: Section 3.2 sketches the main idea of the algorithm, which uses the distance metrics explained in Section 3.3 as well as the machine learning algorithms described in Section 3.4. An extensive evaluation using both the metrics and the classifiers on three different test sets is shown in Section 3.5. A comparison on the performance of the two approach variants is given in Section 3.6, and finally Section 3.7 concludes and discusses future work.

3.2. Algorithm

The number of choices an author has to formulate a sentence syntactically is rather high, and the main assumption followed in this thesis is that the concrete choice is made mostly intuitively and unconsciously. Therefore, by incorporating the promising results of the PQ-Plag-Inn algorithm, the basic idea has been modified and enhanced so that it can be applied to traditional authorship attribution. Evaluations shown in Section 3.5 reenforce that solely grammar syntax represents a significant feature that can be used to distinguish between authors.

Globally, the approach comprises the following three steps:

1. Creating a grammar profile for each author.

2. Creating a grammar profile for the unlabeled document.

3. Assigning an author to the document by

    a) Calculating the distance between each author profile and the document profile and assigning the author having the lowest distance or the highest similarity, depending on the metric chosen (see Section 3.3).

    b) Using the author profiles to train classification algorithms and using the document profile to make a machine learning prediction (see Section 3.4).

Calculating Grammar Tree Profiles

One key criterion of the algorithm is the creation of author profiles which are used as a basis to predict authorships. In order to calculate a grammar profile for an author or a document, a similar procedure like it has been used for the PQ-Plag-Inn variant is applied:

1. Concatenate all text samples (of one genre) for the author into a single, large sample document.

2. Split the resulting document into single sentences and calculate a full parse tree for each sentence.

3. Calculate the pq-gram index for each sentence and compose the final grammar profile from the normalized frequencies of pq-grams.

As this approach is profile-based, a single document is created by concatenating all text samples of an author, which serves as a basis for further analyses. On the other hand, for the unlabeled document nothing has be concatenated, and it serves as it is for subsequent computations.

In traditional authorship attribution there usually exist several text samples per author, whereby a recent study found out that "the minimal amount of textual data needed for reliable authorship attribution turned out to be method-independent" [39]. Disregarding the language, the authors state that e.g. for modern novels the minimal amount per author should be around 5,000 words, but also that too much training data leads to worse results. Considering these results, the approach presented in this chapter uses respective data sets, which are shown in Section 3.5.

As a standard procedure, every document is at first cleaned to contain alphanumeric characters and punctuation marks only. Then it is split into single sentences and grammatically parsed like it has been shown earlier with other algorithms. Further on the pq-gram index is calculated for each sentence, which contains all occurring pq-grams of the parse tree. Like it is done in Section 2.6 for the PQ-Plag-Inn algorithm, a pq-gram profile is computed which contains the mostly used pq-grams of an author and the corresponding number of occurrences, which is normalized by the total number of all appearing pq-grams. As an example, the three mostly used pq-grams of a sample document together with their normalized frequencies are {`[NP-NN-*-*-*]`, 2.7%}, {`[PP-IN-*-*-*]`, 2.3%}, and {`[NP-DT-*-*-*]`, 2.0%}. The final pq-gram profile then again consists of the complete table of pq-grams and their occurrences in the given document.

With the use of the grammar tree profiles calculated for each candidate author as well as for the unlabeled document, the last part is to predict the correct author. In order to do this, two approaches have been evaluated: First, a distance or similarity, respectively, is calculated for each candidate author profile and the document profile using different metrics. In this case, the document in question is simply labeled with the author of the best matching profile. Second, machine learning algorithms are trained using the author profiles, and utilized to assign one of the candidate authors to the document.

In the following sections these two methods are explained in detail.

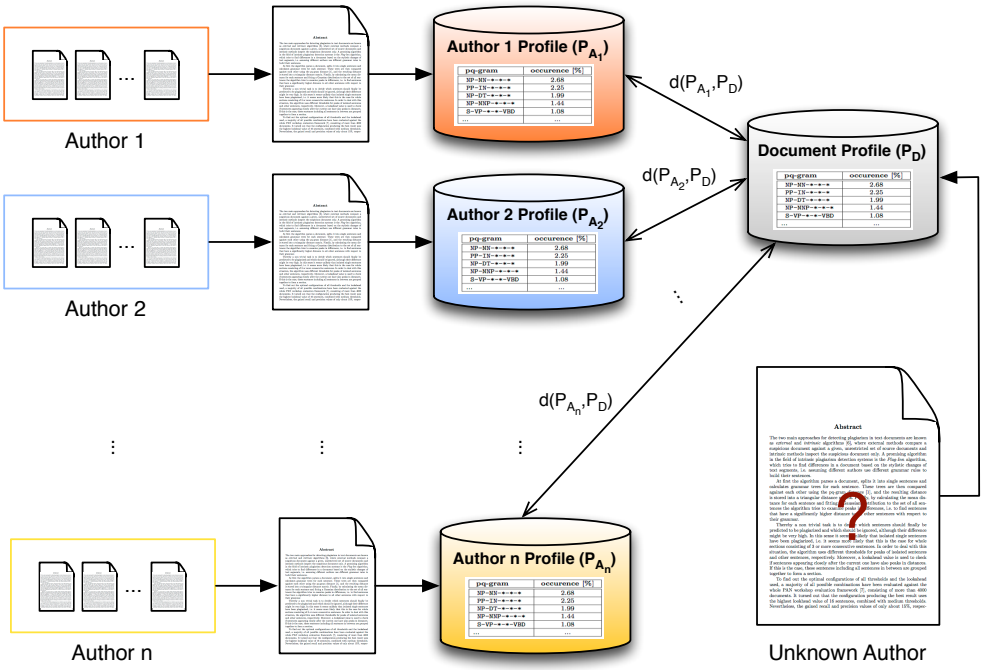


Figure 3.2.: Overview of the Authorship Attribution Approach Using Distance Metrics.

3.3. Distance and Similarity Metrics

The main idea using different distance and similarity metrics to quantify the amount of conformance between author profiles and the document profile is illustrated in Figure 3.2. For each author, all known text samples are concatenated and transformed into a pq-gram profile. Likewise a profile is calculated for the unlabeled document. The key idea is now to calculate differences between the profiles by utilizing different metrics, and to choose the author with the best matching profile, i.e., the profile with the lowest distance in case of a distance-metric or the profile with the highest similarity value in case of a similarity metric, respectively.

Formally, this procedure could be written as follows: Given the grammar profiles $P_1, ..., P_n$ corresponding to authors $A_1, ..., A_n$ and the document profile P_D, the document is labeled with author A_x if

$$\begin{cases} d_{P_x,P_D} = min(d_{P_1,P_D}, ..., d_{P_n,P_D}) \\ \qquad \text{if the metric used calculates distances} \\ s_{P_x,P_D} = max(s_{P_1,P_D}, ..., s_{P_n,P_D}) \\ \qquad \text{if the metric used calculates similarities} \end{cases}$$

where d_{P_x,P_D} represents the distance between author profile P_x and the document profile P_D, and s_{P_x,P_D} represents the similarity between those profiles, respectively.

To investigate on the best distance or similarity metric to be used for this approach, several well-known metrics[2] for this problem have been adapted and evaluated. Subsequently the metrics are shown in detail.

CNG

This metric was proposed by Keselj et al. [86] for comparing common n-grams (CNG) of two profiles by calculating the dissimilarity as follows:

$$d_{P_x,P_D} = \sum_{p \in P_x \cup P_D} \left(\frac{2(f_x(p) - f_D(p))}{f_x(p) + f_D(p)} \right)^2$$

where $f_x(p)$ and $f_D(p)$ denote the normalized frequencies of occurrences of pq-gram p in author profile P_x and the document profile P_D, respectively. For example, given the toy profiles (of pq-grams with $p = q = 1$)

$P_x =$

pq-gram	f_x [%]
S-NP	15
NP-NP	50
NP-VP	35

$P_D =$

pq-gram	f_D [%]
S-NP	10
S-VP	20
NP-NP	30
NP-VP	40

the distance can be calculated as follows:

[2]The metric names are only used as a reference, but are not originally proposed like this by the authors.

$$
\begin{aligned}
d_{P_x,P_D} &= \sum_{p \in P_x \cup P_D} \left(\frac{2(f_x(p) - f_D(p))}{f_x(p) + f_D(p)} \right)^2 \\
&= \left(\frac{2(f_x(\texttt{S-NP}) - f_D(\texttt{S-NP}))}{f_x(\texttt{S-NP}) + f_D(\texttt{S-NP})} \right)^2 + \left(\frac{2(f_x(\texttt{NP-NP}) - f_D(\texttt{NP-NP}))}{f_x(\texttt{NP-NP}) + f_D(\texttt{NP-NP})} \right)^2 \\
&\quad + \left(\frac{2(f_x(\texttt{NP-VP}) - f_D(\texttt{NP-VP}))}{f_x(\texttt{NP-VP}) + f_D(\texttt{NP-VP})} \right)^2 + \left(\frac{2(f_x(\texttt{S-VP}) - f_D(\texttt{S-VP}))}{f_x(\texttt{S-VP}) + f_D(\texttt{S-VP})} \right)^2 \\
&= \left(\frac{2(15 - 10)}{15 + 10} \right)^2 + \left(\frac{2(50 - 30)}{50 + 30} \right)^2 + \left(\frac{2(35 - 40)}{35 + 40} \right)^2 + \left(\frac{2(0 - 20)}{0 + 20} \right)^2 \\
&= 0.16 + 0.25 + 0.02 + 4 = 4.43
\end{aligned}
$$

Stamatatos-CNG

A variation of the latter dissimilarity function was proposed by Stamatatos [169] to reduce the problem that often test corpora are imbalanced. Therefore it uses only the n-grams (pq-grams) that are present in the unseen document's profile:

$$
d_{P_x,P_D} = \sum_{p \in P_D} \left(\frac{2(f_x(p) - f_D(p))}{f_x(p) + f_D(p)} \right)^2
$$

Stamatatos-CNG-CN

The following adaption also proposed by Stamatatos [166] results in a non-symmetric function that additionally introduces a *corpus norm* N, which is the concatenation of all samples of all candidate authors:

$$
d_{P_x,P_D} = \sum_{p \in P_D} \left(\frac{2(f_x(p) - f_D(p))}{f_x(p) + f_D(p)} \right)^2 \cdot \left(\frac{2(f_x(p) - f_N(p))}{f_x(p) + f_N(p)} \right)^2
$$

Here, $f_N(p)$ represents the relative frequency of occurrence of the pq-gram p in the corpus norm N.

Sentence-SPI

Originally proposed by Frantzeskou et al. in the context of source code author identification [48] using n-grams, the *simplified profile intersection (SPI)* similarity metric simply counts the amount of common n-grams of the author profile and the document profile, whereby non-common n-grams are ignored:

$$s_{P_x,P_D} = \sum_{p \in P_D} \begin{cases} 1 & \text{if } p \in P_x \\ 0 & \text{else} \end{cases}$$

According to the idea of this thesis to analyze sentences, the SPI score has been modified for this approach so that each sentence of the unlabeled document is traversed separately. Let S_D be the set of sentences of the document and $I(s)$ the pq-gram index of sentence s, then the Sentence-SPI score is calculated as follows:

$$s_{P_x,P_D} = \sum_{s \in S_D} \sum_{p \in I(s)} \begin{cases} 1 & \text{if } p \in P_x \\ 0 & \text{else} \end{cases}$$

3.4. Machine Learning Algorithms

3.4.1. Utilized Classifiers

A different way to predict authorships is the utilization of state-of-the-art machine learning algorithms. As shown in Figure 3.3 the general method is to use the profiles of the candidate authors to train the respective classifiers. Then, given the profile of the unseen document, the algorithms are asked to make a prediction of authorship.

In particular, the following classifiers have been utilized using the WEKA toolkit as a general framework [67]:

- Naive Bayes classifier [78]
- Bayes Network using the K2 classifier [34]
- Large Linear Classification using LibLinear [45]
- Support vector machines using LIBSVM with nu-SVC classification [29]
- A k-nearest-neighbors classifier (kNN) using $k = 1$ [4]

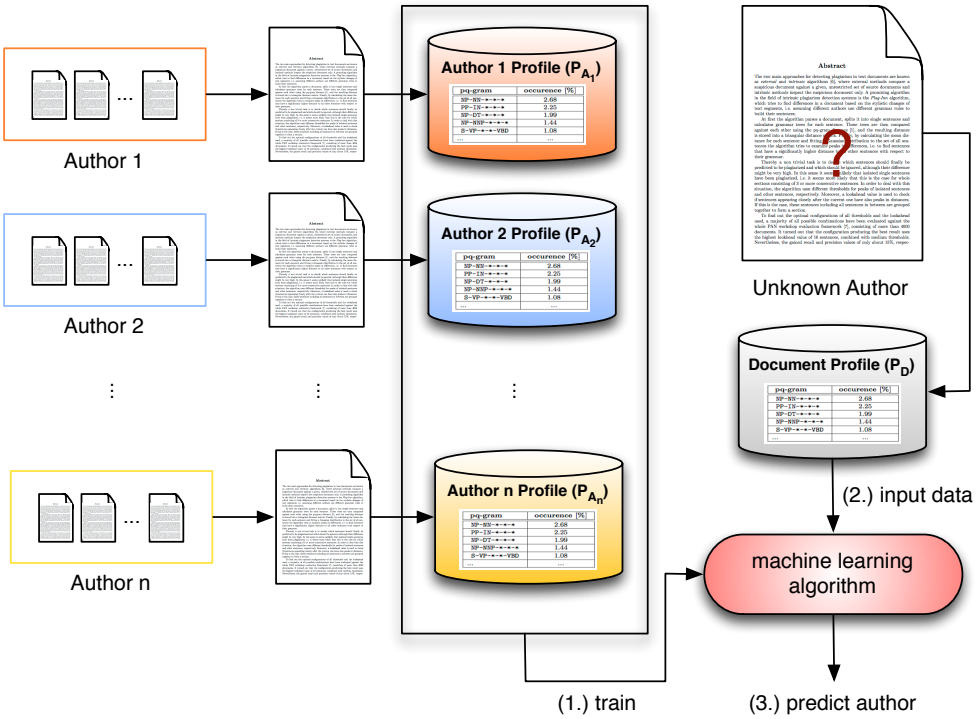


Figure 3.3.: Overview of the Authorship Attribution Approach Using Machine Learning Algorithms.

- Pruned C4.5 decision tree (J48) [148]

For each classification algorithm the most commonly used parameter configuration has been chosen for the evaluation.

3.4.2. Features

Basically, every pq-gram including the frequency of occurrence represents a feature that is used as input to train the classifiers. Depending on the size of the document, the number of distinct features, i.e., pq-grams, range between 1,000 and 15,000. In addition, the profile is sorted descending by the normalized occurrence, and an additional rank value is introduced that simply defines a natural order. For example, Table 3.1 shows the most frequently used pq-grams (with $p = 2, q = 2$) of the conclusion chapter of this thesis[3]. Both the occurrence rate and the rank are used as input features for classification algorithms, as well as a combination of both.

[3]Note that the sentence tag `S` is not among the five mostly used pq-grams, which results from the rather long sentences as a stylistic characteristic of the author of this thesis.

pq-gram	Occurrence [%]	Rank
NP-NN-*-*	4.07	1
NP-DT-*-*	2.94	2
NP-NNS-*-*	2.90	3
PP-IN-*-*	2.56	4
NP-JJ-*-*	2.18	5

Table 3.1.: Example of the Five Mostly Used pq-grams of the Conclusion Chapter of this Thesis.

In order to use the rank value as input for machine learning algorithms, respective features are created by appending the original pq-gram with the token `--RANK`. A small excerpt of a complete feature list is depicted in Table 3.2. Moreover, if a document does not contain a specific feature, i.e., a pq-gram, the feature value for the pq-gram as well as for the corresponding rank is set to −1. In case of the example given in Table 3.2, author C didn't use the structure `[PP-IN-*-*-*]` to build his/her sentences, and thus the according feature values are set to −1.

Feature	Author A	Author B	Author C
NP-NN-*-*	4.07	1.89	2.84
NP-NN-*-*--RANK	1	6	2
NP-DT-*-*	2.94	0.24	-1
NP-DT-*-*-RANK	2	153	-1
NP-NNS-*-*	2.90	2.11	1.23
NP-NNS-*-*-RANK	3	2	11
...	...	...	...

Table 3.2.: Example of a Feature List Serving as Input for Classification Algorithms.

3.5. Evaluation

The authorship attribution approach using grammar analysis has been extensively evaluated using different unrelated data sets, which are described in Section 3.5.1. Moreover, the algorithm using the distance metrics has been optimized by introducing and adapting several parameters. The optimization and the final results using the distance metrics with the adjusted parameter settings are shown in Section 3.5.2, and the results gained from the utilization of the machine learning algorithms are presented in Section 3.5.3.

Task	Candidate Authors	Docs (Train)	Docs (Test)	Words/Doc
A	3	2	6	1800-6060
B	3	2	6	1800-6060
C	8	2	8	up to 13000
D	8	2	8	up to 13000

Table 3.3.: Statistics of the PAN12 Corpus.

3.5.1. Test Data Sets

Three different English data sets have been chosen to evaluate the approach, whereby all sets are completely unrelated and of different types. In case of the variant using distance metrics, one of the data sets has been used to train the algorithm, i.e. to optimize parameters, while the remaining sets have been used as test sets. On the other hand, for the machine learning algorithms that are not depending on parameters, each data set has been evaluated individually.

The PAN'12 competition corpus (PAN12)

As a well-known, state-of-the-art corpus especially created for the use in authorship identification, parts of the PAN2012 corpus [81] have been integrated. The corpus is composed of several fiction texts and split into several subtasks that cover small- and common-length documents (1800-6060 words) as well as larger documents (up to 13000 words) and novel-length documents (up to 170,000 words).

For each subtask of the original workshop competition the number of authors are different, numbering from three up to eight candidate authors that have to be labeled on six up to eight unseen documents. Corresponding to every author present in the data set, separate samples are provided. The PAN 2012 corpus also includes open-class tasks, which have been removed from the set as this thesis addresses only closed-class attribution.

Concretely, the data from the tasks as shown in Table 3.3 of the original competition have been used and modified (open-class documents have been removed).

The 2004-competition corpus (CC04)

From the data set originally created for the Ad-hoc-Authorship Attribution Competition workshop [82] held in 2004[4], the training set has been used. It consists of eight chapters of novels written by Anne, Charlotte and Emily

[4] http://www.mathcs.duq.edu/~juola/authorship_contest.html, visited Oct. 2013

Bronte and an additional candidate author that is not present in the text samples (i.e. the latter should not be labeled to any of the documents).

For each author, a sample document of about 2,500 words is given, and the size of the eight documents to be labeled range from about 1,500 to 2,500 words.

The Federalist Papers (FED)

Probably the mostly referred text corpus in the field of authorship attribution is a series of 85 political essays called "The Federalist Papers" written by John Jay, Alexander Hamilton and James Madison in the 18th century. While most of the authorships are undoubted, many works have studied and questioned the correct authorship of 12 disputed essays [119].

Although there is a common opinion on the authorship of the latter essays, an amount of uncertainty remains. Therefore, the disputed essays have been removed for the evaluation. Additionally, three essays co-written by two of the authors have also been discarded as this would be a multi-class problem which is not addressed in this approach.

To build the author profiles the first three documents written by each author have been concatenated (paper numbers {1, 6, 7} for Hamilton, {2, 3, 4} for Jay and {10, 14, 37} for Madison). These documents were also excluded for the evaluation, resulting in an overall set of 61 documents to be attributed. Thereby the amount of words per document ranges from 900 up to 3,500.

3.5.2. Distance Metrics Results

Parameter Optimization

The algorithm using the distance metrics depends on several parameters that have been optimized to gain the best results. Basically the most important parameters are the selection of the distance or similarity metric as well as the size of pq-grams, i.e., the length of the stem p and the base q. By choosing large values for the latter the grammar structure becomes more important, whereas on the other side low values result in less structure information (e.g. by choosing $p = 1$ and $q = 0$ the pq-grams of a grammar profile are equal to the POS tag which provides no structure data at all).

Besides the metric and values for p and q, two additional optimization variables have been integrated for the similarity metric Sentence-SPI:

- **topPQGramCount t_c:** by assigning a value to this parameter, only the corresponding amount of mostly used pq-grams of a grammar profile are used. For example, by setting t_c = 100 only the hundred mostly used pq-grams of the author profiles as well as of the document profile are used.

- **topPQGramOffset t_o:** based on the idea that all authors might have a frequently used and common set of syntax rules that are predefined by a specific language, this parameter allows to ignore the given amount of mostly used pq-grams. For example if t_o = 3 in Table 3.1, the three mostly used pq-gram patterns are ignored by the individual metrics, i.e. the first pq-gram to be used would be `[NP-NNP-*-*-*]`.

  If a common set of grammar rules exists, this parameter helps to optimize the algorithm by reducing the blur emerging from these rules, and helps to focus on actual differences between authors. As evaluations indicate, a common set indeed exists - at least for the evaluated English language.

For each of the parameters different values have been tested, and all resulting permutations of variable assignments have been used for the evaluation. The concrete assignments are shown in the Table 3.4.

Param.	**Range**
p, q	$[0, \ldots, 6]$ ($p = 0, q = 0$ is excluded)
t_c	$[-, 5, 10, 15, \ldots, 200]$
t_o	$[-, 3, 5, 10, 15, 20, 25, 30]$

Table 3.4.: Configuration Ranges for Parameter Optimization.

Results

The evaluation results are shown in Table 3.5. It shows the rate of correct author attributions based on the grammar feature presented in this paper. Therefore three different evaluations have been made using all three data sets as optimization sets: i.e. one of the data sets has been chosen to optimize the algorithm, and the remaining ones have been evaluated with the optimized configurations.

For each table the best result for each distance metric is listed, and for the similarity metric *Sentence-SPI* the best three results are shown as there are more parameters to be optimized on one side, and additionally this metric produced the best score on the other side. All optimization sets could be optimized very well, headed by the CC04 set with a 100% rate of correct author attributions and followed by the Federalist Papers with 95%.

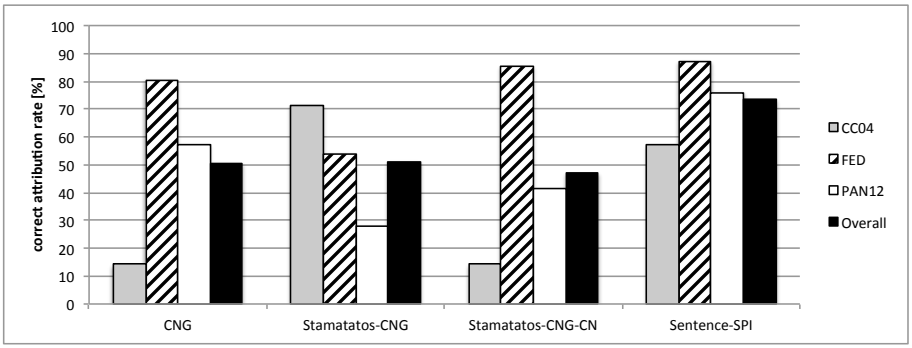


Figure 3.4.: Authorship Attribution: Distance Metrics Results.

Composed by the average correct attribution rate of the remaining test sets, the overall results are shown in the last two columns, where the latter includes the optimized data set. Generally, the algorithm worked best using the *Sentence-SPI* score, which led to a correct-rate of 73% by using the PAN12 data set for optimization. The optimal configuration uses $p = 3$ and $q = 2$, which is the same configuration that was used in [12] (for identifying and mapping street names) to produce the best results.

It can be seen that the highest scores are gained by using a limit of top pq-grams ($t_c \sim 65$) and by ignoring the first three pq-grams ($t_o = 3$), which indicates that it is sufficient to limit the number of syntax structures and that there exists a certain number (3) of general grammar rules for English which are used by *all* authors. Consequently those rules cannot by used to infer information about individual authors (for example every sentence starts with `[S-...]`).

Moreover, all results are better using the PAN12 data set for optimization, which may be because this set is the most heterogeneous one: The Federalist Papers contain only political essays written some time ago, and the CC04 set only uses literary texts written by four authors.

Finally, an overview of the best performances by each distance metric is illustrated in Figure 3.4.

3.5.3. Machine Learning Results

Using the same data sets, the approach idea has been evaluated with the specified classifiers and feature sets. In particular, the feature sets

- `OCCURRENCE-RATE` (concrete percentages of pq-gram occurrences)

metric	**p**	**q**	t_c	t_o	**FED (Opt.)**	**PAN12**	**CC04**	**Overall**	***Overall Opt.***
Sentence-SPI	4	0	65	3	**95.08**	48.91	14.29	31.60	*52.76*
Sentence-SPI	3	2	70	3	90.16	57.04	**57.14**	57.09	*68.11*
Sentence-SPI	3	2	65	3	86.89	**63.79**	**57.14**	**60.46**	***69.27***
Stamatatos-CNG	2	2	-	-	81.97	46.53	14.29	30.41	*47.60*
Stamatatos-CNG-CN	0	2	-	-	85.25	32.54	14.29	23.42	*44.03*
CNG	0	2	-	-	81.97	29.76	14.29	22.03	*42.01*

(a) Results Using FED for Optimization.

metric	**p**	**q**	t_c	t_o	**PAN12 (Opt.)**	**FED**	**CC04**	**Overall**	***Overall Opt.***
Sentence-SPI	3	6	90	10	**87.50**	70.49	28.57	49.53	*62.19*
Sentence-SPI	3	2	65	3	76.04	**86.89**	**57.14**	**72.02**	***73.36***
Sentence-SPI	3	3	60	10	68.75	83.61	**57.14**	70.38	*69.83*
Stamatatos-CNG	2	3	-	-	60.42	78.69	14.29	46.49	*51.13*
CNG	0	2	-	-	57.29	80.33	14.29	47.31	*50.64*
Stamatatos-CNG-CN	4	5	-	-	64.58	3.28	42.86	23.07	*36.91*

(b) Results Using PAN12 for Optimization.

metric	**p**	**q**	t_c	t_o	**CC04 (Opt.)**	**FED**	**PAN12**	**Overall**	***Overall Opt.***
Sentence-SPI	2	5	55	5	**100.0**	63.93	46.23	55.08	***70.05***
Sentence-SPI	2	5	35	25	**100.0**	60.66	29.27	44.96	*63.31*
Sentence-SPI	5	0	45	3	85.71	**67.21**	**47.13**	**57.17**	*66.68*
Stamatatos-CNG	1	4	-	-	71.43	54.10	23.51	38.81	*49.68*
Stamatatos-CNG-CN	1	4	-	-	42.86	52.46	19.64	36.05	*38.32*
CNG	1	2	-	-	57.14	18.03	27.98	23.00	*34.38*

(c) Results Using FED for Optimization.

Table 3.5.: Evaluation Results in Percent By Using Distance Metrics and Different Data Sets For Parameter Optimization.

- `RANK` (position in descending ordered list of pq-gram occurrences), and
- `ALL` (combination of both)

have been evaluated. As with the distance metric evaluation, for each data set the corresponding known text samples of each author have been used to train the classifiers, while the remainder has been used for testing.

Incorporating the functioning of machine learning algorithms, the previously shown optimization using the two parameters t_c and t_o has not been used in this case. It can be assumed that the classification algorithms optimize these parameters implicitly by automatically selecting and deselecting individual features in order to produce optimal results. Nevertheless, concrete attribute selection algorithms [65] like best-first, branch-and-bound, simulated annealing [106] or greedy algorithms [28] have not been utilized for this evaluation, but should be addressed in future work.

A complete list of results evaluating the classifiers with pq-gram settings of $2 \leq p \leq 4$ and $2 \leq q \leq 4$ are presented in Table 3.6 for the `OCCURRENCE-RATE` feature set, in Table 3.7 for the `RANK` feature set and in Table 3.8 for the `ALL` feature set, respectively. In each table the three best results are highlighted.

Generally, the `ALL` feature set achieved the best results, followed by the `RANK` and `OCCURRENCE-RATE` feature sets. With an overall accuracy of 76.9% the classification worked best by using the LibLinear classifier with all features and $p = 2, q = 4$, outperforming the distance metric results by approximately 4%. In all three evaluations the linear classifier as well as the support vector machine classifier worked best, whereas the decision tree produced the worst results. The general poor performance of decision trees in the field of authorship attribution has also been confirmed in a comparative study using many different features [204].

Like it is the case in the previously shown distance metric evaluation, the classification using the grammar profiles works very well for the Federalist Papers and PAN12 data sets, but lacks of predicting the CC04 data set. For example, the best configuration achieved 90.0% for the Federalist Papers and 96.9% for the PAN12 data set, respectively, but reached only 42.9% for CC04. It can be seen in Table 3.8 that the individual data sets - except CC04 - could be classified very well: The linear classifier could attribute 97% of the Federalist papers correctly, and kNN could even classify 100% of the PAN12 data set correctly (i.e. all four subtasks could be attributed completely correct).

An overview of the classification algorithms and their performance on the respective data sets is depicted in Figure 3.5.

					PAN12					
Classifier	**p**	**q**	**CC04**	**FED**	**A**	**B**	**C**	**D**	**Avg**	**Overall**
kNN	2	2	28.6	71.2	100.0	100.0	87.5	100.0	96.9	65.6
Naive Bayes	2	2	28.6	78.8	33.3	33.3	62.5	87.5	54.2	53.8
BayesNet	2	2	28.6	80.8	33.3	33.3	62.5	87.5	54.2	54.5
LibLinear	2	2	28.6	81.4	50.0	66.7	75.0	87.5	69.8	59.9
LibSVM	2	2	14.3	80.0	50.0	66.7	75.0	87.5	69.8	54.7
J48	2	2	57.1	77.5	33.3	33.3	12.5	12.5	22.9	52.5
kNN	2	3	28.6	75.8	100.0	100.0	87.5	100.0	96.9	67.1
Naive Bayes	2	3	42.9	82.6	33.3	33.3	62.5	62.5	47.9	57.8
BayesNet	2	3	42.9	85.9	33.3	33.3	62.5	75.0	51.0	59.9
LibLinear	2	3	42.9	87.1	100.0	83.3	62.5	87.5	83.3	71.1
LibSVM	2	3	28.6	84.5	100.0	83.3	62.5	75.0	80.2	64.4
J48	2	3	14.3	83.3	33.3	33.3	12.5	12.5	22.9	40.2
kNN	2	4	14.3	84.8	100.0	100.0	87.5	87.5	93.8	64.3
Naive Bayes	2	4	42.9	88.6	33.3	33.3	75.0	50.0	47.9	59.8
BayesNet	2	4	42.9	89.4	33.3	33.3	75.0	50.0	47.9	60.1
LibLinear	2	4	42.9	89.8	100.0	100.0	62.5	62.5	81.3	71.3
LibSVM	2	4	28.6	86.7	100.0	100.0	62.5	75.0	84.4	66.5
J48	2	4	42.9	85.9	33.3	33.3	12.5	12.5	22.9	50.5
kNN	3	2	28.6	77.3	100.0	83.3	87.5	87.5	89.6	65.1
Naive Bayes	3	2	42.9	85.6	33.3	33.3	62.5	62.5	47.9	58.8
BayesNet	3	2	28.6	87.4	33.3	33.3	62.5	62.5	47.9	54.6
LibLinear	3	2	42.9	88.3	83.3	83.3	75.0	87.5	82.3	71.1
LibSVM	3	2	28.6	85.5	66.7	83.3	75.0	75.0	75.0	63.0
J48	3	2	71.4	83.8	33.3	33.3	25.0	37.5	32.3	62.5
kNN	3	3	28.6	81.8	100.0	100.0	87.5	100.0	96.9	69.1
Naive Bayes	3	3	42.9	87.1	33.3	33.3	62.5	87.5	54.2	61.4
BayesNet	3	3	42.9	89.4	33.3	33.3	62.5	87.5	54.2	62.1
LibLinear	3	3	42.9	89.4	66.7	83.3	62.5	87.5	75.0	69.1
LibSVM	3	3	42.9	86.4	66.7	66.7	75.0	87.5	74.0	67.7
J48	3	3	14.3	83.8	33.3	33.3	12.5	25.0	26.0	41.4
kNN	3	4	42.9	78.8	83.3	83.3	87.5	87.5	85.4	69.0
Naive Bayes	3	4	42.9	84.1	33.3	33.3	75.0	87.5	57.3	61.4
BayesNet	3	4	42.9	85.9	33.3	33.3	75.0	87.5	57.3	62.0
LibLinear	3	4	57.1	87.5	50.0	100.0	75.0	87.5	78.1	74.3
LibSVM	3	4	42.9	84.8	50.0	83.3	75.0	87.5	74.0	67.2
J48	3	4	14.3	85.6	33.3	33.3	37.5	25.0	32.3	44.1
kNN	4	2	28.6	77.3	100.0	83.3	62.5	100.0	86.5	64.1
Naive Bayes	4	2	57.1	81.8	33.3	33.3	50.0	62.5	44.8	61.3
BayesNet	4	2	57.1	83.3	33.3	33.3	50.0	62.5	44.8	61.8
LibLinear	**4**	**2**	**42.9**	**84.8**	**66.7**	**66.7**	**50.0**	**75.0**	**64.6**	**64.1**
LibSVM	**4**	**2**	**57.1**	**82.7**	**83.3**	**66.7**	**62.5**	**100.0**	**78.1**	**72.7**
J48	4	2	57.1	81.8	33.3	33.3	12.5	37.5	29.2	56.0
kNN	4	3	42.9	78.8	83.3	100.0	75.0	75.0	83.3	68.3
Naive Bayes	4	3	57.1	83.3	33.3	33.3	50.0	62.5	44.8	61.8
BayesNet	4	3	71.4	84.3	33.3	33.3	50.0	75.0	47.9	67.9
LibLinear	**4**	**3**	**71.4**	**84.8**	**50.0**	**83.3**	**50.0**	**75.0**	**64.6**	**73.6**
LibSVM	4	3	57.1	82.7	66.7	83.3	62.5	75.0	71.9	70.6
J48	4	3	14.3	81.6	33.3	33.3	25.0	25.0	29.2	41.7
kNN	4	4	42.9	84.8	66.7	83.3	75.0	87.5	78.1	68.6
Naive Bayes	4	4	57.1	86.4	33.3	33.3	75.0	62.5	51.0	64.8
BayesNet	4	4	57.1	86.9	33.3	33.3	62.5	62.5	47.9	64.0
LibLinear	4	4	57.1	87.5	50.0	83.3	62.5	75.0	67.7	70.8
LibSVM	4	4	57.1	84.8	50.0	83.3	62.5	75.0	67.7	69.9
J48	4	4	71.4	82.6	33.3	33.3	25.0	25.0	29.2	61.1

Table 3.6.: Evaluation Results in Percent By Utilizing Classifiers With the Occurence-Rate Feature Set.

Classifier	p	q	CC04	FED	PAN12 A	PAN12 B	PAN12 C	PAN12 D	PAN12 Avg	Overall
kNN	2	2	14.3	74.2	100.0	100.0	100.0	100.0	100.0	62.8
Naive Bayes	2	2	42.9	81.1	33.3	33.3	87.5	75.0	57.3	60.4
BayesNet	2	2	28.6	83.8	33.3	33.3	87.5	75.0	57.3	56.6
LibLinear	**2**	**2**	**42.9**	**86.7**	**100.0**	**100.0**	**87.5**	**100.0**	**96.9**	**75.5**
LibSVM	**2**	**2**	**42.9**	**84.2**	**100.0**	**100.0**	**87.5**	**100.0**	**96.9**	**74.7**
J48	2	2	42.9	81.8	33.3	33.3	12.5	12.5	22.9	49.2
kNN	2	3	14.3	80.3	100.0	100.0	100.0	100.0	100.0	64.9
Naive Bayes	2	3	28.6	85.6	33.3	33.3	87.5	100.0	63.5	59.2
BayesNet	2	3	28.6	87.4	33.3	33.3	87.5	100.0	63.5	59.8
LibLinear	2	3	28.6	89.0	100.0	100.0	87.5	100.0	96.9	71.5
LibSVM	2	3	28.6	86.1	100.0	100.0	87.5	100.0	96.9	70.5
J48	2	3	28.6	82.8	33.3	33.3	12.5	12.5	22.9	44.8
kNN	2	4	14.3	87.9	100.0	100.0	100.0	100.0	100.0	67.4
Naive Bayes	2	4	28.6	89.4	33.3	33.3	75.0	87.5	57.3	58.4
BayesNet	2	4	28.6	89.9	33.3	33.3	75.0	87.5	57.3	58.6
LibLinear	**2**	**4**	**28.6**	**90.5**	**100.0**	**100.0**	**87.5**	**100.0**	**96.9**	**72.0**
LibSVM	2	4	28.6	87.3	100.0	100.0	87.5	100.0	96.9	70.9
J48	2	4	28.6	84.8	33.3	33.3	25.0	12.5	26.0	46.5
kNN	3	2	28.6	77.3	100.0	100.0	100.0	87.5	96.9	67.6
Naive Bayes	3	2	42.9	82.6	33.3	33.3	75.0	75.0	54.2	59.9
BayesNet	3	2	42.9	84.3	33.3	33.3	75.0	75.0	54.2	60.5
LibLinear	3	2	28.6	86.4	100.0	83.3	87.5	100.0	92.7	69.2
LibSVM	3	2	14.3	83.9	100.0	83.3	87.5	87.5	89.6	62.6
J48	3	2	85.7	81.6	33.3	33.3	37.5	12.5	29.2	65.5
kNN	3	3	28.6	80.3	100.0	100.0	87.5	100.0	96.9	68.6
Naive Bayes	3	3	42.9	84.8	33.3	33.3	75.0	87.5	57.3	61.7
BayesNet	3	3	42.9	86.4	33.3	33.3	75.0	87.5	57.3	62.2
LibLinear	3	3	28.6	87.9	100.0	100.0	87.5	100.0	96.9	71.1
LibSVM	3	3	28.6	85.2	100.0	100.0	87.5	100.0	96.9	70.2
J48	3	3	28.6	82.8	33.3	33.3	25.0	25.0	29.2	46.9
kNN	3	4	28.6	77.3	83.3	100.0	100.0	87.5	92.7	66.2
Naive Bayes	3	4	42.9	84.8	33.3	33.3	75.0	75.0	54.2	60.6
BayesNet	3	4	42.9	86.9	33.3	33.3	75.0	75.0	54.2	61.3
LibLinear	3	4	28.6	88.6	100.0	100.0	87.5	87.5	93.8	70.3
LibSVM	3	4	28.6	85.8	100.0	100.0	87.5	87.5	93.8	69.4
J48	3	4	14.3	83.3	33.3	33.3	37.5	37.5	35.4	44.3
kNN	4	2	28.6	74.2	83.3	100.0	87.5	75.0	86.5	63.1
Naive Bayes	4	2	42.9	81.8	33.3	33.3	50.0	75.0	47.9	57.5
BayesNet	4	2	42.9	84.3	33.3	33.3	50.0	75.0	47.9	58.4
LibLinear	4	2	28.6	85.6	66.7	100.0	75.0	75.0	79.2	64.4
LibSVM	4	2	28.6	83.3	33.3	83.3	75.0	87.5	69.8	60.6
J48	4	2	57.1	84.1	33.3	33.3	12.5	0.0	19.8	53.7
kNN	4	3	42.9	78.8	83.3	100.0	87.5	75.0	86.5	69.4
Naive Bayes	4	3	42.9	85.6	33.3	33.3	75.0	75.0	54.2	60.9
BayesNet	4	3	42.9	87.9	33.3	33.3	75.0	75.0	54.2	61.6
LibLinear	4	3	42.9	88.6	50.0	100.0	75.0	75.0	75.0	68.8
LibSVM	4	3	42.9	85.8	33.3	100.0	87.5	75.0	74.0	67.5
J48	4	3	14.3	83.3	33.3	33.3	25.0	37.5	32.3	43.3
kNN	4	4	14.3	77.3	83.3	83.3	87.5	75.0	82.3	58.0
Naive Bayes	4	4	42.9	84.8	33.3	33.3	75.0	75.0	54.2	60.6
BayesNet	4	4	57.1	86.9	33.3	33.3	75.0	75.0	54.2	66.1
LibLinear	4	4	42.9	88.6	66.7	83.3	75.0	75.0	75.0	68.8
LibSVM	4	4	42.9	85.8	50.0	83.3	87.5	75.0	74.0	67.5
J48	4	4	71.4	83.6	33.3	33.3	37.5	12.5	29.2	61.4

Table 3.7.: Evaluation Results in Percent By Utilizing Classifiers With The Rank Feature Set.

					PAN12					
Classifier	**p**	**q**	**CC04**	**FED**	**A**	**B**	**C**	**D**	**Avg**	**Overall**
kNN	2	2	42.9	74.2	100.0	100.0	100.0	100.0	100.0	72.4
Naive Bayes	2	2	14.3	74.2	83.3	83.3	87.5	50.0	76.0	54.8
BayesNet	2	2	28.6	86.4	33.3	33.3	87.5	87.5	60.4	58.5
LibLinear	2	2	28.6	90.9	100.0	100.0	87.5	100.0	96.9	72.1
LibSVM	2	2	28.6	74.2	83.3	100.0	87.5	100.0	92.7	65.2
J48	2	2	42.9	60.0	33.3	33.3	12.5	12.5	22.9	41.9
kNN	2	3	28.6	78.8	100.0	100.0	100.0	100.0	100.0	69.1
Naive Bayes	2	3	28.6	90.9	83.3	83.3	75.0	62.5	76.0	65.2
BayesNet	2	3	28.6	90.9	33.3	33.3	75.0	75.0	54.2	57.9
LibLinear	**2**	**3**	**42.9**	**92.4**	**100.0**	**100.0**	**75.0**	**100.0**	**93.8**	**76.4**
LibSVM	2	3	42.9	74.2	100.0	100.0	87.5	87.5	93.8	70.3
J48	2	3	28.6	77.3	33.3	33.3	12.5	12.5	22.9	42.9
kNN	2	4	14.3	89.4	100.0	100.0	100.0	100.0	100.0	67.9
Naive Bayes	2	4	42.9	74.2	66.7	100.0	25.0	75.0	66.7	61.3
BayesNet	2	4	28.5	92.4	33.3	33.3	75.0	87.5	57.3	59.4
LibLinear	**2**	**4**	**42.9**	**90.9**	**100.0**	**100.0**	**87.5**	**100.0**	**96.9**	**76.9**
LibSVM	2	4	42.9	74.2	100.0	100.0	87.5	100.0	96.9	71.3
J48	2	4	28.6	80.3	33.3	33.3	12.5	12.5	22.9	43.9
kNN	3	2	28.6	80.3	100.0	100.0	87.5	87.5	93.8	67.6
Naive Bayes	3	2	42.9	74.2	66.7	83.3	50.0	37.5	59.4	58.8
BayesNet	3	2	42.9	87.9	33.3	33.3	62.5	75.0	51.0	60.6
LibLinear	3	2	28.6	92.4	100.0	83.3	75.0	87.5	86.5	69.2
LibSVM	3	2	28.6	74.2	100.0	83.3	75.0	87.5	86.5	63.1
J48	3	2	85.7	72.7	33.3	33.3	37.5	25.0	32.3	63.6
kNN	3	3	28.5	81.8	100.0	100.0	100.0	100.0	100.0	70.1
Naive Bayes	**3**	**3**	**42.9**	**92.4**	**100.0**	**100.0**	**62.5**	**87.5**	**87.5**	**74.3**
BayesNet	3	3	42.9	92.4	33.3	33.3	62.5	87.5	54.2	63.2
LibLinear	3	3	28.5	92.4	83.3	100.0	75.0	100.0	89.6	70.2
LibSVM	3	3	42.9	74.2	83.3	100.0	87.5	100.0	92.7	69.9
J48	3	3	28.6	80.3	33.3	33.3	25.0	25.0	29.2	46.0
kNN	3	4	28.5	81.8	83.3	100.0	100.0	87.5	92.7	67.7
BayesNet	3	4	42.9	90.9	33.3	33.3	75.0	87.5	57.3	63.7
Naive Bayes	3	4	42.9	74.2	66.7	66.7	62.5	50.0	61.5	59.5
LibLinear	3	4	28.6	97.0	83.3	100.0	75.0	87.5	86.5	70.7
LibSVM	3	4	28.6	74.2	83.3	100.0	87.5	87.5	89.6	64.1
J48	3	4	14.3	83.3	33.3	33.3	37.5	37.5	35.4	44.3
kNN	4	2	28.6	75.8	83.3	83.3	87.5	100.0	88.5	64.3
Naive Bayes	4	2	42.9	74.2	83.3	100.0	37.5	25.0	61.5	59.5
BayesNet	4	2	42.9	89.4	33.3	33.3	50.0	87.5	51.0	61.1
LibLinear	4	2	28.6	87.9	83.3	100.0	75.0	87.5	86.5	67.7
LibSVM	4	2	28.6	74.2	66.7	83.3	75.0	87.5	78.1	60.3
J48	4	2	57.1	81.8	33.3	33.3	12.5	37.5	29.2	56.0
kNN	4	3	42.9	80.3	83.3	100.0	87.5	75.0	86.5	69.9
Naive Bayes	4	3	14.3	74.2	66.7	83.3	50.0	75.0	68.8	52.4
BayesNet	4	3	42.9	92.4	33.3	33.3	62.5	75.0	51.0	62.1
LibLinear	4	3	42.9	89.4	100.0	100.0	75.0	75.0	87.5	73.3
LibSVM	4	3	42.9	74.2	66.7	100.0	75.0	75.0	79.2	65.4
J48	4	3	14.3	72.7	33.3	33.3	25.0	62.5	38.5	41.8
kNN	4	4	28.6	81.8	83.3	83.3	75.0	75.0	79.2	63.2
Naive Bayes	4	4	57.1	74.2	66.6	83.3	75.0	50.0	68.7	66.7
BayesNet	4	4	57.1	90.9	33.3	33.3	75.0	75.0	54.2	67.4
LibLinear	4	4	42.9	89.4	100.0	100.0	62.5	75.0	84.4	72.2
LibSVM	4	4	42.8	74.2	66.7	100.0	75.0	75.0	79.2	65.4
J48	4	4	71.4	75.8	33.3	33.3	37.5	12.5	29.2	58.8

Table 3.8.: Evaluation Results in Percent By Utilizing Classifiers With All Features.

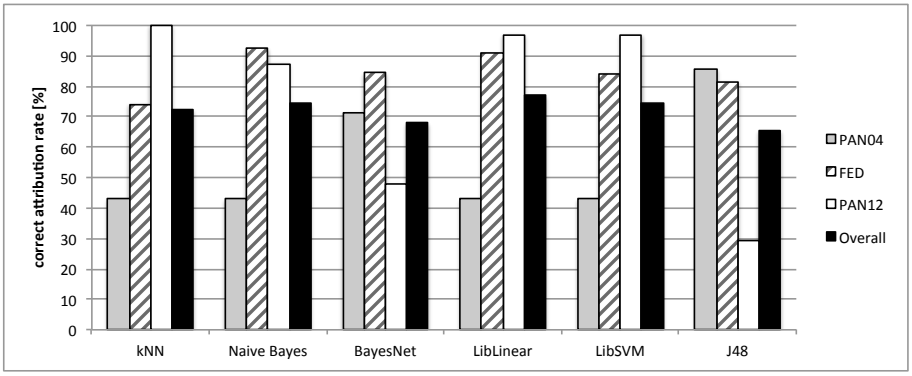


Figure 3.5.: Authorship Attribution: Best Machine Learning Results.

3.6. Comparison of Variants

Both evaluation variants lack of attributing the CC04 set correctly, or at least at an acceptable rate. Considering that the CC04 data set contains only eight documents compared to, for example, the over sixty-document Federalist Papers data set, the calculation of the overall result is decreased significantly as it averages equally over the individual data set results. The assumption that the CC04 data set may not be representative, or at least contains some flaws, is strengthened by evaluations conducted for comparisons presented in Chapter 5, where it is revealed that the approach performs significantly better also on other (self-made) test sets. At last, the poor performance on the CC04 set conforms also to the results of other approaches that found the works of the Bronte sisters - which form the CC04 set - as the hardest to discriminate [98].

In Table 3.9 the evaluation results are revisited and analyzed by the impact of the CC04 data set. It shows the ten best performances of the authorship attribution approach including and excluding the CC04 data set, and discriminates the attribution rate that can be achieved when calculating the overall score by task and by document. Thereby the score by task is equal to the previously presented results, i.e., each data set is treated equally, disregarding the number of documents to be attributed. On the other hand, the attribution rate by document represents the global attribution rate with respect to the quantity of unlabeled documents. For example, if the CC04 data set is included, the overall attribution rate is calculated as follows:

$$rate_{task} = \frac{rate_{CC04}+rate_{FED}+rate_{PAN12}}{3}$$

$$rate_{doc} = \frac{correct_{CC04}+correct_{FED}+correct_{PAN12}}{n_{CC04}+n_{FED}+n_{PAN12}} = \frac{correct_{CC04}+correct_{FED}+correct_{PAN12}}{8+61+(6+6+8+8)}$$

metric	p	q	t_c	t_o	Opt. For	CC04	FED	PAN12	Overall incl. CC04 By Task	Overall incl. CC04 By Doc	Overall excl. CC04 By Task	Overall excl. CC04 By Doc
Sentence-SPI	3	2	65	3	PAN12	57.14	86.89	76.04	73.36	**82.2**	81.5	**84.4**
Sentence-SPI	3	2	65	3	FED	57.14	86.89	76.04	73.36	**82.2**	81.5	**84.4**
Sentence-SPI	3	2	70	3	FED	57.14	90.16	67.71	71.67	**81.8**	78.9	**84.0**
Sentence-SPI	4	0	65	3	FED	14.29	95.08	57.29	55.55	**78.3**	76.2	**84.1**
Sentence-SPI	3	3	60	10	PAN12	57.14	83.61	68.75	69.83	**77.9**	76.2	**79.8**
Sentence-SPI	3	6	90	10	PAN12	28.57	70.49	87.50	62.19	**72.7**	79.0	**76.7**
Stamatatos-CNG	2	2	-	-	FED	14.29	81.97	57.29	51.18	**70.0**	69.6	**75.0**
CNG	0	2	-	-	PAN12	14.29	80.33	57.29	50.64	**68.9**	68.8	**73.9**
Stamatatos-CNG	2	3	-	-	PAN12	14.29	78.69	60.42	51.13	**68.8**	69.6	**73.8**
Stamatatos-CNG-CN	0	2	-	-	FED	14.29	85.25	41.67	47.07	**67.5**	63.5	**72.4**

(a) Distance Metric Results Revisited.

Classifier	p	q	Feature Set	CC04	FED	PAN12	Overall incl. CC04 By Task	Overall incl. CC04 By Doc	Overall excl. CC04 By Task	Overall excl. CC04 By Doc
LibLinear	2	3	ALL	42.9	92.4	93.8	76.4	**89.6**	93.1	**93.9**
LibLinear	2	4	ALL	42.9	90.9	96.9	76.9	**89.6**	93.9	**93.8**
LibLinear	3	4	ALL	28.6	97.0	86.5	70.7	**89.2**	91.8	**94.8**
LibLinear	2	2	ALL	28.6	90.9	96.9	72.1	**88.4**	93.9	**93.8**
LibLinear	2	4	RANK	28.6	90.5	96.9	72.0	**88.2**	93.7	**93.6**
Naive Bayes	3	3	ALL	42.9	92.4	87.5	74.3	**87.8**	90.0	**91.9**
LibLinear	3	3	ALL	28.5	92.4	89.6	70.2	**87.2**	91.0	**92.6**
LibLinear	2	3	RANK	28.6	89.0	96.9	71.5	**87.2**	92.9	**92.5**
kNN	2	4	ALL	14.3	89.4	100.0	67.9	**87.2**	94.7	**93.8**
LibLinear	2	2	RANK	42.9	86.7	96.9	75.5	**86.9**	91.8	**91.0**

(b) Machine Learning Results Revisited.

Table 3.9.: Evaluation Results Analyzing the Impact of the CC04 Data Set.

where $correct_X$ denotes the number of correct attributions of data set X, and n_X the total number of documents to be attributed on the same data set, respectively. The score without the CC04 set is calculated similarly, without the corresponding variables.

By computing the attribution rate by document, the imbalance in quantity of the test sets is compensated. Consequently, the overall score is increased significantly for both the distance metrics comparisons and the machine learning attributions. As with the calculation by task, the approach evaluated by number of documents performs better by utilizing classifier algorithms. The linear classification reaches an attribution rate of 89% when the CC04 data set is included, and an attribution rate of nearly 94% when it is excluded. Again, the machine learning classification works best with using all available features (pq-gram occurrences and ranks) and the pq-gram size of $p = 2, q = 3$.

Likewise, also the distance metrics result is significantly enhanced by inspecting the rate by document. The setting for which the parameters have been optimized for PAN12 reaches 82% (including CC04) and 84% (excluding CC04) by using $p = 3, q = 2$. In addition it can be seen that among the best ten performances the CC04 data set has never been used for optimizing parameters, which strengthens the assumption that the CC04 data set may not be representative for evaluating the approach.

Summarizing, the attribution rates for the distance metrics and the classifiers are presented in Figure 3.6. It shows the percentage of correctly attributed documents over the test data sets, including and excluding the CC04 data set.

3.7. Conclusion and Future Work

This chapter describes how the grammar syntax of writers can be used in the field of authorship attribution. To do this, for every known author full parse trees of all sentences are computed and transformed into pq-gram profiles. These profiles are then compared to the document profile of unknown authorship by following two approaches: (1.) Calculating distances between the profiles by applying different distance metrics, and (2.) utilizing common machine learning algorithms with different sets of features (pq-grams).

Both variants have been evaluated using three different data sets, containing various attribution problems. The distance metric approach has been additionally optimized by introducing parameters to exclude a certain number of mostly used pq-grams and to restrict the set of incorporated pq-grams. It could be examined that the best result with an overall accuracy of 73% sug-

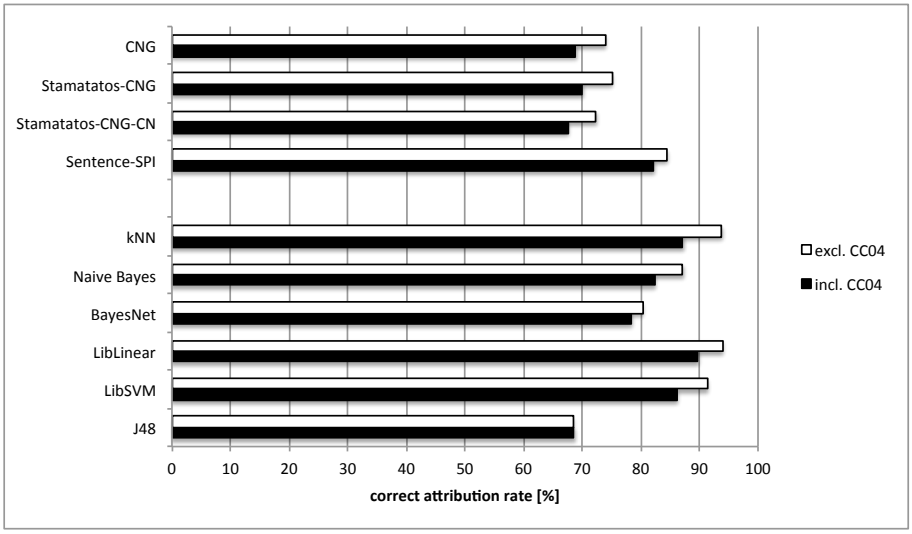


Figure 3.6.: Authorship Attribution: Evaluation Results By Correctly Attributed Documents Including and Excluding the CC04 Data Set.

gests to only use the 65 most frequently used pq-grams for the profile creation, and to ignore the first three pq-grams. On the other side, the machine learning evaluation did not use manual optimization, but nevertheless could outperform the distance metrics and achieve a best accuracy of about 77%. Thereby it could be seen that the best results can be gained by including all available features as input for the classifiers.

Future Work

As this approach is based on the ideas and algorithms of the intrinsic plagiarism detection approaches described in Section 2, future work is similar. Because the authorship attribution approach solely uses the grammar feature and completely ignores information like the concrete words used, vocabulary richness or n-grams, future work should concentrate on integrating other well-known and good-working features to be able to build a reliable and heterogeneous authorship attribution tool. Furthermore, the optimization of the parameters in the distance metrics comparison is currently only applied on the similarity score and could also be integrated with the distance metrics as they led to the best results.

Analogical to the previously presented approaches, research should finally also be done on the applicability on other languages, as syntactically more complex languages may also lead to good or even better authorship attribution rates. Finally, also other heavily used classification algorithms like Markov

Chains [85], decision tables [91] or tree-based techniques like Random Forest [23] should be evaluated, which may produce different results.

CHAPTER 4

Profiling Gender and Age of Authors[1]

4.1. Introduction

The automatic classification of data has become a major research topic in the last years, and especially the analysis of written text has gained attraction due to the easy availability of huge amounts of online documents such as web blogs, social media postings or literary databases. Particularly useful information like the gender or the age of the originator is of interest if, for example, groups of writers are targeted or analyzed and the individual author is dispensable.

In contrast to traditional authorship attribution approaches that try to assign one of several known candidate authors to an unlabeled document as addressed in Chapter 3, the author profiling problem deals with the extraction of concrete meta information about the author. Often this information includes gender

[1]This chapter is based on and contentual partly reused from the paper: M. Tschuggnall and G. Specht. *What Grammar Tells About Gender and Age of Authors.* In Proceedings of the 4th International Conference on Advances in Information Mining and Management (IMMM), Paris, France, July 2014, pages 30–35. [191]

and age of the originator [8, 47, 156], but also other demographic information like cultural background, level of education or psychological traits are examined in recent approaches [201, 128]. Where the mining of such information can be applied very well to commercial applications by knowing the percentages of gender and age commenting on a new product release, for example, it is also of growing importance in juridical applications (*Forensic Linguistics*) [54], where, e.g., the number of possible perpetrators can be reduced. Moreover especially nowadays in the area of cybercrime [127], recent approaches investigate the content of e-mails [2], suicide letters or try to automatically expose sexual predators from chat logs [75].

In this chapter, the ideas, algorithms and results of the plagiarism detection as well as the authorship attribution approaches are adapted in a way that they can be applied to the automatic profiling of authors. Concretely, the assumption that authors can be distinguished by their usage of grammar is reused and evaluated for the the classes *gender* and *age*. Thereby a main difference to the previously described algorithms is that the profiling approach exploits the grammar of whole author groups (e.g. females), instead of individual writers.

The remainder of this chapter is organized as follows: Section 4.2 describes the adapted, machine-learning-based algorithm including the utilized classifiers and features. An extensive evaluation is then presented in Section 4.3, while Section 4.4 concludes and discusses future work.

4.2. Profiling Authors Using pq-gram Profiles

The global idea of this thesis has been consequently reused for the profiling task, i.e., the grammar of authors is analyzed. Concretely, the assumption that individual authors have significantly different writing styles in terms of the syntax that is used to construct sentences has been reutilized. As an additional example to the previously presented, the following sentences[2] are semantically equivalent, but differ with respect to the grammar building structure as can be seen in Figure 4.1:

"My chair started squeaking a few days ago and it's driving me nuts." (S_6)

"Since a few days my chair is squeaking - it's simply annoying." (S_7)

[2]Sentence S_6 has been extracted from an anonymized web blog.

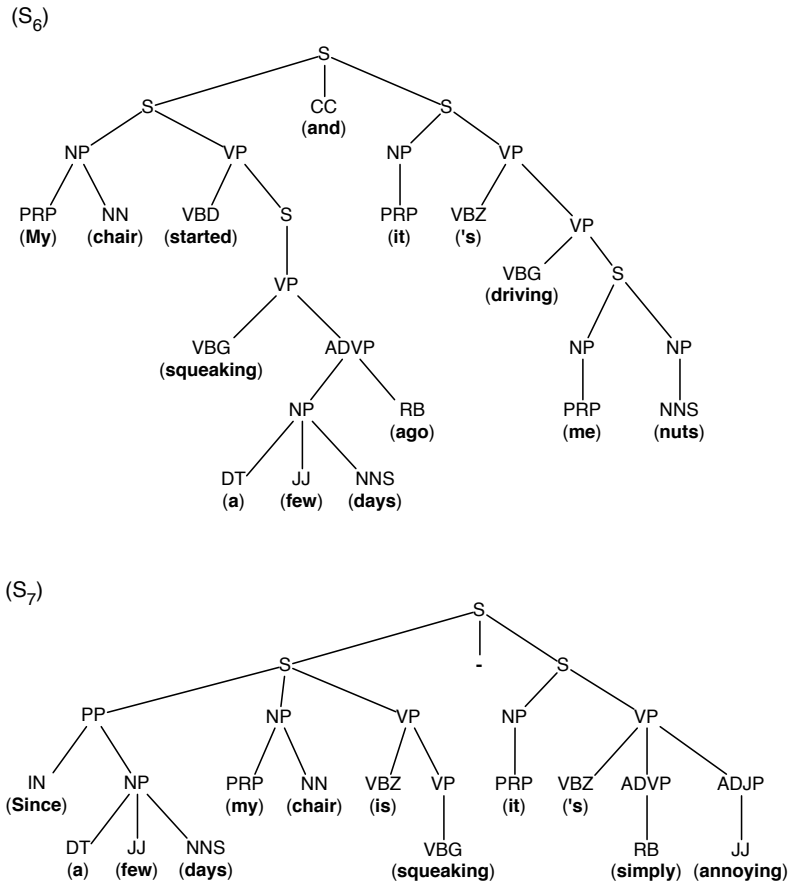


Figure 4.1.: Grammar Trees of Sentences (S_6) and (S_7).

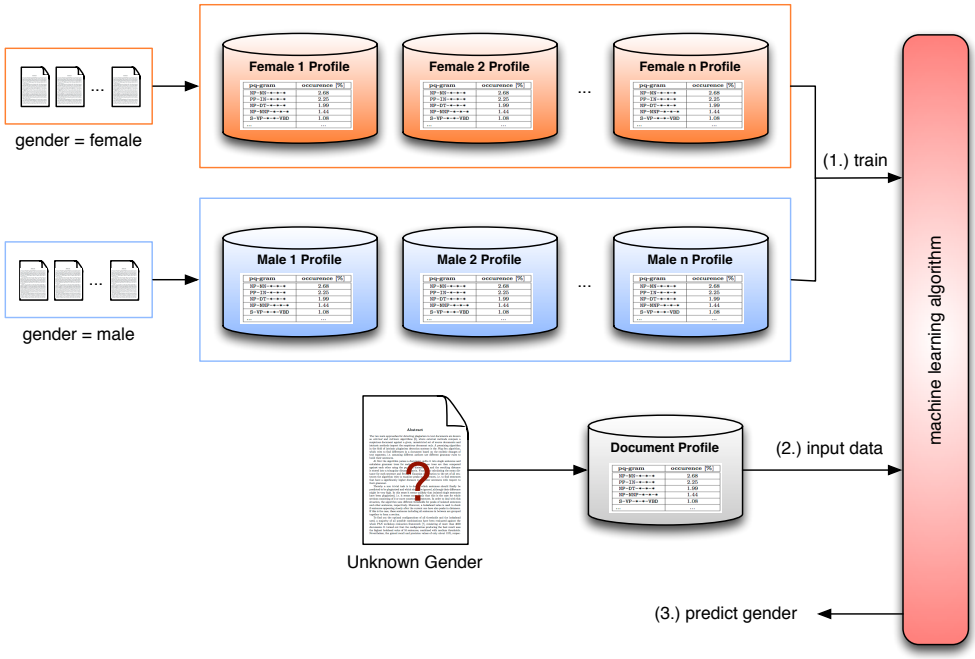


Figure 4.2.: Overview of the Author Profiling Process For Gender Classification.

Similarly to the PQ-Plag-Inn (Section 2.6) and the authorship attribution (Section 3.2) algorithms, pq-gram profiles have also been utilized in this approach. The profiles are again used as input for machine learning algorithms, which are trained in order to predict gender, age and a combination of both. The main contribution is to evaluate how reliable a prediction of an authors meta information is, when solely the pq-gram features are used.

A general overview of the profiling process is illustrated in Figure 4.2. It shows the basic components in the example of predicting the gender of authors using two respective document subsets, i.e., females and males, for training the classifiers. In case of age classification, three distinguishable age groups are used as described in Section 4.3. Finally, for estimating both gender and age the subsets are combined, which results in six distinct training subsets.

4.2.1. Algorithm

Basically the steps to profile a document is similar to the way it is done for authorship attribution with machine learning algorithms in this thesis:

1. At first the document is cleaned from unwanted characters and whitespaces, split by sentences and grammatically parsed. In case of ambiguity

of grammar trees, i.e., if there exist more than one valid parse tree for a sentence, the tree with the highest probability estimated by the parser is chosen.

2. Using the grammar trees of all sentences of the document, the pq-gram index is calculated. By combining all pq-gram indices of all sentences, a pq-gram profile is computed which contains a list of all pq-grams and their corresponding frequency of appearance in the text. The frequency is normalized by the total number of all appearing pq-grams. Additionally to the percentage of occurrence, also the rank of each pq-gram is stored in the profile as is described in the previous chapter (see Table 3.1).

3. Finally, the pq-gram profiles including occurrences and ranks are used as features which are applied to common machine learning algorithms that learn from the profiles and predict the gender and age, respectively.

4.2.2. Utilized Classifiers

To examine the profiling performance, the same classification algorithms that have been used in Chapter 3 have been utilized: Naive Bayes classifier [78], Bayes Network using the K2 classifier [34], Large Linear Classification using LibLinear [45], support vector machines using LibSVM with nu-SVC classification [29], k-nearest-neighbors classifier (kNN) [4] using $k = 1$ and J48, a pruned C4.5 decision tree [148].

4.2.3. Features

The features that have been used as input for the classifiers are extracted from the pq-gram profiles. As described earlier, each pq-gram together with its percentage of occurrence in the text represents a feature (`OCCURRENCE-RATE`), as well as the position of the pq-gram in the descending order by occurrence (`RANK`). A small example of a feature list including the correct gender and age classification is depicted in Table 4.1, whereby details about age groups are explained in Section 4.3. As used with authorship attribution, if a document does not contain a specific pq-gram, the feature value for that pq-gram as well as for the corresponding rank is set to -1.

Depending on the evaluation setup shown subsequently, the number of attributes to be handled by the classification algorithms range from 7,000 up to 20,000.

Feature	Document A	Document B	Document C
NP-NN-*-*-*	2.68	1.89	2.84
NP-NN-*-*-*--RANK	1	6	2
PP-IN-*-*-*	2.25	0.24	-1
PP-IN-*-*-*-RANK	2	153	-1
NP-DT-*-*-*	1.99	2.11	1.23
NP-DT-*-*-*-RANK	3	2	11
...	...	...	...
correct gender	male	female	male
correct age	20s	10s	30s

Table 4.1.: Example of a Feature List Serving as Input for Classification Algorithms.

4.3. Evaluation

Basically, the prediction of gender and age of the author of a text document is made by machine learning algorithms. Independent of the classifier used, the input consists of a large list of features with appropriate values and a corresponding classification class. The class is used to train the algorithms if the document is part of the training set, as well as for evaluating if the document is part of the test set.

4.3.1. Test Data And Experimental Setup

Test Data Set

The approach has been evaluated extensively using a frequently employed test set created by Schler et. al [156], containing thousands of freely accessible English web blogs. For this evaluation, a subset of approximately 8,000 randomly selected blogs has been used, whereby for each blog entry the gender as well as the age of the composer is given.

Regarding the latter, the ages are clustered into three distinct groups, as defined by the original test set [156]: 13-17 (=10s), 23-27 (=20s) and 33-42 (=30s).[3] Each age group is thereby separated by a five-year gap to gain higher

[3]The main reason that no older age groups are considered in the original data set is that at the time of its creation (2006) to few data existed for older ages, i.e., only 3% of the blog authors were older than 42 years. Nevertheless, the percentage of older social media users is increasing rapidly: where in 2008 only 18% of the 50-60 years olds where active on social media platforms, this number goes up to over 60% for 2014. Therefore older age groups should also be considered in future work. Sources http://www.jeffbullas.com/2011/09/02/20-stunning-social-media-statistics/ and http://www.jeffbullas.com/2014/01/17/20-social-media-facts-and-statistics-you-should-know-in-2014, visited July 2014

distinguishability. The corpus is fairly balanced with respect to gender, but has a majority in the 20s group and a minority in the 30s group. A detailed information about the class distribution is shown in Table 4.2. Because of the fact that simply predicting the majority class in all cases would lead to an accuracy of, e.g., 53% for `male`, the baseline which should be exceeded is set accordingly to 53% for gender, 46% for age and 25% for gender+age profiling, respectively.

	female	**male**	**sum**
10s	18%	19%	37%
20s	21%	25%	46%
30s	8%	9%	17%
Sum	47%	53%	*(100%)*

Table 4.2.: Test Data Distribution.

Each blog consists of at least 200 English words and has been textually cleaned in the original test data, i.e all unnecessary whitespace characters and HTML tags etc. have already been removed. Hyperlinks have been replaced by the word '`urlLink`'. Nonetheless, because this approach depends on the calculation of grammar trees, the latter tags have been manually removed for the evaluation, as the computation of grammar trees would be falsified.

Experimental Setup

The computation of the feature list is an essential part of the approach. Basically, it depends on the assignment of p and q, respectively, that is used for the extraction of pq-grams from sentences. For example, by using $p = 1$ and $q = 0$ the pq-grams would be reduced to single POS tags. Nevertheless, based on previous results, such configurations have been excluded as they led to insufficient results. The range of both stem and base of pq-grams has been evaluated in the range between 2 and 4, conforming to the size of n-grams that are used in efficient approaches in information retrieval (e.g. [169]).

An overview of the parameters used can be seen in Table 4.3. All possible settings, i.e., combinations of assignments of p and q with classifiers, have been evaluated on the test set using a 10-fold cross validation.

4.3.2. Profiling Results for Gender

The evaluation results for profiling the gender are listed in Table 4.4, whereby the three best results have are highlighted. The best result using $p = 2$ and $q = 3$ could be achieved with the support vector machine framework *LibSVM*, leading to an accuracy of 69%. It utilizes the occurrence-rate feature set,

Parameter	Assignment
p, q	2 – 4
s_{max}	200
pq_{max}	500
input feature set	`OCCURRENCE-RATE`, `RANK`, `ALL`

Table 4.3.: Parameter Setup Used for the Evaluation.

whereby males could be identified with 71%. Although the prediction rate is a little worse than those of other approaches (e.g. [156] achieves 80% over the full test set using several style and content features), the result is promising as it uses and evaluates only the proposed feature. The baseline of 53% could be surpassed clearly.

4.3.3. Profiling Results for Age

Using an almost identical setting as for gender classification, the maximum accuracy of 63% for age profiling results again from using *LibSVM* and the `OCCURRENCE-RATE` feature set (but with $p = 3$ instead of $p = 2$), as can be seen in Table 4.5. In general the accuracy for the prediction of the age groups `10s` and `20s` are very solid, but all classifiers have problems predicting the `30s` group. For example, the best configuration achieved a rate of 70% for `10s` and 68% for `20s`, respectively, but could only predict 5% correctly in the eldest group. While the other algorithms could profile the latter class at a higher accuracy, interestingly the Naive Bayes classifier even missed it totally.

A reason for the misclassification may be the unbalanced distribution of the test data, which contains only a small amount of `30s` text samples compared to the other groups. It might be the case that the classifiers would have needed more samples to construct a proper prediction model. Even though the unbalanced test set is an immediate consequence of the original test data distribution ([156]), future work should try to create a smaller, but equally distributed test set in order to examine the source of the problems occurring in the `30s` classification.

As with gender, the age results also significantly exceed the baseline of 43%. Like it can be assumed, by taking also other features into account, a higher accuracy can be achieved (e.g. [8] could reach 77% for age profiling).

			Occurrence-Rate			Rank			Combined			
Classifier	**p**	**q**	**female**	**male**	**w. avg**	**female**	**male**	**w. avg**	**female**	**male**	**w. avg**	**Max**
kNN	2	2	48.6	53.6	51.2	54.6	62.9	59.2	47.4	57.9	53.2	**59.2**
Naive Bayes	2	2	67.1	67.5	67.3	65.9	66.9	66.4	66.2	66.7	66.5	**67.3**
BayesNet	2	2	67.1	67.5	67.3	65.9	66.9	66.4	66.2	66.7	66.5	**67.3**
LibLinear	2	2	59.8	63.7	61.8	61.2	65.3	63.3	60.4	63.8	62.2	**63.3**
LibSVM	**2**	**2**	**61.5**	**66.3**	**64.0**	**66.5**	**71.1**	**69.0**	**62.5**	**67.8**	**65.3**	**69.0**
J48	2	2	57.5	62.6	60.2	53.7	60.6	57.4	54.8	61.7	58.5	**60.2**
kNN	2	3	54.2	61.3	58.1	52.8	61.1	57.4	54.2	61.9	58.4	**58.4**
Naive Bayes	2	3	66.9	67.5	67.2	67.3	67.9	67.6	67.4	68.0	67.7	**67.7**
BayesNet	2	3	66.9	67.5	67.2	67.3	67.9	67.6	67.4	68.0	67.7	**67.7**
LibLinear	2	3	56.1	61.2	58.8	60.5	64.8	62.8	58.0	63.3	60.8	**62.8**
LibSVM	2	3	60.2	65.3	62.9	65.0	70.2	67.8	60.7	66.3	63.7	**67.8**
J48	2	3	55.6	61.3	58.6	56.3	59.6	58.0	56.8	62.1	59.6	**59.6**
kNN	2	4	55.7	61.2	58.6	53.1	61.0	57.4	55.2	61.2	58.4	**58.6**
Naive Bayes	2	4	67.5	68.2	67.9	66.5	67.2	66.8	67.4	67.7	67.6	**67.9**
BayesNet	2	4	67.5	68.2	67.9	66.5	67.2	66.8	67.4	67.7	67.6	**67.9**
LibLinear	2	4	56.9	61.5	59.3	59.4	63.0	61.3	59.4	63.5	61.6	**61.6**
LibSVM	**2**	**4**	**60.7**	**64.7**	**62.8**	**66.3**	**69.9**	**68.2**	**62.2**	**66.7**	**64.6**	**68.2**
J48	2	4	54.3	59.3	56.9	55.4	61.8	58.9	55.8	62.8	59.6	**59.6**
kNN	3	2	54.0	62.5	58.7	54.2	61.3	58.1	52.3	61.7	57.5	**58.7**
Naive Bayes	3	2	67.2	68.2	67.7	66.3	67.2	66.7	66.6	67.4	67.0	**67.7**
BayesNet	3	2	67.2	68.2	67.7	66.2	67.1	66.7	66.6	67.4	67.0	**67.7**
LibLinear	3	2	59.2	64.2	61.8	61.4	65.7	63.7	60.0	64.4	62.4	**63.7**
LibSVM	**3**	**2**	**62.2**	**67.6**	**65.1**	**66.3**	**71.1**	**68.8**	**63.3**	**68.0**	**65.8**	**68.8**
J48	3	2	56.7	61.3	59.2	56.3	60.2	58.3	53.9	59.8	57.0	**59.2**
kNN	3	3	51.4	59.3	55.7	55.3	62.1	59.0	51.3	61.3	56.9	**59.0**
Naive Bayes	3	3	67.2	68.1	67.6	66.2	67.1	66.7	67.1	68.0	67.5	**67.6**
BayesNet	3	3	67.1	68.1	67.6	66.2	67.1	66.7	67.1	68.0	67.5	**67.6**
LibLinear	3	3	56.5	61.5	59.2	60.4	64.6	62.6	58.8	63.1	61.0	**62.6**
LibSVM	3	3	60.7	64.6	62.8	64.5	69.0	66.9	61.8	66.2	64.1	**66.9**
J48	3	3	55.8	61.3	58.7	57.5	62.1	60.0	53.9	59.3	56.8	**60.0**
kNN	3	4	50.8	59.8	55.8	54.1	60.3	57.4	52.2	61.0	57.0	**57.4**
Naive Bayes	3	4	67.4	67.8	67.6	66.4	66.8	66.6	67.1	67.3	67.2	**67.6**
BayesNet	3	4	67.4	67.8	67.6	66.3	66.7	66.5	67.1	67.3	67.2	**67.6**
LibLinear	3	4	56.2	60.5	58.5	59.8	63.7	61.8	57.6	62.1	60.0	**61.8**
LibSVM	3	4	60.9	65.9	63.6	65.0	69.5	67.4	62.1	67.0	64.7	**67.4**
J48	3	4	58.4	63.5	61.1	55.6	61.9	59.0	57.2	62.5	60.1	**61.1**
kNN	4	2	48.5	60.9	55.5	54.9	60.0	57.6	49.0	61.4	56.1	**57.6**
Naive Bayes	4	2	67.0	67.6	67.3	66.4	67.2	66.8	66.6	67.1	66.8	**67.3**
BayesNet	4	2	67.0	67.6	67.3	66.3	67.1	66.7	66.6	67.1	66.8	**67.3**
LibLinear	4	2	58.3	63.8	61.2	59.3	64.3	62.0	59.8	64.7	62.4	**62.4**
LibSVM	4	2	62.7	67.3	65.1	65.2	69.5	67.5	63.8	67.9	66.0	**67.5**
J48	4	2	58.5	62.4	60.5	55.2	60.8	58.2	56.5	61.8	59.3	**60.5**
kNN	4	3	51.1	60.1	56.1	55.5	60.3	58.0	51.4	60.3	56.3	**58.0**
Naive Bayes	4	3	66.2	66.8	66.5	66.2	66.7	66.5	66.5	66.9	66.7	**66.7**
BayesNet	4	3	66.2	66.8	66.5	66.2	66.7	66.4	66.5	66.9	66.7	**66.7**
LibLinear	4	3	55.2	60.4	57.9	60.0	64.5	62.4	57.1	62.3	59.9	**62.4**
LibSVM	4	3	59.8	64.0	62.0	63.7	68.8	66.4	60.6	65.6	63.3	**66.4**
J48	4	3	59.3	63.5	61.5	55.2	60.3	57.9	55.9	62.2	59.3	**61.5**
kNN	4	4	54.6	62.6	59.0	55.2	60.9	58.2	52.9	62.7	58.3	**59.0**
Naive Bayes	4	4	66.4	67.2	66.8	65.9	66.4	66.2	65.8	66.1	66.0	**66.8**
BayesNet	4	4	66.3	67.1	66.7	65.8	66.4	66.1	65.8	66.1	66.0	**66.7**
LibLinear	4	4	54.3	59.7	57.2	59.3	64.5	62.1	55.6	61.3	58.6	**62.1**
LibSVM	4	4	59.8	64.9	62.5	63.3	68.8	66.3	61.2	66.5	64.0	**66.3**
J48	4	4	56.6	60.7	58.7	55.4	60.8	58.3	54.3	60.4	57.5	**58.7**

Table 4.4.: Evaluation Results in Percent For Gender Profiling.

Classifier	p	q	Occurrence-Rate				Rank				Combined				Max
			10s	20s	30s	w. avg	10s	20s	30s	w. avg	10s	20s	30s	w. avg	
kNN	2	2	41.0	56.6	25.7	47.0	56.1	56.0	29.4	51.6	46.9	55.6	25.4	47.8	**51.6**
Naive Bayes	2	2	66.0	47.8	38.5	52.4	66.4	48.8	38.7	52.9	66.0	48.2	38.7	52.4	**52.9**
BayesNet	2	2	65.9	47.4	38.6	52.2	66.2	48.4	38.4	52.6	66.2	47.9	38.5	52.4	**52.6**
LibLinear	2	2	58.7	55.0	25.0	51.5	65.9	58.5	29.6	56.6	61.6	55.4	30.4	53.6	**56.6**
LibSVM	**2**	**2**	**64.2**	**60.8**	**15.6**	**56.7**	**68.7**	**68.1**	**6.4**	**62.8**	**66.8**	**64.0**	**13.1**	**59.7**	**62.8**
J48	2	2	55.1	52.4	24.2	48.9	53.0	51.0	27.6	47.8	54.0	53.0	25.3	48.7	**48.9**
kNN	2	3	50.0	54.8	23.9	48.2	56.7	56.3	30.7	52.3	53.2	56.6	23.8	50.2	**52.3**
Naive Bayes	2	3	66.3	47.7	40.0	53.0	66.6	47.6	38.4	52.6	66.7	46.5	38.9	52.4	**53.0**
BayesNet	2	3	66.3	47.7	40.0	53.0	66.6	47.7	38.6	52.6	66.6	46.3	39.2	52.3	**53.0**
LibLinear	2	3	58.9	54.5	29.3	52.0	65.4	60.3	29.5	57.4	62.6	55.6	28.9	53.9	**57.4**
LibSVM	2	3	63.5	62.7	18.5	57.9	68.1	67.8	5.0	62.2	64.8	64.9	15.7	59.5	**62.2**
J48	2	3	54.8	50.7	23.5	47.8	53.2	51.4	25.3	47.8	55.3	52.8	24.5	49.3	**49.3**
kNN	2	4	52.9	55.1	24.2	49.3	56.2	55.9	26.2	51.1	52.0	54.6	23.9	48.7	**51.1**
Naive Bayes	2	4	66.8	48.1	39.8	53.2	67.6	48.1	39.7	53.5	67.5	47.6	40.6	53.5	**53.5**
BayesNet	2	4	66.7	48.0	39.7	53.1	67.6	48.1	39.7	53.5	67.5	47.5	40.3	53.4	**53.5**
LibLinear	2	4	60.3	55.5	27.4	52.9	64.9	57.9	25.6	55.7	62.6	57.3	27.5	54.5	**55.7**
LibSVM	**2**	**4**	**64.5**	**64.2**	**22.1**	**59.5**	**69.4**	**68.2**	**4.4**	**62.8**	**66.9**	**65.4**	**15.4**	**60.5**	**62.8**
J48	2	4	55.9	53.6	26.5	50.2	53.2	52.3	24.6	48.3	57.3	52.1	24.8	49.6	**50.2**
kNN	3	2	52.5	54.2	23.2	48.3	55.9	55.9	26.2	51.3	51.8	54.6	23.0	48.4	**51.3**
Naive Bayes	3	2	66.8	46.2	38.9	52.4	67.6	46.0	38.5	52.4	67.1	45.4	39.2	52.2	**52.4**
BayesNet	3	2	66.8	46.4	39.3	52.6	67.6	46.0	38.5	52.4	67.1	45.9	39.5	52.4	**52.6**
LibLinear	3	2	58.8	53.2	27.9	51.0	64.4	57.5	26.0	55.0	61.3	55.1	28.6	52.9	**55.0**
LibSVM	**3**	**2**	**67.0**	**66.1**	**19.9**	**61.1**	**70.1**	**68.4**	**5.0**	**63.2**	**68.2**	**67.5**	**18.0**	**62.4**	**63.2**
J48	3	2	51.9	53.9	29.0	49.0	55.3	50.9	21.7	47.7	52.8	50.7	23.8	47.2	**49.0**
kNN	3	3	51.2	56.8	27.2	49.8	54.4	53.3	25.1	48.9	53.5	56.8	26.5	50.5	**50.5**
Naive Bayes	3	3	67.3	47.0	40.1	53.1	67.5	45.5	39.5	52.5	67.2	44.1	39.1	51.7	**53.1**
BayesNet	3	3	67.3	47.2	39.8	53.1	67.6	45.5	39.3	52.5	67.2	44.4	39.3	51.9	**53.1**
LibLinear	3	3	62.1	55.7	26.7	53.2	63.0	58.6	25.7	55.1	64.5	57.5	26.1	55.1	**55.1**
LibSVM	3	3	68.3	65.1	17.6	60.7	68.5	67.6	5.0	62.2	68.9	66.0	13.3	61.3	**62.2**
J48	3	3	53.6	52.8	20.1	48.1	54.4	52.2	22.0	48.1	52.9	52.5	22.7	48.0	**48.1**
kNN	3	4	52.8	54.0	26.3	48.6	58.7	53.1	28.4	50.5	56.5	54.3	24.1	49.5	**50.5**
Naive Bayes	3	4	67.3	45.8	39.2	52.4	67.0	43.6	39.1	51.5	67.4	43.3	38.9	51.5	**52.4**
BayesNet	3	4	67.4	45.6	39.1	52.4	66.9	43.8	39.3	51.6	67.3	43.0	38.9	51.3	**52.4**
LibLinear	3	4	60.7	55.3	28.2	53.0	64.1	57.6	28.2	55.4	62.4	55.4	28.1	53.6	**55.4**
LibSVM	3	4	67.3	64.3	17.8	60.2	67.5	66.9	5.0	61.4	68.7	66.2	16.1	61.7	**61.7**
J48	3	4	52.3	54.3	27.2	49.2	56.1	53.8	26.3	50.2	55.8	53.8	24.3	49.6	**50.2**
kNN	4	2	49.8	53.0	25.2	47.2	54.6	56.1	20.4	50.4	51.6	53.5	23.9	47.9	**50.4**
Naive Bayes	4	2	67.3	45.3	38.2	52.0	67.3	45.9	38.8	52.3	67.9	45.2	39.1	52.4	**52.4**
BayesNet	4	2	67.3	45.6	38.7	52.3	67.4	46.0	39.1	52.4	67.9	44.5	38.7	52.0	**52.4**
LibLinear	4	2	59.3	54.7	28.2	52.2	62.4	58.7	26.2	55.0	60.3	55.5	28.1	52.9	**55.0**
LibSVM	4	2	64.9	64.2	19.8	59.4	67.7	68.1	4.4	62.2	67.8	66.6	14.7	61.6	**62.2**
J48	4	2	51.6	50.8	23.3	46.5	52.4	50.9	20.6	46.5	53.6	52.1	20.9	47.6	**47.6**
kNN	4	3	51.8	54.8	25.0	48.6	52.7	54.5	22.5	48.8	53.4	55.5	25.4	49.6	**49.6**
Naive Bayes	4	3	67.0	45.4	38.7	52.1	67.8	44.2	39.1	52.0	67.7	44.1	38.9	51.9	**52.1**
BayesNet	4	3	67.0	45.2	38.9	52.1	67.8	44.2	39.1	52.0	67.7	44.2	39.0	51.9	**52.1**
LibLinear	4	3	58.2	54.3	25.3	51.1	62.2	60.0	27.4	55.9	61.5	56.0	26.6	53.4	**55.9**
LibSVM	4	3	65.1	63.4	14.4	58.6	67.0	67.0	2.6	61.2	66.2	65.3	12.3	60.0	**61.2**
J48	4	3	54.5	52.3	22.5	48.1	53.1	52.2	23.6	48.0	52.3	50.3	27.1	47.2	**48.1**
kNN	4	4	50.7	53.4	27.3	47.8	53.1	54.7	24.0	49.3	52.4	54.7	27.4	49.2	**49.3**
Naive Bayes	4	4	67.2	45.6	38.5	52.3	67.3	43.8	38.7	51.5	67.1	43.6	38.4	51.4	**52.3**
BayesNet	4	4	67.2	45.4	38.6	52.2	67.3	43.7	38.6	51.5	67.2	43.3	38.3	51.3	**52.2**
LibLinear	4	4	59.8	53.7	29.2	52.0	63.4	59.8	30.1	56.6	61.8	56.5	29.5	54.2	**56.6**
LibSVM	4	4	65.8	64.2	16.5	59.5	67.1	67.4	3.2	61.6	67.9	66.5	13.6	61.4	**61.6**
J48	4	4	54.6	52.5	21.3	48.3	56.3	52.5	23.1	49.2	55.3	55.3	21.9	50.0	**50.0**

Table 4.5.: Evaluation Results in Percent For Age Profiling.

4.3.4. Profiling Results for Gender And Age

The evaluation results for the combined gender and age profiling problem are shown in Table 4.6. Here, the combinations of gender and age, i.e., six classes, had to be predicted. The baseline coming from the majority class `male-20s` is 25% and could also be surpassed using the *LibLinear* classifier. With relatively large structure fragments resulting from the assignments $p = 4$ and $q = 3$, an accuracy of 39% could be achieved using the `RANK` feature set.

Due to visibility reasons the details for the individual sub results have been omitted in the table. Nonetheless the experimental data shows that the combined gender and age classification also suffers from predicting the male/female classes of the `30s` age group correctly.

4.3.5. Confusion Matrices

A detailed analysis of the best working classifications is shown in the confusion matrices in Table 4.7. When predicting the gender, the number of false-positives for `male` as well as for `female` are approximately the same. On the other side, the classification of age groups had massive problems concerning the `30s` group, where only 0.5% have been labeled correctly. The majority of this group has been predicted as `20s`, which represents also the majority group of the test data.

As already mentioned, a possible explanation might be the unbalanced test set. This is reinforced by the fact that mostly all false-positives of the `10s` group have also been labeled as `20s`. But what also seems plausible is the hypothesis that the grammar of 13-17 (`10s`) year olds differs significantly from that of 23-27 (`20s`) year olds, where on the other hand the grammatical style of the latter is similar to 33-42 (`30s`) year olds. Intuitively this seems reasonable when looking at sample documents, but future work should investigate further to verify or falsify this assumption.

A visualization of the confusion matrices is depicted in Figure 4.3. Thereby the values have been normalized per class, i.e., for each row. It can be seen in diagram (a) that the gender plot perfectly emphasizes the diagonal, which means that the majority of each gender group could be profiled properly: 72% of the males and 66% of the females could be attributed correctly. Diagram (b) shows the confusion matrix for age, which denotes good results for the `10s` age group with an accuracy of 68% and the `20s` groups with an accuracy of even 81%. Nevertheless and as mentioned before, the problem with the `30s` group can also be seen clearly where only 3% could be classified correctly and the majority of 87% is falsely attributed as `20s`. Finally, diagram (c) illustrates the percentages for gender including age profiling. Here, also acceptable

Classifier	p	q	Occurrence-Rate	Rank	All	Max
Naive Bayes	2	2	30.2	30.3	30.3	**30.3**
BayesNet	2	2	36.3	35.3	35.2	**36.3**
LibLinear	2	2	36.1	32.6	32.6	**36.1**
LibSVM	2	2	25.4	25.4	25.4	**25.4**
kNN	2	2	32.6	25.6	25.5	**32.6**
J48	2	2	27.9	27.9	28.2	**28.2**
Naive Bayes	2	3	31.7	29.5	31.6	**31.7**
BayesNet	2	3	35.5	36.2	35.5	**36.2**
LibLinear	**2**	**3**	**37.8**	**31.6**	**32.9**	**37.8**
LibSVM	2	3	25.4	25.4	32.9	**32.9**
kNN	2	3	30.2	29.4	30.9	**30.9**
J48	2	3	29.1	26.9	31.0	**31.0**
Naive Bayes	2	4	31.5	28.3	30.9	**31.5**
BayesNet	2	4	36.1	36.4	36.0	**36.4**
LibLinear	2	4	34.3	31.2	33.6	**34.3**
LibSVM	2	4	25.4	25.4	25.4	**25.4**
kNN	2	4	30.1	29.8	30.0	**30.1**
J48	2	4	29.0	27.6	28.6	**29.0**
Naive Bayes	3	2	32.0	31.3	31.3	**32.0**
BayesNet	3	2	35.3	35.8	35.8	**35.8**
LibLinear	3	2	35.7	34.7	34.7	**35.7**
LibSVM	3	2	25.4	25.4	25.4	**25.4**
kNN	3	2	30.6	27.3	27.3	**30.6**
J48	3	2	29.9	28.2	28.2	**29.9**
Naive Bayes	3	3	31.1	28.9	30.2	**31.1**
BayesNet	3	3	35.6	35.2	34.7	**35.6**
LibLinear	3	3	33.9	31.3	33.1	**33.9**
LibSVM	3	3	25.4	25.4	25.4	**25.4**
kNN	3	3	29.3	28.6	29.0	**29.3**
J48	3	3	31.1	28.2	27.1	**31.1**
Naive Bayes	3	4	31.8	29.7	31.8	**31.8**
BayesNet	3	4	34.7	35.8	34.7	**35.8**
LibLinear	3	4	34.8	31.7	33.5	**34.8**
LibSVM	3	4	25.4	25.4	25.4	**25.4**
kNN	3	4	28.2	27.8	26.8	**28.2**
J48	3	4	29.0	28.0	28.3	**29.0**
Naive Bayes	4	2	34.8	35.7	35.1	**35.7**
BayesNet	4	2	34.9	35.7	34.8	**35.7**
LibLinear	4	2	33.8	32.1	33.7	**33.8**
LibSVM	**4**	**2**	**37.2**	**34.5**	**25.8**	**37.2**
kNN	4	2	28.9	27.1	28.0	**28.9**
J48	4	2	29.5	24.6	27.5	**29.5**
Naive Bayes	4	3	35.0	35.1	34.5	**35.1**
BayesNet	4	3	34.8	34.9	34.7	**34.9**
LibLinear	**4**	**3**	**33.9**	**39.1**	**30.9**	**39.1**
LibSVM	4	3	36.2	32.6	35.6	**36.2**
kNN	4	3	27.3	27.4	28.1	**28.1**
J48	4	3	27.5	26.7	27.9	**27.9**
Naive Bayes	4	4	35.7	35.5	35.2	**35.7**
BayesNet	4	4	35.5	35.6	35.6	**35.6**
LibLinear	4	4	33.2	31.3	33.3	**33.3**
LibSVM	4	4	25.4	34.0	34.7	**34.7**
kNN	4	4	27.5	29.5	30.4	**30.4**
J48	4	4	26.6	27.7	28.2	**28.2**

Table 4.6.: Evaluation Results in Percent For Gender and Age Profiling.

	classified as [%]	
	female	**male**
female	**30.8**	16.1
male	15.0	**38.1**

(a) Gender

	classified as [%]		
	10s	**20s**	**30s**
10s	**25.3**	11.6	0.4
20s	7.8	**37.5**	0.9
30s	1.7	14.3	**0.5**

(b) Age

	classified as [%]					
	female-10s	**female-20s**	**female-30s**	**male-10s**	**male-20s**	**male-30s**
female-10s	**9.8**	3.1	0.2	3.5	1.3	0.1
female-20s	3.3	**7.8**	1.1	2.1	6.0	0.5
female-30s	0.6	2.8	**1.0**	0.5	2.7	0.6
male-10s	5.2	2.1	0.4	**6.4**	4.6	0.7
male-20s	1.0	4.7	0.9	3.7	**13.3**	1.7
male-30s	0.1	1.0	0.5	0.9	5.1	**0.7**

(c) Gender And Age

Table 4.7.: Confusion Matrices of the Best Results For Gender, Age and Gender Including Age Profiling.

attributions could be done for all classes not containing the `30s` age group. On the other hand, the remaining classes consequently also have been profiled mainly incorrectly.

4.4. Conclusion and Future Work

Conclusion

In this chapter, the previously described grammar profiles have been adapted in order to automatically profile the author of a text document. Based on the idea presented in Chapter 3, substructures of parse trees are utilized by using pq-grams, and machine learning algorithms are finally applied on pq-gram profiles to learn and predict the gender and age of the originator.

An extensive evaluation using a state-of-the art test set shows that pq-grams can be used as significant features in text classification, whereby gender and age can be predicted with an accuracy of 69% and 63%, respectively. With respect to the fact that the experiment in this paper solely uses the presented feature, the results are promising.

Despite of the class to predict, the support vector machine framework *LibSVM* and the large linear classification *LibLinear* worked best, whereas the kNN classifier and the C4.5 decision tree produced worse results. Also, the combined feature set using the frequencies of occurrences together with the ranks is

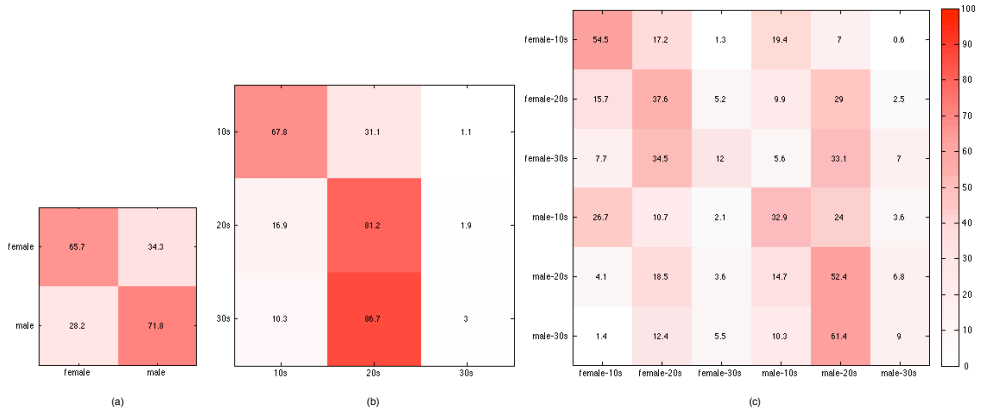


Figure 4.3.: Profiling: Confusion Matrices in Percent and Normalized per Class.

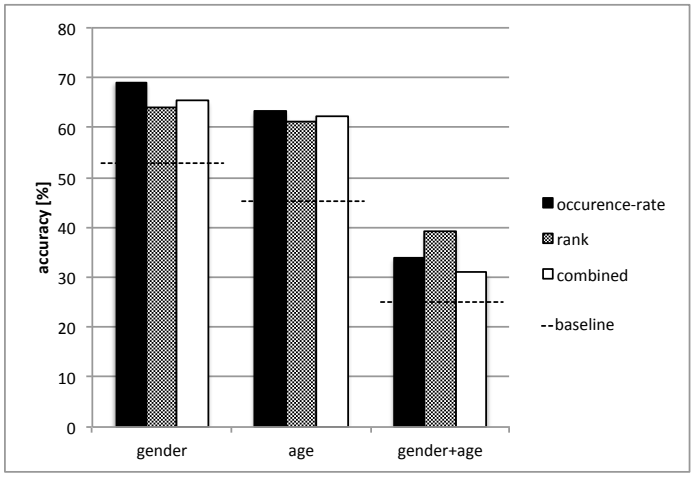


Figure 4.4.: Profiling: Evaluation Results Using Different Feature Sets.

always inferior to the isolated subsets, which may possibly be correlated to the large amount of features employed with this set (double the size of the other sets).

Summarizing, Figure 4.4 illustrates the evaluation results for all three classification problems using the different feature sets. In general, the results could significantly exceed the corresponding baselines, which represent the prediction rate if simply the majority classes are selected in all cases. The result manifests that solely the grammar of authors - analyzed with syntax trees and pq-grams - serves as a distinct feature for author profiling as well.

Future Work

In general the results are promising, especially when taking into account that only one feature type has been used, whereas other approaches make use of up to hundreds of feature types. Nevertheless evaluation results showed that the approach has problems predicting the `30s` age group of the provided test data set. Although hypothesis explaining the problem have been stated, they should be verified or falsified in detail by utilizing different, heterogeneous test sets.

In order to build a reliable text classification approach, the grammar feature should be combined with other commonly used style and content feature sets in future work. Besides the utilization of common lexical, syntactic or complexity features, the usage of vocabulary or neologisms should be considered, especially when analyzing online content. Moreover it should be investigated whether the proposed feature is also applicable to shorter text samples such as chat logs or even single-line Twitter postings.

Finally and as with all algorithms described in this thesis, research should also examine whether the approach is also exploitable to other languages. In particular the investigation of achievable profiling accuracies of online content written in different languages could be interesting, as demographic features may also come into play that have to considered.

CHAPTER 5

Decomposition of Multi-Author Documents[1]

5.1. Introduction

Approaches which attempt to divide a single text document into distinguishable units like different topics, for example, are usually referred to as *text segmentation* approaches (e.g. [30], [117]). Here, also many features including statistical models, similarities between words or other semantic analyses are used. Moreover, text clusters are also used in recent plagiarism detection algorithms (e.g. [207]) which try to build a cluster for the main author and one or more clusters for intrusive paragraphs. Another scenario where the clustering of text is applicable is the analysis of multi-author academic papers: especially the verification of collaborated student works such as bachelor or master theses can be useful in order to determine the amount of work done by each student.

[1] This chapter is based on and contentual partly reused from the paper: M. Tschuggnall and G. Specht. *Automatic Decomposition of Multi-Author Documents Using Grammar Analysis.* In Proceedings of the 26th GI-Workshop on Grundlagen von Datenbanken (GvD), Bozen, Italy, October 2014. [189]

The intention of the work presented in this chapter is to evaluate the pq-gram feature sets in combination with current clustering algorithms, i.e., to verify if the grammar of authors - represented by pq-grams - can also be used to intelligently build clusters. To do this, the ideas of the previous chapters have been adapted und evaluated on specifically created test data sets.

The rest of this chapter is organized as follows: Section 5.2 recapitulates the basic algorithm, which is then evaluated in Section 5.3 by using different clustering algorithms and data sets. A comparison of the clustering and classification approaches is discussed in Section 5.4, and finally a conclusion and future work directions are given in Section 5.5.

5.2. Algorithm

The main idea is to utilize common state-of-the-art clustering algorithms, which build clusters of text documents or paragraphs based on pq-gram features sets. Given a document, the algorithm basically consists of the following steps:

1. The document is preprocessed by eliminating whitespaces and other non-alphanumeric characters, and then it is partitioned into single paragraphs.

2. Each paragraph is then split into single sentences, which are then parsed and from which pq-grams are extracted subsequently.

3. Finally, for each paragraph a pq-gram profile is calculated which contains all occurring pq-grams and their respective percentages. Each paragraph-profile is then provided as input for clustering algorithms, which try to build clusters based on the pq-grams. Concretely, the feature sets `OCCURRENCE-RATE`, `RANK` and `ALL` have been used and evaluated separately like it has been done in the previous chapters.

Utilized Clustering Algorithms

Using the WEKA framework [67], the following clustering algorithms have been evaluated:

- K-Means [11]

- Cascaded K-Means (the number of clusters is cascaded and automatically chosen) [26]

- X-Means [138]
- Agglomerative Hierarchical Clustering [121]
- Farthest First [36]

For the clustering algorithms *K-Means*, *Hierarchical Clustering* and *Farthest First* the number of clusters has been predefined according to the respective test data. This means if the test document has been collaborated by three authors, the number of clusters has also been set to three. On the other hand, the algorithms *Cascaded K-Means* and *X-Means* implicitly decide which amount of clusters is optimal. Therefore these algorithms have been limited only in ranges, i.e., the minimum and maximum number of clusters has been set to two and six, respectively.

5.3. Evaluation

The utilization of pq-gram profiles as input features for modern clustering algorithms has been extensively evaluated using different documents and data sets. As the clustering and classification problems are closely related, the global aim was to experiment on the accuracy of automatic text clustering using solely the proposed grammar feature, and furthermore to compare it to those of current classification techniques.

5.3.1. Test Data and Experimental Setup

In order to evaluate the idea, different documents and test data sets have been used, which are explained in more detail in the following. Thereby single documents have been created which contain paragraphs written by different authors, as well as multiple documents, whereby each document is written by one author. In the latter case, every document is treated as one (large) paragraph for simplification reasons.

As before, different parameter settings have been evaluated, i.e., the pq-gram values p and q have been varied from 2 to 4, in combination with the feature sets `OCCURRENCE-RATE`, `RANK` and `ALL`.

Twain-Wells

This document has been specifically created for the evaluation of in-document clustering. It contains 50 paragraphs of the book *"The Adventures of Huckleberry Finn"* by Mark Twain, and 50 paragraphs of *"The Time Machine"* by

H. G. Wells[2]. All paragraphs have been randomly shuffled, whereby the size of each paragraph varies from approximately 25 words up to 280 words.

Twain-Wells-Shelley

In a similar fashion a three-author document has been created. It again uses (different) paragraphs of the same books by Twain and Wells, and appends it by paragraphs of the book *"Frankenstein; Or, The Modern Prometheus"* by Mary Wollstonecraft Shelley. Summarizing, the document contains 50 paragraphs by Mark Twain, 50 paragraphs by H. G. Wells and another 50 paragraphs by Mary Shelley, whereby the paragraph sizes are similar to the Twain-Wells document.

The Federalist Papers and the PAN12 data set

To be able to compare the clustering algorithms to the classification algorithms, the data sets used for authorship attribution (see Chapter 3) have been reused. As stated before, each document is thereby treated as one paragraph, written by the respective author. In contrast to the attribution data sets all available documents have been included as the clustering process does not need training data. Also, from the PAN12 data set only the subproblems A and B have been used, as the other problems contain only one sample per author and thus the calculation of clusters would be meaningless.

In total the Federalist Papers contain 70 documents (paragraphs) written by three authors, and both subsets of PAN12 contain 14 documents (paragraphs) written by three authors and distributed equally.

5.3.2. Results

The best results of the clustering evaluation are presented in Table 5.1, where the best performance for each clusterer over all data sets is shown in subtable (a), and the best performance for each data set is shown in subtable (b), respectively. With an accuracy of 63.7% the K-Means algorithm worked best by using $p = 2, q = 3$ and by utilizing all available features. Interestingly, the X-Means algorithm also achieved good results considering the fact that in this case the number of clusters has been assigned automatically by the algorithm. Finally, the hierarchical cluster performed worst gaining an accuracy of nearly 10% less than K-Means.

Regarding the best performances for each test data set, the results for the manually created data sets from novel literature are generally poor. For ex-

[2]The books have been obtained from the Project Gutenberg library [180]

ample, the best result for the two-author document Twain-Wells is only 59.6%, i.e., the accuracy is only slightly better than the baseline percentage of 50%, which can be achieved by randomly assigning paragraphs into two clusters.[3] On the other hand, the data sets reused from authorship attribution, namely the Federalist Papers and the PAN12 data set, achieved very good results with an accuracy of about 89% and 83%, respectively. Nevertheless, as the other data sets have been specifically created for the clustering evaluation, these results may be more expressive. Therefore a comparison between clustering and classification approaches is discussed in Section 5.4, showing that the latter achieve significantly better results on those data sets when using the same features.

Method	**p**	**q**	**Feature Set**	**Accuracy**
K-Means	3	2	All	**63.7**
X-Means	2	4	RANK	**61.7**
Farthest First	4	2	Occurrence-Rate	**58.7**
Cascaded K-Means	2	2	RANK	**55.3**
Hierarchical Clusterer	4	3	Occurrence-Rate	**54.7**

(a) Clustering Algorithms

Data Set	**Method**	**p**	**q**	**Feature Set**	**Accuracy**
Twain-Wells	X-Means	3	2	All	**59.6**
Twain-Wells-Shelley	X-Means	3	4	All	**49.0**
FED	Farthest First	4	3	RANK	**89.4**
PAN12-A/B	K-Means	3	3	All	**83.3**

(b) Test Data Sets

Table 5.1.: Best Evaluation Results for each Clustering Algorithm and Test Data Set in Percent.

Detailed results are shown in Table 5.2 for every data set and the feature sets `OCCURRENCE-RATE` (OR), `RANK` (RK) and the combined set. The three best results are highlighted.

5.4. Comparison of Clustering and Classification Approaches

For the given data sets, any clustering problem can be rewritten as classification problem with the exception that the latter need training data. Although a direct comparison should be treated with caution, it still gives an insight of

[3] In this case X-Means dynamically created two clusters, but the result is still better than that of other algorithms using a fixed number of clusters.

			Twain-Wells			Twain-Wells-Shelly			Federalist Papers			PAN12-A/B			Average			
Clusterer	**p**	**q**	**OR**	**RK**	**ALL**	**OR**	**RK**	**ALL**	**OR**	**RK**	**ALL**	**OR**	**RK**	**ALL**	**OR**	**RK**	**ALL**	**Max**
K-Means	2	2	51.5	51.5	51.5	44.3	36.9	39.6	54.5	74.2	65.2	83.3	83.3	83.3	**58.4**	**61.5**	**59.9**	**61.5**
Hierarchical Clusterer	2	2	50.5	50.5	50.5	34.2	34.2	34.2	75.8	74.2	75.8	50.0	50.0	50.0	**52.6**	**52.2**	**52.6**	**52.6**
Farthest First	2	2	50.5	50.5	50.5	34.9	34.9	34.9	74.2	77.3	75.8	58.3	58.3	58.3	**54.5**	**55.3**	**54.9**	**55.3**
X-Means	2	2	57.6	57.6	57.6	34.2	36.9	34.2	48.5	68.2	51.5	58.3	58.3	58.3	**49.7**	**55.3**	**50.4**	**55.3**
Cascaded K-Means	2	2	57.6	57.6	57.6	34.2	36.9	34.2	48.5	68.2	51.5	58.3	58.3	58.3	**49.7**	**55.3**	**50.4**	**55.3**
K-Means	2	3	51.5	51.5	51.5	38.3	37.6	38.3	68.2	69.7	72.7	83.3	75.0	83.3	**60.3**	**58.4**	**61.5**	**61.5**
Hierarchical Clusterer	2	3	50.5	50.5	50.5	34.2	34.2	34.2	75.8	75.8	75.8	50.0	50.0	50.0	**52.6**	**52.6**	**52.6**	**52.6**
Farthest First	2	3	50.5	50.5	50.5	34.2	34.2	34.2	78.8	77.3	77.3	58.3	58.3	58.3	**55.5**	**55.1**	**55.1**	**55.5**
X-Means	2	3	56.6	56.6	56.6	38.3	36.9	35.6	74.2	72.7	72.7	58.3	58.3	58.3	**56.8**	**56.1**	**55.8**	**56.8**
Cascaded K-Means	2	3	56.6	56.6	56.6	38.3	36.9	35.6	74.2	72.7	72.7	58.3	58.3	58.3	**56.8**	**56.1**	**55.8**	**56.8**
K-Means	**2**	**4**	**52.5**	**52.5**	**52.5**	**40.9**	**38.9**	**40.3**	**69.7**	**75.8**	**75.8**	**83.3**	**83.3**	**83.3**	**61.6**	**62.6**	**63.0**	**63.0**
Hierarchical Clusterer	2	4	50.5	50.5	50.5	34.2	34.2	34.2	75.8	74.2	75.8	50.0	50.0	50.0	**52.6**	**52.2**	**52.6**	**52.6**
Farthest First	2	4	50.5	50.5	50.5	34.2	34.2	34.2	72.7	71.2	77.3	58.3	58.3	58.3	**53.9**	**53.6**	**55.1**	**55.1**
X-Means	2	4	57.6	57.6	57.6	45.6	35.6	38.9	68.2	78.8	69.7	66.7	75.0	75.0	**59.5**	**61.7**	**60.3**	**61.7**
Cascaded K-Means	2	4	57.6	57.6	57.6	45.6	35.6	38.9	68.2	78.8	69.7	66.7	75.0	75.0	**59.5**	**61.7**	**60.3**	**61.7**
K-Means	**3**	**2**	**53.5**	**53.5**	**53.5**	**44.3**	**34.2**	**36.2**	**77.3**	**80.3**	**81.8**	**75.0**	**83.3**	**83.3**	**62.5**	**62.8**	**63.7**	**63.7**
Hierarchical Clusterer	3	2	50.5	50.5	50.5	34.2	34.2	34.2	75.8	74.2	75.8	50.0	50.0	50.0	**52.6**	**52.2**	**52.6**	**52.6**
Farthest First	3	2	50.5	50.5	50.5	34.2	34.9	34.9	78.8	77.3	78.8	58.3	58.3	58.3	**55.5**	**55.3**	**55.6**	**55.6**
X-Means	3	2	59.6	59.6	59.6	45.0	38.3	41.6	69.7	78.8	75.8	50.0	58.3	50.0	**56.1**	**58.7**	**56.7**	**58.7**
Cascaded K-Means	3	2	59.6	59.6	59.6	45.0	38.3	41.6	69.7	78.8	75.8	50.0	58.3	50.0	**56.1**	**58.7**	**56.7**	**58.7**
K-Means	**3**	**3**	**51.5**	**51.5**	**51.5**	**37.6**	**34.9**	**35.6**	**75.8**	**78.8**	**77.3**	**83.3**	**83.3**	**83.3**	**62.0**	**62.1**	**61.9**	**62.1**
Hierarchical Clusterer	3	3	50.5	50.5	50.5	34.2	34.2	34.2	77.3	75.8	77.3	50.0	50.0	50.0	**53.0**	**52.6**	**53.0**	**53.0**
Farthest First	3	3	50.5	50.5	50.5	34.9	34.2	34.2	66.7	75.8	72.7	66.7	58.3	58.3	**54.7**	**54.7**	**53.9**	**54.7**
X-Means	3	3	53.5	53.5	53.5	35.6	47.0	43.6	51.5	66.7	60.6	58.3	58.3	58.3	**49.7**	**56.4**	**54.0**	**56.4**
Cascaded K-Means	3	3	53.5	53.5	53.5	35.6	47.0	43.6	51.5	66.7	60.6	58.3	58.3	58.3	**49.7**	**56.4**	**54.0**	**56.4**
K-Means	3	4	50.5	50.5	50.5	38.9	35.6	36.9	78.8	77.3	75.8	75.0	75.0	75.0	**60.8**	**59.6**	**59.5**	**60.8**
Hierarchical Clusterer	3	4	50.5	50.5	50.5	34.2	34.2	34.2	75.8	75.8	75.8	50.0	50.0	50.0	**52.6**	**52.6**	**52.6**	**52.6**
Farthest First	3	4	50.5	50.5	50.5	34.2	34.2	34.2	51.5	86.4	81.8	75.0	58.3	66.7	**52.8**	**57.4**	**58.3**	**58.3**
X-Means	3	4	52.5	52.5	52.5	39.6	43.0	49.0	69.7	68.2	68.2	58.3	58.3	58.3	**55.0**	**55.5**	**57.0**	**57.0**
Cascaded K-Means	3	4	52.5	52.5	52.5	39.6	43.0	49.0	69.7	68.2	68.2	58.3	58.3	58.3	**55.0**	**55.5**	**57.0**	**57.0**
K-Means	4	2	52.5	52.5	52.5	35.6	34.2	34.2	74.2	72.7	72.7	83.3	83.3	83.3	**61.4**	**60.7**	**60.7**	**61.4**
Hierarchical Clusterer	4	2	50.5	50.5	50.5	34.2	34.2	34.2	75.8	74.2	75.8	50.0	50.0	50.0	**52.6**	**52.2**	**52.6**	**52.6**
Farthest First	4	2	50.5	50.5	50.5	34.2	34.2	34.9	83.3	86.4	84.8	66.7	58.3	58.3	**58.7**	**57.4**	**57.1**	**58.7**
X-Means	4	2	51.5	51.5	51.5	34.2	36.2	34.2	51.5	62.1	59.1	58.3	75.0	66.7	**48.9**	**56.2**	**52.9**	**56.2**
Cascaded K-Means	4	2	51.5	51.5	51.5	34.2	36.2	34.2	51.5	62.1	59.1	58.3	75.0	66.7	**48.9**	**56.2**	**52.9**	**56.2**
K-Means	4	3	50.5	50.5	50.5	35.6	34.2	34.9	80.3	77.3	78.8	75.0	83.3	83.3	**60.3**	**61.3**	**61.9**	**61.9**
Hierarchical Clusterer	4	3	50.5	50.5	50.5	34.2	34.2	34.2	75.8	74.2	75.8	58.3	50.0	50.0	**54.7**	**52.2**	**52.6**	**54.7**
Farthest First	4	3	52.5	52.5	52.5	34.9	34.2	34.2	75.8	89.4	84.8	58.3	58.3	58.3	**55.4**	**58.6**	**57.5**	**58.6**
X-Means	4	3	52.5	52.5	52.5	33.6	34.9	34.9	63.6	69.7	68.2	58.3	58.3	66.7	**52.0**	**53.9**	**55.6**	**55.6**
Cascaded K-Means	4	3	52.5	52.5	52.5	33.6	34.9	34.9	63.6	69.7	68.2	58.3	58.3	66.7	**52.0**	**53.9**	**55.6**	**55.6**
K-Means	4	4	50.5	50.5	50.5	34.2	34.2	34.2	78.8	77.3	78.8	75.0	83.3	83.3	**59.6**	**61.3**	**61.7**	**61.7**
Hierarchical Clusterer	4	4	50.5	50.5	50.5	34.2	34.2	34.2	77.3	75.8	75.8	50.0	50.0	50.0	**53.0**	**52.6**	**52.6**	**53.0**
Farthest First	4	4	51.5	51.5	51.5	34.2	34.2	34.2	74.2	84.8	80.3	66.7	58.3	58.3	**56.7**	**57.2**	**56.1**	**57.2**
X-Means	4	4	50.5	50.5	50.5	35.6	33.6	33.6	59.1	74.2	65.2	66.7	66.7	58.3	**53.0**	**56.2**	**51.9**	**56.2**
Cascaded K-Means	4	4	50.5	50.5	50.5	35.6	33.6	33.6	59.1	74.2	65.2	66.7	66.7	58.3	**53.0**	**56.2**	**51.9**	**56.2**

Table 5.2.: Clustering Evaluation Results in Percent For Different Test Data.

how the two different approaches perform over the same data sets. Therefore an additional evaluation is shown in the following, which compares the performance of the clustering algorithms to the performance of the classification algorithms used earlier in this thesis. To compensate the missing training data, a 10-fold cross-validation has been used for each classifier.

Table 5.3 shows the performance of each classifier compared to the best clustering result over the same data and pq-setting. It can be seen that the classifiers significantly outperform the clustering results for the Twain-Wells and Twain-Wells-Shelley documents. As expected from the authorship attribution and profiling results, respectively, the support vector machine framework (LibSVM) and the linear classifier (LibLinear) perform best, reaching a maximum accuracy of nearly 87% for the Twain-Wells document. Moreover, the average improvement is given in the bottom line, showing that most of the classifiers outperform the best clustering result by over 20% in average. Solely the kNN algorithm achieves minor improvements as it attributed the two-author document with a poor accuracy of about 60% only.

A similar general improvement could be achieved on the three-author document Twain-Wells-Shelley as can be seen in subtable (b). Again, LibSVM could achieve an accuracy of about 75%, whereas the best clustering configuration could only reach 49%. Except for the kNN algorithm, all classifiers significantly outperform the best clustering results for every configuration. In average, LibSVM could even improve the accuracy by 29%, whereas kNN reaches only 5% due to similar reasons as stated before.

Quite different comparison results have been obtained for the Federalist Papers and PAN12 data sets, respectively, which have been used for the authorship attribution experiments in Chapter 3. Here, the improvements gained from the classifiers are only minor, and in some cases are even negative, i.e., the classification algorithms perform worse than the clustering algorithms. A general explanation is the good performance of the clustering algorithms on these data sets, especially by utilizing the Farthest First and K-Means algorithms.

In case of the Federalist Papers data set shown in subtable (c), all algorithms except kNN could achieve at least some improvement. Although the LibLinear classifier could reach an outstanding accuracy of 97% using the configuration $p = 3$ and $q = 4$, the global improvement is below 10% for all classifiers. As before, the kNN algorithm performs worst and in average is even inferior to the clustering results.

For the sake of completeness also the authorship attribution data set PAN12 has been compared to the clustering results as shown in subtable (d). Here, the outcome is quite diverse as some classifiers could improve the clusterers

p	q	Algorithm	Max	N-Bay	Bay-Net	LibLin	LibSVM	kNN	J48
2	2	X-Means	57.6	77.8	82.3	85.2	86.9	62.6	85.5
2	3	X-Means	56.6	79.8	80.8	81.8	83.3	60.6	80.8
2	4	X-Means	57.6	76.8	79.8	82.2	83.8	58.6	81.0
3	2	X-Means	59.6	78.8	80.8	81.8	83.6	59.6	80.8
3	3	X-Means	53.5	76.8	77.8	80.5	82.3	61.6	79.8
3	4	X-Means	52.5	81.8	79.8	81.8	83.8	63.6	82.0
4	2	K-Means	52.5	86.9	83.3	83.5	84.3	62.6	81.8
4	3	X-Means	52.5	79.8	79.8	80.1	80.3	59.6	77.4
4	4	Farth. First	51.5	72.7	74.7	75.8	77.0	60.6	75.8
average improvement				**24.1**	**25.0**	**26.5**	**27.9**	**6.2**	**25.7**

(a) Twain-Wells

p	q	Algorithm	Max	N-Bay	Bay-Net	LibLin	LibSVM	kNN	J48
2	2	K-Means	44.3	67.8	70.8	74.0	75.2	51.0	73.3
2	3	X-Means	38.3	65.1	67.1	70.7	72.3	48.3	70.2
2	4	X-Means	45.6	63.1	68.1	70.5	71.8	49.0	69.3
3	2	X-Means	45.0	51.7	64.1	67.3	68.8	45.6	65.4
3	3	X-Means	47.0	57.7	64.8	67.3	68.5	47.0	65.9
3	4	X-Means	49.0	67.8	67.8	70.5	72.5	46.3	68.3
4	2	X-Means	36.2	61.1	67.1	69.1	69.5	50.3	65.1
4	3	K-Means	35.6	53.0	63.8	67.6	70.0	47.0	66.6
4	4	X-Means	35.6	57.7	66.1	68.5	69.3	42.3	66.8
average improvement				**18.7**	**24.8**	**27.7**	**29.0**	**5.6**	**26.0**

(b) Twain-Wells-Shelley

p	q	Algorithm	Max	N-Bay	Bay-Net	LibLin	LibSVM	kNN	J48
2	2	Farth. First	77.3	81.1	86.4	90.9	84.2	74.2	81.8
2	3	Farth. First	78.8	85.6	87.4	92.4	89.0	78.8	82.8
2	4	X-Means	78.8	89.4	92.4	90.9	87.3	89.4	85.9
3	2	K-Means	81.8	82.6	87.9	92.4	85.5	80.3	83.8
3	3	K-Means	78.8	92.4	92.4	92.4	86.4	81.8	83.8
3	4	Farth. First	86.4	84.8	90.9	97.0	85.8	81.8	85.6
4	2	Farth. First	86.6	81.8	89.4	87.9	83.3	77.3	84.1
4	3	Farth. First	89.4	85.6	92.4	89.4	85.8	80.3	83.3
4	4	Farth. First	84.8	86.4	90.9	89.4	85.8	84.8	83.6
average improvement				**3.0**	**7.5**	**8.9**	**3.4**	**-1.6**	**1.3**

(c) Federalist Papers

p	q	Algorithm	Max	N-Bay	Bay-Net	LibLin	LibSVM	kNN	J48
2	2	K-Means	83.3	83.3	33.3	100.0	100.0	100.0	33.3
2	3	K-Means	83.3	83.3	33.3	100.0	100.0	100.0	33.3
2	4	K-Means	83.3	83.4	33.3	100.0	100.0	100.0	33.3
3	2	K-Means	83.3	75.0	33.3	91.7	91.7	100.0	33.3
3	3	K-Means	83.3	100.0	33.3	100.0	91.7	100.0	33.3
3	4	Farth. First	75.0	66.7	33.3	100.0	100.0	91.7	33.3
4	2	K-Means	83.3	91.7	33.3	91.7	75.0	91.7	33.3
4	3	K-Means	83.3	75.0	33.3	100.0	75.0	91.7	33.3
4	4	K-Means	83.3	75.0	33.3	100.0	83.4	83.4	33.3
average improvement				**-0.9**	**-49.1**	**15.8**	**8.4**	**13.0**	**-49.1**

(d) PAN12-A/B

Table 5.3.: Best Evaluation Results for each Clustering Algorithm and Test Data Set in Percent.

significantly up to 15%, whereas others worsen the accuracy even more drastically down by 50%. A possible explanation might be the small data set (as before, only the subproblems A and B have been used), which may not be suited very well for a reliable evaluation of the clustering approaches. Nevertheless, LibLinear and LibSVM also reach very good results and could often classify documents (paragraphs) 100% correctly. Interestingly, the kNN classifier which performed poor on all other data sets could often also gain 100%. On the contrary the previously good working classifiers BayesNet and J48 could constantly only attribute one third correctly.

Summarizing, the comparison of the different algorithms reveal that in general classification algorithms perform better than clustering algorithms when provided with the same (pq-gram) feature set. Nevertheless, the results of the PAN12 experiment are very diverse and indicate that there might be a problem with the test data set itself, and that this comparison should be treated carefully - especially when taking into account that all other experiments show consistent outcomes.

A visualization of the average improvement for each classifier, pq-gram configuration and data set is illustrated in Figure 5.1. The more colored a square is, the higher the average improvement compared to the best clustering results is. Also here it can be seen clearly that the kNN classifier lacks significantly of attributing the Twain-Wells/-Shelley documents correctly, whereas it performs better on the Federalist Papers and especially on the PAN12 problems.

5.5. Conclusion and Future Work

Conclusion

In this chapter the automatic creation of paragraph clusters based on the grammar of authors has been evaluated. Different state-of-the-art clustering algorithms have been utilized with different input features and tested on different data sets. The best working algorithm K-Means could achieve an accuracy of about 63% over all test sets, whereby good individual results of up to 89% could be reached for some configurations on the reutilized authorship attribution data sets. On the contrary the specifically created documents incorporating two and three authors could only be clustered with a maximum accuracy of 59%.

A comparison between clustering and classification algorithms using the same input features has been implemented. Disregarding the missing training data, it could be observed that classifiers generally produce higher accuracies with improvements of up to 29%. On the other hand, some classifiers perform worse

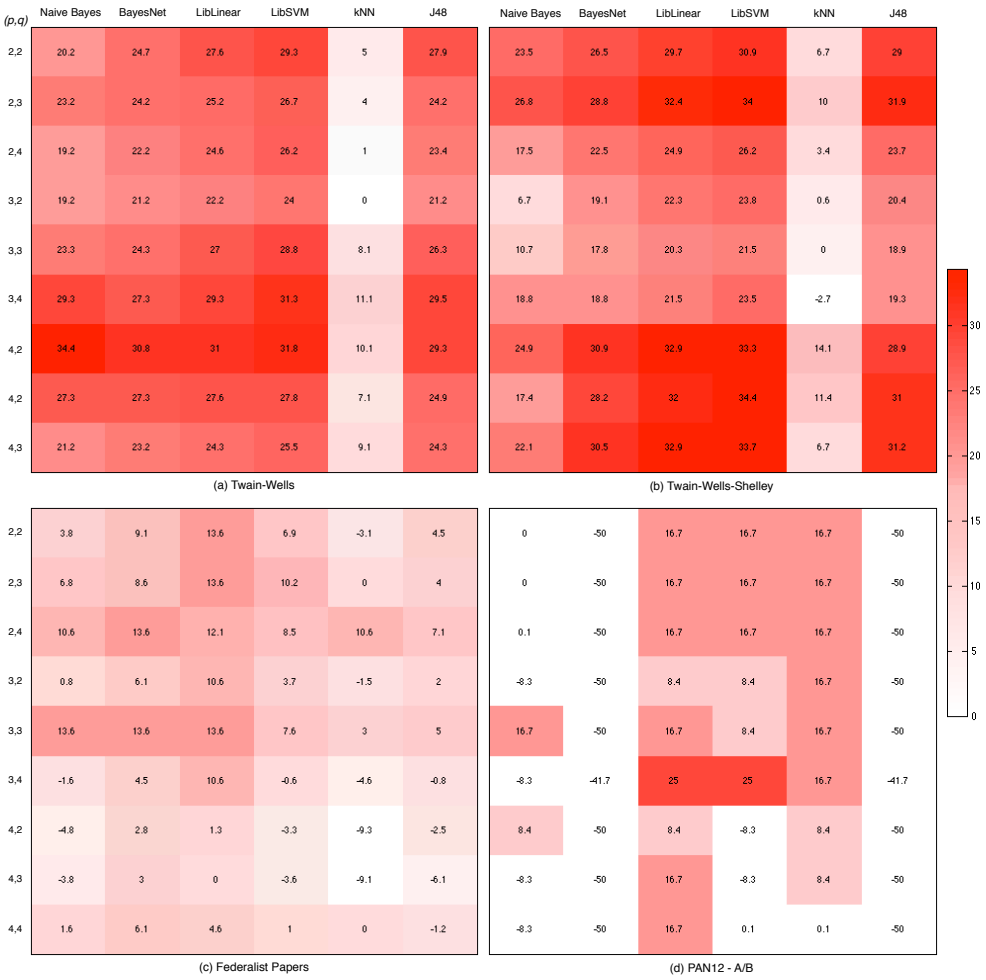


Figure 5.1.: Improvement in Percent of Different Classifiers Compared To Best Clustering Results.

on average than clustering algorithms over individual data sets when using some pq-gram configurations. Nevertheless, if the maximum accuracy for each algorithm is considered, all classifiers perform significantly better as can be seen in Figure 5.2. Here the best performances of all utilized classification and clustering algorithms are illustrated. The linear classification algorithm LibLinear could reach nearly 88%, outperforming K-Means by 25% over all data sets.

Finally, the best classification and clustering results are shown for each data set in Figure 5.3. Consequently the classifiers achieve higher accuracies, whereby the PAN12 subsets could be classified 100% correctly. As can be seen, a major improvement can be gained for the novel literature documents. For example,

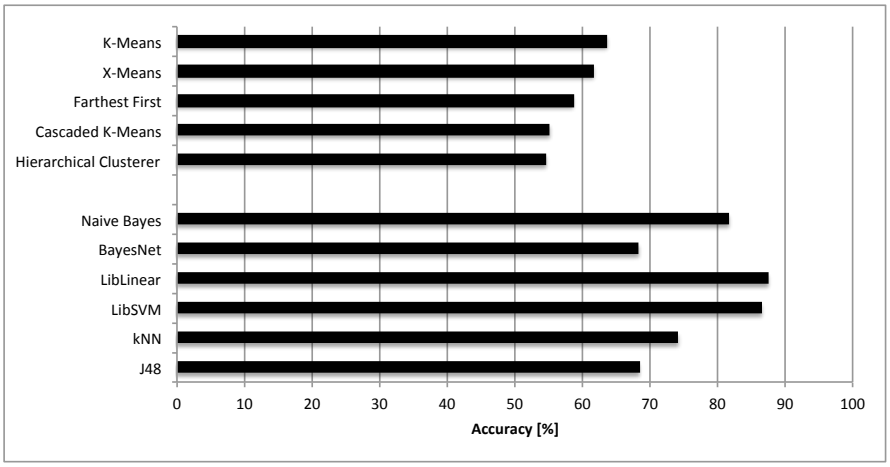


Figure 5.2.: Best Evaluation Results Over All Data Sets For All Utilized Clustering and Classification Algorithms.

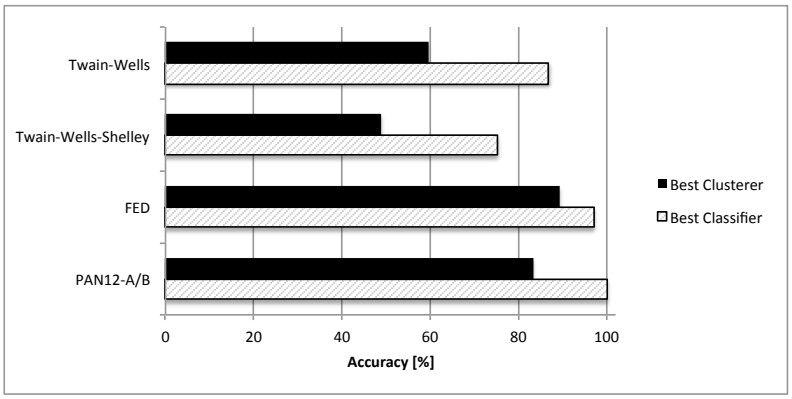


Figure 5.3.: Best Clustering and Classification Results For Each Data Set.

the best classifier reached 87% on the Twain-Wells document, whereas the best clustering approach achieved only 59%.

Future Work

As shown in this chapter, documents can be clustered based on grammar features, but the accuracy is below that of classification algorithms. Although the two algorithm types should not be compared directly as they are designed to manage different problems, the significant differences in accuracies indicate that classifiers can handle the grammar features better. Nevertheless future work should focus on evaluating the same features on larger data sets, as clustering algorithms may produce better results with increasing amount of sample data.

Another possible application could be the creation of whole document clusters, where documents with similar grammar are grouped together. Despite the fact that such huge clusters are very difficult to evaluate - due to the lack of ground truth data - a navigation through thousands of documents based on grammar may be interesting like it has been done for music genres (e.g. [155, 114]) or images (e.g. [44, 108]). Moreover, grammar clusters may also be utilized for modern recommendation algorithms once they have been calculated for large data sets. For example, by analyzing all freely available books from libraries like Project Gutenberg, a system could recommend other books with a similar style based on the users reading history. Also, an enhancement of current commercial recommender systems that are used in large online stores like Amazon is conceivable.

Finally and similarly to the previous chapters, the clustering approach should be evaluated on data sets with different languages, and also with other clustering algorithms like EM [118], CobWeb [46] or Self-Organizing Maps (SOM) [92].

CHAPTER 6

Related Work

This chapter summarizes other scientific work that has been published in areas that are closely related to this thesis. In particular, an overview of recent approaches and algorithms as well as a general overview concerning the respective chapters is given. Section 6.1 discusses related work in the field of plagiarism detection, focussing on intrinsic methods but also briefly discussing external approaches. Often used authorship attribution algorithms are presented in Section 6.2, and Section 6.3 gives an overview of current author profiling approaches. Finally, Section 6.4 concludes with a summary of related work in the field of text-based clustering.

6.1. Plagiarism Detection

Basically the two different approaches for identifying plagiarism in text documents are usually referred to as *external* and *intrinsic* algorithms [142], where external algorithms compare a suspicious document against a given unrestricted set of source documents - for example large databases collected from the world wide web - and intrinsic methods are allowed to analyze solely the document itself. A similar classification lists three plagiarism detection categories that are currently used in practice [113]: (i) document source com-

parison (conforms to external detection), (ii) stylometry (conforms to intrinsic detection) and (iii) manual search of characteristic phrases. When performing the latter type - which has been found surprisingly successful in practical situations - the examiner manually chooses a characteristic paragraph or a couple of consecutive sentences, which are then copied and pasted into common internet search engines like Google, hoping that the search engine's internal algorithms will find a corresponding source document. Because this method is very labor-intensive, an automatic cut-and-paste detection system has been developed in [126] by using the (meanwhile deprecated) Google Web Search API[1].

Many of the approaches or at least the main ideas of recent plagiarism detection algorithms have been originally proposed, elaborated and published through the PAN workshop "on uncovering plagiarism, authorship, and social software misuse" [136], which has been and still is organized by the Bauhaus-University of Weimar on a regular basis for several years. Focussing on plagiarism detection [143], the workshop also covers related topics like author identification [83], author profiling [149], Wikipedia quality flaw prediction [5] or even sexual predator identification [75].

Intrinsic Plagiarism Detection

Intrinsic plagiarism detection requires the analysis of different stylometric features, whereby *"most stylometric features fall in one of the following five categories: (i) text statistics, which operate at the character level, (ii) syntactic features, which measure writing style at the sentence-level, (iii) part-of-speech features to quantify the use of word classes, (iv) closed-class word sets to count special words, and (v) structural features, which reflect text organization"* [208]. An overview of features that have been used in stylometry is given in [172] and summarized in Table 6.1. In the following an excerpt of some recent intrinsic plagiarism detection methods incorporating some of the listed features is given.

An approach that is very similar to the PQ-Plag-Inn algorithm (see Section 2.6), but uses character n-gram profiles to identify plagiarism is described in [169]. With this method a profile of the whole document as well as profiles for sliding windows traversing the document are created, whereby a profile is a vector of normalized frequencies of all occurring character trigrams. Each sliding window profile is then compared with the document profile by applying a style change function which is similar to the distance metrics described in Section 3.3. Generally, when n-grams are used often also preprocessing steps

[1] https://developers.google.com/web-search, visited June 2014

Type	Feature	Reference
lexical (character-based)	character-frequency	[205]
	character n-grams	[169, 154, 80, 89, 97]
	frequency of special characters	[205]
	compression rate	[157, 168]
lexical (word-based)	average word/sentence length	[205, 73]
	word frequency	[73, 119, 97]
	word n-grams	[154]
	number of hapax dis-/legomena	[193, 205]
	type-token ratio	[202, 73, 205]
syntactic	POS tags	[168, 97]
	POS n-grams	[96, 97]
	frequency of function words	[119, 73, 205, 10, 96]
	frequency of punctuations	[205]
structural	average paragraph length	[205]
	indentation	[205]
	greetings, farewells, signatures	[205, 168]

Table 6.1.: Excerpt of Important Features Used in Stylometric Analysis [172].

like transforming all letters into lowercase are performed. In this approach, additionally every n-gram containing no letter characters is removed. The approach reaches an F-score of 28% and 30%, respectively, on two different data sets, whereby the recall is significantly higher than the precision. Finally, also experiments have been conducted on varying the text length, with the summarizing outcome that the recall value increases with the length of the text, but at the cost of the precision.

Character n-grams and in particular trigrams are also used in [87]. Here, the document is also traversed using sliding windows, but no document profile is calculated. Instead, each window is compared against each other window using a proposed symmetric distance score for n-grams, and each distance is stored into a distance matrix. Thus this approach is similar to the Plag-Inn algorithm described in Section 2.4, with the main difference that the analysis is based on larger text windows rather than single sentences (besides the fact that n-grams are used with a different comparison metric). While the Plag-Inn algorithm then performs further processing using a Gaussian normal distribution, this approach uses outlier identification algorithms by incorporating Principal Components Analysis (PCA), *"a standard technique for dimensionality reduction, commonly applied in stylometry"* [21]. On the PAN-PC-11 corpus [147] the approach could reach a plagiarism detection rate of 27.6%.

Another approach described in [132] also uses the sliding window technique, but is based on word frequencies and the assumption that authors use a significant set of words. After eliminating whitespaces and transforming all characters into lowercase, a vector of word unigrams is extracted together with the respective frequencies of the words. Finally, the frequency vector of each sliding window is then compared to the "document footprint" by using a proposed metric, and every window that has a higher distance than a predefined threshold is marked as potentially plagiarized. The approach achieved an overall score of 32% over the intrinsic part of the PAN-PC-11 corpus, and was also tested on the external part of the corpus, where an F-score of about 16% could be reached.

An algorithm that tries to recognize paraphrased sections based on the phrase structure of sentences and the structure of verb phrases is described in [195]. In this work, sentence-initial and sentence-final phrases are inspected together with predefined semantic classes of verbs [105] by part-of-speech tagging. It uses POS-tags only, i.e., without referring to computationally intense full parse tree, in combination with several other linguistic features. In an evaluation using whole chapters of different books an F-score of over 75% could be gained. Nevertheless, the experiment has been conducted in a way that cannot be directly compared to intrinsic plagiarism detection, but rather to results of authorship attribution.

[157] discusses the comparison of binary strings calculated from word groups like nouns or verbs using complexity analysis, i.e., the Kolmogorov complexity [93]. In particular, statistical compressing algorithms have been trained by using the respective occurrences of word groups over the whole document, and then utilized to compress a given text segment. The main idea can be summarized in the following hypothesis: the similar the writing style of a paragraph is compared to the document, the better the compression algorithm works, i.e., the more the paragraph can be compressed. Accordingly, potential plagiarism is then detected by analyzing the compression rate of different text fragments. It has been shown that by additionally incorporating complexity analysis together with neural networks, the accuracy of intrinsic plagiarism detection algorithms can be increased by up to 4% (on the PAN 09 intrinsic plagiarism competition corpus [40]).

Further research using complexity analysis has shown that even non-trained compression algorithms like Lempel-Ziv [206] can be used to distinguish authentic from non-authentic text [173, 163]. Hereby the basic assumption - which has been solidified - is that real authors frequently repeat words and ideas to increase readability, which leads to a higher compression rate. This idea has been successfully used to distinguish authentic from non-authentic

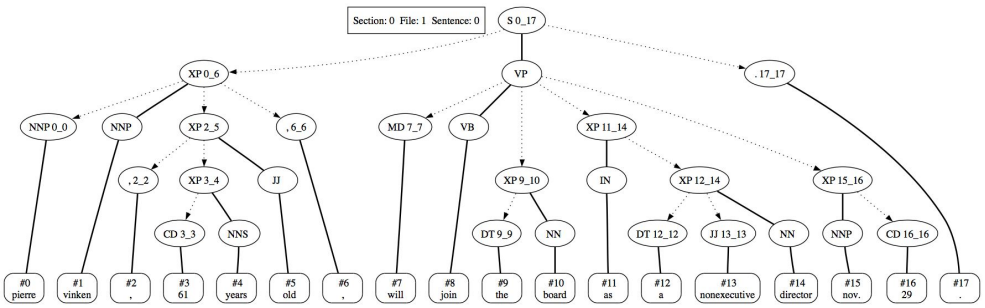


Figure 6.1.: Example of a LTAG-spinal tree[2].

text, i.e., coherent text that has been written by real human authors from computer-generated text like automatically created spam [164].

As an alternative to the heavily used pq-grams throughout this thesis, also lexicalized tree-adjoining-grammars (LTAG) could be considered. Originally proposed in [79] as a ruleset to construct grammar syntax by using partial subtrees, LTAG's or the linguistically enhanced LTAG-spinal variant [159] may also be used to analyze the grammar structure of sentences. The basic idea is thereby to provide a set of rules to rewrite trees instead of symbols like in context-free grammars. Finally, the result of multiple tree rewriting leads to grammar trees which comprise the subtrees inferred from the applied rules, and which can be parsed by using targeted LTAG parsers. An example[2] of an LTAG-spinal derivation is illustrated in Figure 6.1.

External Plagiarism Detection

Given a suspicious document and a large document collection, the task of external plagiarism detection basically consists of three phases as can be seen in Figure 6.2 [174]: (i) retrieving relevant documents from the collection (*source retrieval*), (ii) comparing the candidate documents with the suspicious document for similarity (*text alignment*) and (iii) knowledge-based post-processing, where similar text pairs are filtered, cleaned and presented to the user. One of the key criteria for good working plagiarism detection algorithms is thereby the quality of the source document database, which may be stored statically as a result from earlier data gathering, but may also be available dynamically by utilizing search engines, for example. Moreover, a hybrid variant is possible.

To be able to compare current external plagiarism approaches in more detail, the two major subtasks *source retrieval* and *text alignment* are often inspected

[2]LTAG-spinal: Treebank and parsers, http://www.cis.upenn.edu/~xtag/spinal, visited June 2014

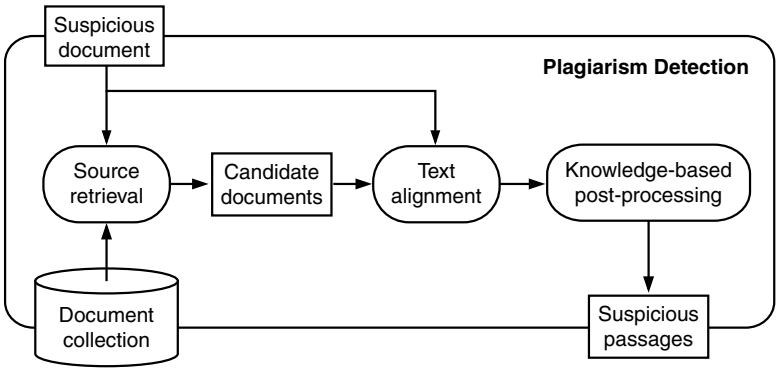


Figure 6.2.: Overview of the External Plagiarism Detection Process [174].

individually in scientific communities. In the following an overview of strategies covering these subtasks is given, which is mainly based on the survey paper of Potthast et al. [144].

Source Retrieval

Most of the source retrieval approaches are based on the following five-step strategy:

1. **Chunking:** As a first step the suspicious document is split into several text chunks, which represent the basis for further analyses. Commonly used solutions include the chunking by a predefined number of words [196], sentences [200], lines [42] or paragraphs [101], no-chunking (processing the whole document at once) [177], and also the utilization of advanced tools like TextTiling [68] to split the document by topically unrelated paragraphs [66]. Additionally, intrinsic plagiarism approaches are used to identify irregular paragraphs or sentences [129].

2. **Keyphrase extraction:** Probably one of the most important steps for source retrieval is the selection of phrases, words or other fragments out from the previously calculated text chunks. The global aim is to *"select only those phrases (or words) which maximize the chance of retrieving source documents matching the suspicious document"* [144]. As further processing is often time consuming and computationally costly, an additional objective is the quantitative reduction of relevant text. Algorithms often combine different strategies, including the extraction of the top noun phrases [42] and keyphrases [66] using different existing tools [16, 41] or the usage of n-grams with $2 \leq n \leq 10$ [196, 200, 77]. To find relevant n-grams, Pat Trees [59] or n-gram reference corpora

extracted from external sources like Google Books are used [102, 101]. Additionally, the information retrieval measure $tf-idf$ is repeatedly applied to identify the relevance of n-grams or words [42, 177], whereby the document frequencies are generally obtained from existing external corpora.

3. **Query Formulation:** Using the keywords extracted from the previous step, one or more queries are formulated which are passed to the API of a search engine. The aim is to create queries in a way that the possibilities of the search engine in use are exploited as much as possible. Thereby many plagiarism detectors use search engines that operate on the *ClueWeb corpus 2009*[3], a corpus that consists of about one billion multi-language web pages. Examples of publicly available search engines incorporating the corpus are Indri[4], which among others supports phrasal search, or ChatNoir [145], which makes use of PageRank and SpamRank scores.

4. **Search Control:** The task of a search controller is to schedule the queries to the search engine and to dynamically react to incoming search results. Depending on the size of the corpus and the number of queries, it is advisable to reduce or reformulate queries in order to reduce processing time. For example, in an optimal scenario the first query result already reveals the source of plagiarism, and thus no further queries should be scheduled. In reality current algorithms have to consider deeper strategies to decide whether current queries should be reformulated or further queries should be aborted. For example, in [66] queries are dropped if 60% of its terms are present in a previously obtained document, or in [101] the submission of queries is stopped if the amount of already found plagiarism is above a certain threshold.

5. **Download Filtering:** Finally, the algorithms have to decide how much of the query results are relevant for the further text alignment process, i.e., which documents should be downloaded and added to the set of potential candidates. A straight forward approach is thereby to consider the top-k results, e.g., $k = 1$ [66], $k = 3$ [200] or $k = 10$ [42] and to download them if certain conditions hold. For example, documents are downloaded *"when at least 90% of the words in a 160-character snippet are contained in the suspicious document"* [144, 42] or the cosine-similarity of the query result and the suspicious document exceed a certain threshold

[3] http://lemurproject.org/clueweb09, visited June 2014

[4] http://lemurproject.org/clueweb09/index.php#Services, visited June 2014

[102]. Additionally, also the frequency of a specific documents appearing in several different query results are considered [101].

Using the Webis-TRC-2012 corpus [146] the best tradeoff between recall and precision of source retrieval approaches reaches an F-score of 47% [200]. It is based on sentence chunking in combination with n-gram keyphrase extraction and downloading the top three query results. A different approach that splits the suspicious document by topically related paragraphs [66] achieves a high precision of 63% at the cost of a low recall, resulting in an overall F-score of 44% over the same data set.

Text Alignment

Within the text alignment phase the suspicious document is compared to a possible source document, obtained from the previously described source retrieval. The global aim is to find matching text passages of both documents, even if the text has been altered or obfuscated such that - beside stop words - nearly no lexical similarity is present. In the following an overview of the basic three-step process is given [144]:

1. **Seeding:** In the first step, text matches (usually referred to as "seeds") between the suspicious and the source document are identified, whereby this can be exact matches or matches that are based on one or more specific seed heuristics. The global aim is thereby not to already produce final plagiarism detection results, but to find as much reasonable plagiarism origins as possible, which are then extended in the subsequent step. Examples for seed heuristics include sorted word-n-grams [135, 182, 177], unsorted stop word-n-grams [170] or word-n-grams which contain at least one named entity [160]. Moreover, seeds are also created using sentence pairs that exceed a certain similarity threshold [102]. Often also preprocessing steps like the removal of whitespaces, cases, stop words (if not used) or word stemming are applied.

2. **Extension:** Similar to the sentence selection algorithm presented in this thesis (see Section 2.4.2), the aim of the extension phase of external plagiarism detectors is to merge previously found seeds into aligned text passages. The basic idea is to present whole passages of plagiarism rather than multiple seeds, especially if any kinds of (word) n-grams have been used as seeds. For example, word-3-grams should be extended to whole sentences or even paragraphs if possible. To do this, most of the algorithms are rule-based and combine seeds if they are adjacent in the source as well as in the suspicious document. Additionally, gaps between different seeds are also incorporated if they fulfill certain thresholds. For example, in [177] two seeds are merged if the gap between them is less or

equal to 4,000 characters. While most of the algorithms use rules like in the previous example, also other approaches exist that use unsupervised learning, e.g., by utilizing clustering algorithms [135].

3. **Filtering:** As a postprocessing step, the detected passages are once again filtered if needed. Basically, passages are excluded from the final result set if they do not meet certain criteria like a minimum character size [176, 134], for example. Other approaches discard passages shorter than a fixed amount of words that fall below a given cosine-similarity [55]. Also, overlapping text passages are revised in this phase, e.g., if the Jaccard similarity coefficient[5] of overlapping words is below a certain threshold [103]. Summarizing, the filtering step is not crucial to the algorithms, but is needed to be able to present the result nicely to a user. This step may also be integrated in the previous extension phase.

As a reference of performance, algorithms could reach an overall plagiarism detection rate[6] of over 80% [182, 102], whereby the precision is most often significantly higher than the recall.

In conclusion, it can be stated that the difficulties for external plagiarism detectors are located in the source retrieval part, as can be inferred from evaluation results. Thereby the retrieval of relevant documents is highly dependent on the underlying data set. Beside the employed algorithms, the latter is also a key criterion for good working commercial plagiarism tools, which are briefly summarized in the following section.

Commercial Software

In recent years the use of commercial plagiarism detection software has become a standard practice in academia, and is increasingly required as a mandatory step before submitting (doctoral) theses. An extensive evaluation of the powerfulness and expressiveness of current software has been performed by Weber-Wulff et al. [197, 198] The authors tested 26 systems and stated that only a small amount is "partly usable", i.e., the best systems only found 60-70% of plagiarism. Moreover, plagiarism having books as sources, plagiarism by translation or structural plagiarism[7] have not been found by any system. Additionally the authors claim that the usability and in particular the presentation of results is poor in most systems. For example, for a well-known

[5] http://en.wikipedia.org/wiki/Jaccard_index, visited June 2014

[6] the detection rate used incorporates F-score as well as the granularity value

[7] the reasoning, arguments or footnotes are reused in the same order, but the actual text is rewritten individually

Category	Software	Reference
partially useful	Turnitin	www.turnitin.com
	Copyscape	www.copyscape.com
marginally useful	Ephorus	www.ephorus.com
	PlagAware	www.plagaware.com
	PlagScan	www.plagscan.com
	Plagiarism Detect	www.plagiarism-detect.org
useless for academic purposes	PlagTracker	www.plagtracker.com
	Plagiarisma	www.plagiarisma.net

Table 6.2.: Results of a Recent Study[9] on Commercial Plagiarism Detection Systems.

suspicious thesis[8] submitted to the systems, some of them report more than thousand plagiarism sources, of which the majority is based on text fragments of less than 20 sentences. In reality it has been found by manual crowd-sourced efforts that there are only 131 sources.

In a recent study[9] many systems have been tested and categorized. An excerpt of the results is given in Table 6.2.

The quality of commercial plagiarism detectors is highly dependent on the underlying data that can be consulted when searching for plagiarism. For example, the mostly used[10] tool *Turnitin* claims that their *"database contains 45+ billion web pages, 337+ million student papers and 130+ million articles from academic books and publications"*[11] that can be accessed while searching. Unfortunately, the question of *how* the checks are performed and which algorithms are used remains unclear, which is understandable from a business point of view.

6.2. Authorship Attribution

The Federalist Papers

Without doubt the most influential work in the field of authorship attribution is the study of Mosteller and Wallace on the authorships of *"The Federalist*

[8] the dissertation of Karl-Theodor zu Guttenberg from 2009

[9] HTW Berlin Plagiats Portal, Softwaretest 2013, http://plagiat.htw-berlin.de/software-en/test2013, visited June 2014

[10] according to the Turnitin website: http://turnitin.com/en_us/features/faqs, visited June 2014

[11] http://turnitin.com/en_us/features/originalitycheck, visited June 2014

Papers"[12] [119] in the second half of the 20th century. Where beforehand only manual statistical analysis of text documents have been conducted (e.g. [202]), the authors successfully discriminated the authorships of nearly hundred political essays by applying a fairly simple statistical analysis of frequencies of common words like articles (e.g. "'a", 'the") or prepositions (e.g. "in", "of')'. For the 12 essays with questionable authorships (usually referred to as "the disputed papers"), a plausible answer could be given for the first time based on a scientific analysis: *"In summary, we can say with better foundation than ever before that Madison was the author of the 12 disputed papers."* [119, 49].

Since then many approaches revisited the Federalist Papers or used them as a test data set, which is also the case in this thesis. For example, in a popular study by Holmes et al. three novel features have been tested by using the papers, whereby *"the techniques examined are a multivariate approach to vocabulary richness, analysis of the frequencies of occurrence of sets of common high-frequency words, and use of a machine-learning package based on a 'genetic algorithm"'* [74]. One of the first attempts which applied support vector machines to the field of authorship attribution also analyzed the disputed papers [49]. In this work an author-separating mathematical hyperplane could be found that is based only on the occurrences of the three words "to", "upon" and "would". With this method all disputed papers were also attributed to Madison. Another approach [22] also tried to find hyperplanes with linear programming techniques, again using only three words: "as", "our", "upon". Also in this paper the authors concluded with the same result regarding the authorship of the disputed papers. Generally, more than 20 studies (the number *"is hardly complete"* [80]) on the authorships of the disputed Federalist Papers have been published [153], nearly all concluding and confirming the original result of Mosteller and Wallace, which claim that Madison is the author of the disputed documents.

Approaches

Using several features, machine learning algorithms are often applied to learn from author profiles and to predict unlabeled documents. Among methods that are utilized in authorship attribution as well as the related problem classes like text categorization are support vector machines [167, 154, 38], neural networks [194], naive bayes classifiers [115] or decision trees [195]. Thereby the list of features ranges from lexical, syntactic or semantic categories to application-specific features (e.g. the "technical structure" like fonts, colors, images or hyperlinks [1]) as depicted previously in Table 6.1.

[12] The Federalist Papers are a collection of political essays written by Alexander Hamilton, James Madison and John Jay in 1787/88 for the citizens of New York, with the aim to convince the people to adopt the newly formulated constitution of the United States.

In the following an excerpt of approaches using these features is given. Detailed and comprehensive surveys on the topic are given by Holmes [72] for methods until 1994, by Juola [80] until 2006 and by Stamatatos [168] and Koppel et al. [97] until 2009, respectively. An overview of recent approaches can be found in the PAN 2013 overview paper [83].

Lexical Features

Although the number of proposed features has been increasing in the last years, it has been shown that simple word-based statistics are among the best to discriminate between authors. For example, on a test data set using various chapters of twenty different books, an accuracy of 99% could be gained by only using the vectors of most frequent words [9]. The basic principle behind the analysis of these so-called *function words* is that they are topic-independent and that they are used mainly unconsciously by an author. Thus, function words are heavily used in authorship attribution (e.g. in [204] as the only features given to a support vector machine), or represent at least part of the feature list. Thereby a main challenge is to define the number of words that are considered, which ranges from 100 [25] over 480 [96] up to 1,000 [165] words. Moreover, also approaches exist that include all occurring words, for example in [38] an accuracy of 60-80% could be gained on a German newspaper dataset by feeding all words into a support vector machine.

Also in authorship attribution, n-grams are used successfully to distinguish between writers. A recent evaluation [64] comparing almost fourty different features concludes with the result that n-grams with $n = 2, 3, 4$ gain the best results, e.g., discriminating two columnists of *The Telegraph* with an accuracy of up to 94%, three authors with 91% and ten authors with up to 79%. Interestingly, the accuracy drops significantly with $n \geq 5$. Profiles of unigrams of punctuation marks are also in the top results of the study. Finally, and conforming to other tests, the best result could be gained by using word profiles combined with unigrams of punctuation marks. Other work also experimented with n-grams in other languages than English (e.g. Greek [139]) and proved their language-independence, which is probably the most significant advantage of n-grams in general. Nevertheless, some algorithms and workarounds are needed to extract n-grams from certain languages like Traditional Chinese [125] or Japanese [122]. Controversially to the language-independent nature of n-grams, a study [62] revealed that n-grams can also be used to *detect* the language of short query-style texts (with best results using a Naive Bayes classifier on 5-grams).

Syntactic Features

The first attempt to go beyond character- and word-level was proposed in 1996 [13], where also syntactic patterns have been analyzed. By using the so-called *TOSCA* scheme, which is different from the nowadays commonly used Penn Treebank set, the frequencies of (complex) grammatical rewrite rules are examined and compared to word-based methods on the same corpus. Strengthened by many experiments, the pioneer study concludes with the suggestion that *"syntactic annotation provides excellent clues for authorship attribution, and that measures focusing on syntactic creativity are especially promising."* [13]

Since then many authors included syntactic features in their studies. For example, in [50] the previously described rewrite-rule technique has been simplified, compared to and combined with many other features like length of noun phrases, sentences, function word frequencies, POS trigrams and semantic features. In experiments using (only) five works of three novel authors, the syntactic features performed worst when examined isolated (although reaching an accuracy of over 85%), but it could be shown that by adding syntactic information to the set of features, the performance of authorship attribution systems can be enhanced significantly (reaching up to 97% on that data set).

A further example of an approach that uses syntactic information to discriminate the writers of short[13] texts is proposed in [71]. Also here the frequencies of rewrite rules are considered, but by using a partial parser [3] instead of a full parser. The outcome of a partial parser is a structural analysis of sentences that contains deeper information than just pure POS tags, but less than that of full parse trees. The result is therefore not that exact and to can be noisy to some extent. In an evaluation of samples of two authors only an accuracy of over 90% could be achieved by combining this feature with another syntactic feature (frequencies of POS bigrams) and other lexical features like vocabulary-richness or average word/sentence lengths.

Similar to the grammar tree analysis conducted within this thesis, in [112] the authors also experimented with structural parse tree features for other text categorization problems like nationality detection or sentiment analysis. Thereby also parse trees of sentences are calculated, of which several features like average tree branch factors, tree depths, frequencies of subtrees according to specified rewrite rules or subtree skeletons are extracted. As illustrated in Figure 6.3, the latter incorporate the structure patterns of parse trees, but without the concrete labeling of (inner) nodes like with pq-grams. In addition to the tree features, also common characteristics like word n-grams, POS tags

[13] less than 500 words, i.e., approximately 20-40 sentences

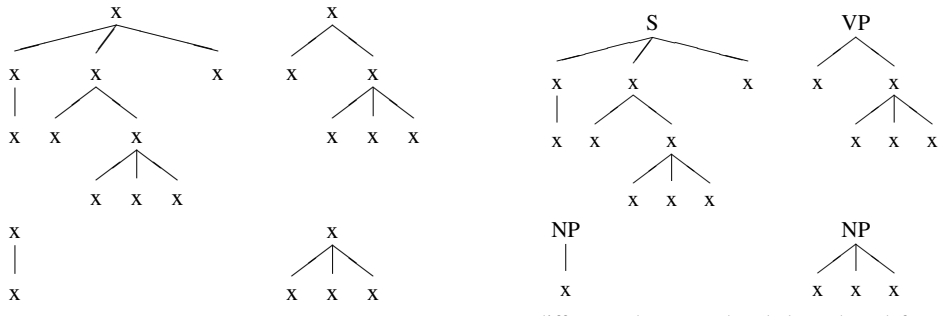


Figure 6.3.: Grammar Tree Skeletons Taken From and Used In [112].

or function words have been examined for an evaluation on manually created test data sets. It is shown that the basic word and POS tag features led to the best results when examined individually, e.g., reaching over 80% on nationality detection. The tree features alone led to significantly less accuracy, but it could be shown that by combining these structural features - especially the skeletons - with the common features, the overall result can be enhanced significantly.

Other Features

As already can be seen from the previous examples, most of the current approaches do not include only one feature, but several distinct measures. The number of utilized features in authorship attribution studies was estimated by Rudman in 1998 to a number of over 1,000 [152], and thus a challenge is to select the most appropriate ones. For example, the experiment in [205] used different machine learning algorithms with more than 270 distinct features, coming from many lexical, syntactic, structural and content-specific categories. An evaluation on English and Chinese online-newsgroup messages reached an accuracy of 70% up to 95%. Notably and conforming to the results presented in this thesis, the support vector machine classifier outperformed all other evaluated learning algorithms also in this study.

Besides the common learning algorithms like Naive Bayes, decision trees or support vector machines, also Markov chains have been utilized. For example, in [88] chains of letter bigrams are computed and tested on 380 random documents from Project Gutenberg. In this experiment, an attribution rate of 74% could be gained using the Markov chains. Moreover, also similar compression-based approaches like in plagiarism detection are used. A study [111] comparing the compression algorithms RAR, GZIP and LZW on different data sets (one of them being the Federalist Papers) reveals that such approaches

can gain an accuracy of approximately 60%-90%, whereby the result is heavily dependent on the compression algorithm and the size of the training data.

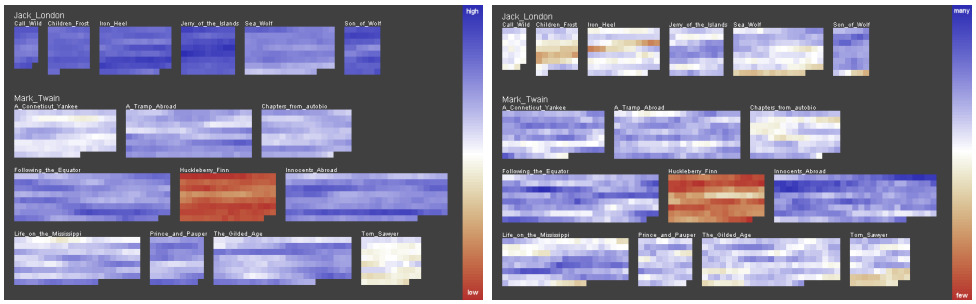
Another interesting approach used in authorship attribution that tries to detect the writing style of authors by analyzing the occurrences and variations of spelling errors ("idiosyncrasies") is proposed in [96]. It is based on the assumption that authors tend to make similar spelling and/or grammar errors, whereby nearly hundred different error features are considered. Examples of such measures are repeated/missing words, mismatched tenses, missing hyphens, repeated/inserted letters, single consonants instead of double, or the confusion of concrete letters like 'x' and 'y'. An evaluation on English email discussion group documents reached an accuracy of nearly 70% by solely using spelling idiosyncrasies with decision trees.

A historic example of the utilization of a specific misspelling of the word "touch" that has been used in court to identify the writer of a document is given by Wellman [199] and summarized in [80] as follows: *"He* [Wellman] *had noted this particular misspelling, and (under the guise of eliciting handwriting samples) was able to persuade the witness to write a phrase containing "touch" while on the witness stand. Showing that she had in fact written '*toutch,' Wellman was able to satisfy the court that was how she spelled that particular word, and hence that the document with that misspelling had been written by the witness."*

Originally coming from the data visualization field, the authors in [84] applied their techniques for the analysis of literary works. In this study, text characteristics like average sentence length, frequencies of function words or hapax legomena (words that occur exactly once) are visualized in order to provide a framework for users to immediately see outstanding facts. Besides an analysis of the Bible, also novels of Mark Twain have been visualized. Figure 6.4 shows an excerpt of the visualizations provided in the paper, where it can be seen that the book *"The adventures of Huckleberry Finn"* has significantly other characteristics than other works by the same author.

6.3. Automatic Author Profiling

The profiling of authors falls under the problem class usually referred to as text categorization [158], whereby also here an often applied concept is the utilization of different machine learning algorithms based on a previously selected set of features. The main problem types are differentiated between single-label and multi-label classification problems, respectively, where the first type assigns only one label for a document (e.g. the gender or age of the author) and



(a) Function words (b) Hapax legomena

Figure 6.4.: Visualization of Different Text Features. Source: [84].

the latter type is allowed to assign more labels (e.g. the content type of an article: sports, religion, science, etc.) [192].

Within the single-label text categorization problem the gender and age of the author of a text document has been analyzed frequently. Thereby the first attempts to distinguish between women and men were motivated by sociological studies (e.g. [18, 33]), which state that *"in postindustrial settings, women are often characterized as more emotionally extravagant, communicatively indirect, and solidarity seeking than men"*, whereby *"linguistic evidence provided in support of these views includes women's exploitation of [...] their frequent use of tag questions [...] and their frequent use of affect-enhancing linguistic indexes such as intensifying adverbs and modals"*. [18]

With the progress in text categorization and authorship attribution, many approaches also tried to automatically detect meta-information like the gender and age of authors, most often by reusing or adapting stylometric features that have been used in other fields. Beside this core information also many other characteristics have been profiled, including the level of education, the geographical origin or psychological types like extrovertism or neuroticism. In the following some examples of current profiling approaches are given.

Gender and Age

Based on the work of [95] that analyzes the gender of the author and also automatically distinguishes between fiction and non-fiction documents, the web blog corpus created by Schler et al. - which is also used in this thesis - has been created to classify gender and age based on many style and content features [156]. Beside basic features like the frequencies of function words, pronouns, determiners or the average number of words per post, also blogwords (neologisms) like *'lol'*, *'haha'* or *'ur'* as well as the frequency of hyperlinks have been analyzed. With a proposed so-called *Multi-Class Real Winnow* learning

algorithm, the gender of the authors of the web blogs could be profiled with an accuracy of 80%, and the age with an an accuracy of 76%, respectively. Similarly to the profiling results described in this thesis, the authors also report significant problems discriminating mid-twenty year olds from mid-thirty year olds.

An extension to the previous work that additionally attempts to classify the language and personality of a writer has been proposed in [8] by utilizing taxonomies of POS tags combined with other style and content-specific features. By using a Bayesian Multinomial Regression learning algorithm [52] on the same web blog corpus, 76% accuracy on gender and 77% accuracy on age could be gained.

Two new feature sets using POS tag patterns are proposed in [120] to enhance current state-of-the art profiling approaches. In simplified terms, the frequencies of POS-n-grams (where n is not fixed) are collected, rated in terms of significance and used as features if some conditions hold. An evaluation performed also on a (different) blog corpus, the effectiveness of the two new features has been tested. The best result using a support vector machine and incorporating the large number of nearly 24,000 features could enhance the previously described result of Schler et al. by 8%, i.e., reaching 88% on their data set.

In [24] the authors try to automatically expose the gender of writers of Twitter messages by incorporating the huge amount of over 15 million features. The origins of the features are thereby quite simple and can be categorized into character {1-5}-grams and word {1-2}-grams of the actual tweets, complemented by the corresponding n-grams of the user's profile information like the screen name (display name), full name or the profile description. Due to the large amount of features standard learning algorithms were not feasible, and therefore the authors used a parameter-optimized version of the Balanced Winnow algorithm [107]. As expected, the best result could be gained by using the full name n-grams, reaching 89% accuracy. Combining all types of n-grams, the result could be enhanced further by 3%. By analyzing the tweets only, i.e., without profile information, the gender could be attributed to about 75% correctly. In addition, the latter result has been compared to the manual prediction of humans - conducted through the Amazon Mechanical Turk framework - and it could be shown that the computer-based algorithm works significantly better than the human judgements, which achieved only an accuracy of about 60-66%.

An interesting approach that also analyzes the gender of web blog authors is presented in [201]. Besides commonly used features in the field of text categorization the focus has been laid on blog-specific features. The approach

thereby considers the usage of background colors, emoticons like `;-)` or `:-D`, punctuation marks or fonts. It is shown that the prediction of gender can be enhanced by using these features. Moreover, as a result from the experiment, a list of words which occur in male but rarely/not in female blogs (e.g. *"psst"*, *"income"* or *"wasup"*) and vice versa (e.g. *"muah"*, *"jewelry"* or *"kissme"*) is presented. On the other side, examples of the most gender-discriminant words of the study are: *"peace"*, *"shit"*, *"yo"*, *"man"*, *"fuck"*, *"damn"*.

Other Information

Many studies (e.g. [7, 109]) have analyzed the five psychological traits: neuroticism, extraversion, openness, agreeableness and conscientiousness. One key problem for verifying such approaches is thereby the lack of test data, i.e., the ground truth is always manually created and thus subjective to some extent. For example, for the evaluation in [140] psychology students have been asked to write a random essay within 20 minutes, whereby the categorization of personality has been made by filling out an additional questionnaire. In another paper [130] web blogs have been psychologically and gender-wise analyzed. Here, 71 bloggers have been asked to submit previously written text, and to additionally fill out a sociobiographic questionnaire as well as an online implementation of a psychological categorization test. By inspecting only eight different POS frequencies like the number of nouns, adjectives or articles, every personality trait of the authors could be predicted with an accuracy of 50-60% in this study. Additionally it has been found that openness is the only trait which has influence on the F-score, which may be correlating to the fact that openness is also seen as a factor of intellect and is thus more easily measurable [70].

English emails have been profiled into ten classes including gender, age, geographic origin or level of education as well as into the five psychological traits in [43]. The authors use several character-level, lexical and structural features and report a similar accuracy for gender classification as the outcome presented in this thesis, but show a worse result for age classification. But it has to be stressed that emails are typically significantly shorter than blogs, and thus the result should not be compared directly.

With the recent rise of social media networks, also content such as chat lines, Facebook postings or tweets have been analyzed and automatically profiled. It is shown (e.g. in [47] or [116]) that a well-defined set of style and content features can be used to expose meta information of chat logs, also in other languages such as Spanish. Nevertheless, the authors in [137] show that the application of common text categorization techniques using natural language processing is challenging - but possible - when facing highly limited data sets.

It is demonstrated that even for text samples containing only approximately 12 tokens, the classification of gender and age is feasible.

Beside gender and age (older/younger), also the political affiliation has been successfully profiled in [35]. For the study, manuscripts of parliament speeches of Swedish politicians have been used, whereby only speakers for whom more than 20 manuscripts exist have been taken into account. The political affiliation has been categorized manually into either left- or right-wing. By only utilizing word-based features extracted from the 200 first words of the speeches, the authors report an accuracy of nearly 90% for the political affiliation by using a support vector machine.

6.4. Text-Based Clustering

Most of the traditional document clustering approaches are based on occurrences of words, i.e., inverted indices are built and used to group documents. Thereby a unit to be clustered conforms exactly to one document. The main idea is often to compute topically related document clusters and to assist web search engines to be able to provide better results to the user, whereby the algorithms proposed frequently are also patented (e.g. [6, 27]). Regularly applied concepts in the feature extraction phase are the term frequency tf, which measures how often a word in a document occurs, and the term frequency-inverse document frequency $tf - idf$, which measures the significance of a word compared to the whole document collection. An example of a classical approach using these techniques is published in [100]. In this approach, a modified version from the original K-Means algorithm is used, which at first performs seed selection, i.e., searches for possible cluster centers, and subsequently assigns each document to one of the seeds. The approach is evaluated against different corpora containing from about 1,000 up to 9,000 documents. The performance is measured by the correct cluster assignment according to predefined topics, whereby an accuracy of 50% up to 80% could be gained, depending on the data set.

Beside the straight-forward computations of tf and $tf - idf$ weights, some algorithms also utilize other ideas like the analysis of documents which are linked by co-citations [27]. Another method to cluster documents by using suffix trees instead of simple word frequencies is proposed in [203]. After cleaning the document, suffix trees are calculated for words and phrases and used to create base clusters. In the final step, the latter are combined into final clusters, i.e., clusters are merged if they are similar according to some measures. With an accuracy of about 40% on a web document collection, it is shown that suffix trees perform better than common algorithms like K-Means (at least for the data set used).

The literature on cluster analysis within a single document to discriminate the authorships in a multi-author document like it is done in this thesis is surprisingly sparse. On the other hand, many approaches exist to separate a document into paragraphs of different topics, which are generally called *text segmentation* problems. In this domain, the algorithms often perform vocabulary analysis in various forms like word stem repetitions [141] or word frequency models [150], whereby *"methods for finding the topic boundaries include sliding window, lexical chains, dynamic programming, agglomerative clustering and divisive clustering"* [30]. Despite the given possibility to modify these techniques to also cluster by authors instead of topics, this is rarely done. In the following some of the existing methods are shortly summarized.

Probably one of the first approaches that uses stylometry to automatically detect boundaries of authors of collaboratively written text is proposed in [57]. Thereby the main intention was not to expose authors or to gain insight into the work distribution, but to provide a methodology for collaborative authors to equalize their style in order to achieve better readability. To extract the style of separated paragraphs, common stylometric features such as word/sentence lengths, POS tag distributions or frequencies of POS classes at sentence-initial and sentence-final positions are considered. An extensive experiment uses a data set which has been created by ten students that were asked to summarize an episode of a television episode, whereby one student described the first half and another student the second half. In conclusion, the authors state that styolmetric features can be used to find authorship boundaries, but that there has to be done additional research in order to increase the accuracy and informativeness.

In [63] the authors also tried to divide a collaborative text into different single-author paragraphs. In contrast to the previously described handmade corpus, a large data set has been computationally created by using (well-written) articles of an internet forum. At first, different neural networks have been utilized using several stylometric features. By using 90% of the data for training, the best network could achieve an F-score of 53% for multi-author documents on the remaining 10% of test data. In a second experiment, only letter-bigram frequencies are used as distinguishing features. Thereby an authorship boundary between paragraphs was marked if the cosine distance exceeded a certain threshold. This method reached an F-score of only 42%, and it is suspected that letter-bigrams are not suitable for the (short) paragraphs used in the evaluation.

A two-stage process to cluster Hebrew Bible texts by authorship is proposed in [94]. Because a first attempt to represent chapters only by bag-of-words led to negative results, the authors additionally incorporated sets of synonyms (which could be generated by comparing the original Hebrew texts with an

English translation). With a modified cosine-measure comparing these sets for given chapters, two core clusters are compiled by using the *ncut* algorithm [37]. In the second step, the resulting clusters are used as training data for a support vector machine, which finally assigns every chapter to one of the two core clusters by using the simple bag-of-words features tested earlier. Thereby it can be the case, that units originally assigned to one cluster are moved to the other one, depending on the prediction of the support vector machine. With this two-stage approach the authors report a good accuracy of about 80%, whereby it should be considered that the size of potential authors has been fixed to two in the experiment. Nevertheless, the authors state that their approach could be extended for more authors with less effort.

As a last example, a mathematical approach that splits a multi-author document into single-author paragraphs is presented in [53]. In the first step the document is divided into subsequences of consecutive sentences that are written by the same author. Roughly speaking, this is done with a stochastic generative model on the occurrences of words, where the maximum (log-joint)-likelihood is computed by applying Dijkstra's algorithm on finding paths. Thereby the 500 most frequent words, represented as binary vectors (word occurs / doesn't occur) constitute the input for the calculation. After the boundaries between authors are found, a clustering algorithm using eigenvectors of matrices is proposed to group together subsequences written by the same author. As a consequence, text segments can be attributed to the same author, even if the segments are widely apart from each other. Moreover, the number of clusters is determined automatically and has not to be predefined. An evaluation on manually created multi-author documents from blog-posts and writings of New York Times columnists yields an accuracy of 50-60%.

CHAPTER 7

Conclusion

Textual plagiarism is a frequently emerging problem in modern society, especially due to the fact that more and more text documents have been made publicly available through large literary databases or published work that can be found by search engines. Given the huge amounts of possible sources, it has become easier to find appropriate passages that can be reused, where on the other hand the automatic detection becomes harder.

As one possible countermeasure, this thesis introduced three intrinsic plagiarism algorithms, which solely investigate a given suspicious document. The key concept is thereby the analysis of the grammar syntax that is used by authors to formulate their sentences, in combination with the assumption that the syntax is used mostly unconsciously and that this information can be used to distinguish between writers. In other words, if the style of a document is largely consistent except for specific parts, the latter are likely to be written by a different author.

Once the grammar of authors can be automatically analyzed, the idea can also be applied to related problems like authorship attribution, author profiling and the decomposition of multi-author documents, which has been done in

this thesis. In the following a short summary of the contributions on the respective subjects is given.

Intrinsic Plagiarism Detection

In this thesis three different variants to intrinsically detect plagiarism in text documents have been proposed. The main idea is to split a document into single sentences and to analyze their grammatical structure. At first, the grammar trees of sentences are compared against each other, which results in distances that can be analyzed. If then a specific sentence differs significantly with respect to all other sentences, this sentence is marked as suspicious. To make the final prediction, also an algorithm has been proposed that combines single sentences into whole plagiarized sections according to different parameters. Second, a variant has been depicted which only uses part-of-speech tags, whereby the distance comparisons are based on dynamic programming algorithms. Finally, a third variant computes profiles from the grammar trees and uses sliding windows to compare portions of the text to the profile of the whole document. All variants have been evaluated using a state-of-the-art data set, revealing promising results, especially for the latter variant which outperforms current state-of-the-art algorithms.

Authorship Attribution

The idea of creating profiles from grammar trees of sentences has also been applied to the field of authorship attribution. Here, a profile of mostly occurring grammar patterns is calculated for each candidate author, and the resulting profiles are used to predict the author of an unlabeled document. The prediction has thereby been performed by either using respective distance metrics or by utilizing classification algorithms from the machine learning field. Experiments showed that authorship attribution works well by using both proposed methods, indicating small benefits for the classification algorithms.

Author Profiling

Machine learning algorithms have also been used with grammar profiles to estimate the gender and age of authors. Using a large blog data set, evaluations revealed promising results and indicate that the grammar of writers can be used to profile meta information from authors, but that there exist certain problems distinguishing specific age groups with this methodology.

Decomposition of Multi-Author Documents

Finally, the previous ideas have also been adapted to be able to detect different authorships in a collaboratively written document. By using grammar profiles

of paragraphs, different clustering algorithms have been evaluated to determine paragraph groups of same authorships. Evaluations showed that this approach is feasible, but detailed experiments indicate that the classification algorithms used in the authorship attribution and profiling domains perform better than the clustering algorithms.

In summary, all research questions stated in Section 1 can be answered with a short and simple "yes": it has been shown that solely the utilization of grammar syntax analysis like it has been done in this thesis can be used to identify plagiarism, assign authorships, profile authors and to identify the portions of different authors in a collaboratively written document. With respect to the fact that no other commonly used measures have been integrated to approach the respective problems, the results are very promising.

APPENDIX A

Appendix

A.1. Penn Treebank Tags

In the following tables the complete list of the Penn Treebank II Tag Set [110] is shown[1].

Tag	Description	Example
CC	conjunction, coordinating	and, or, but
CD	cardinal number	five, three, 13%
DT	determiner	the, a, these
EX	existential there	there were six boys
FW	foreign word	mais
IN	conjunction, subordinating or preposition	of, on, before, unless
JJ	adjective	nice, easy
JJR	adjective, comparative	nicer, easier
JJS	adjective, superlative	nicest, easiest
LS	list item marker	
MD	verb, modal auxillary	may, should
NN	noun, singular or mass	tiger, chair, laughter
NNS	noun, plural	tigers, chairs, insects
NNP	noun, proper singular	Germany, God, Alice
NNPS	noun, proper plural	we met two Christmases ago

[1] examples taken from http://www.clips.ua.ac.be/pages/mbsp-tags, visited March 2014

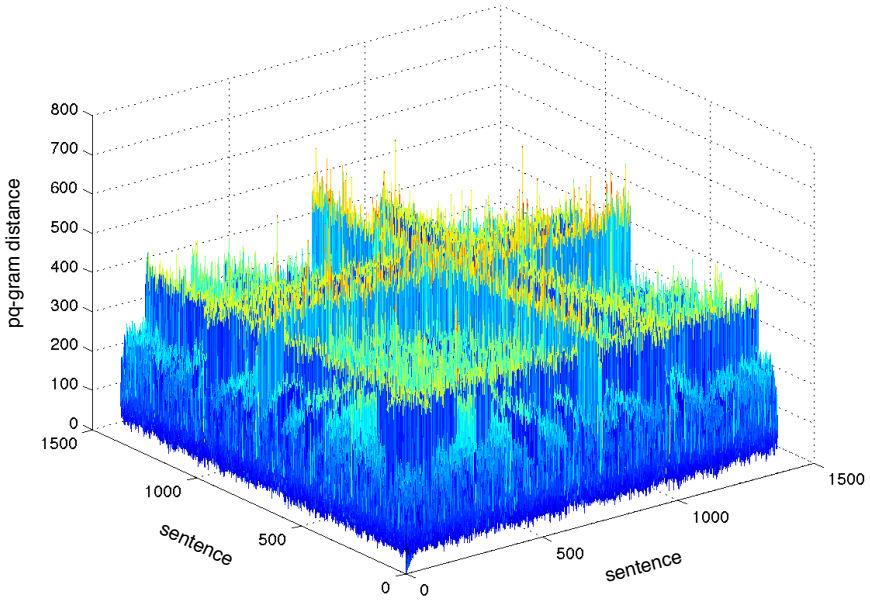
Tag	Description	Example
PDT	predeterminer	both his children
POS	possessive ending	's
PRP	pronoun, personal	me, you, it
PRP$	pronoun, possessive	my, your, our
RB	adverb	extremely, loudly, hard
RBR	adverb, comparative	better
RBS	adverb, superlative	best
RP	adverb, particle	about, off, up
SYM	symbol	%
TO	infinitival to	what to do?
UH	interjection	oh, oops, gosh
VB	verb, base form	think
VBZ	verb, 3rd person singular present	she thinks
VBP	verb, non-3rd person singular present	I think
VBD	verb, past tense	they thought
VBN	verb, past participle	a sunken ship
VBG	verb, gerund or present participle	thinking is fun
WDT	wh-determiner	which, whatever, whichever
WP	wh-pronoun, personal	what, who, whom
WP$	wh-pronoun, possessive	whose, whosever
WRB	wh-adverb	where, when
.	punctuation mark, sentence closer	.;?*
,	punctuation mark, comma	,
:	punctuation mark, colon	:
(	contextual separator, left paren	(
)	contextual separator, right paren	)

Table A.1.: Penn Treebank II Tag Set: Part-of-Speech Tags.

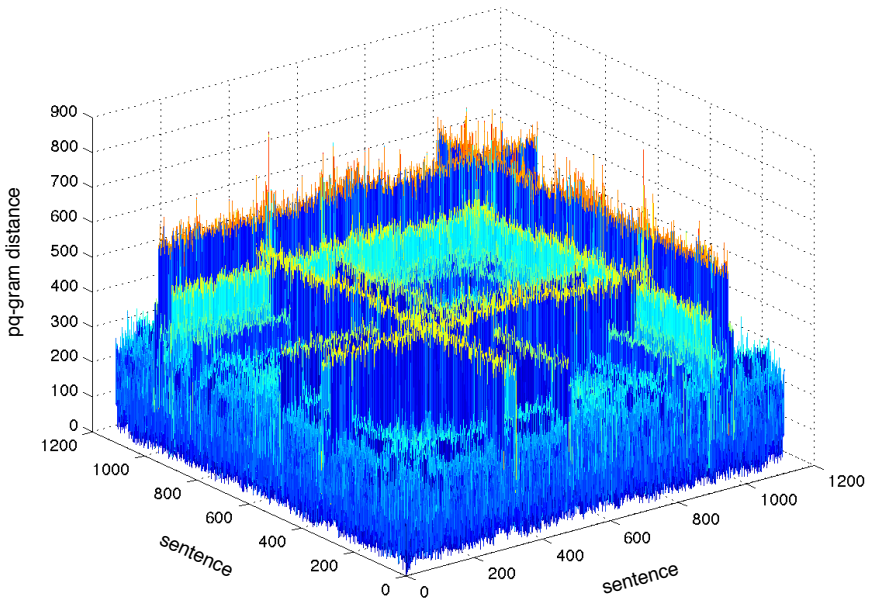
Tag	Description	Words	Example
ADJP	adjective phrase	CC+RB+JJ	warm and cosy
ADVP	adverb phrase	RB	also
INTJ	interjection	UH	hello
NP	noun phrase	DT+RB+JJ+NN + PR	the strange bird
PNP	prepositional noun phrase	PP+NP	as of today
PP	prepositional phrase	TO+IN	in between
PRT	particle	RP	up the stairs
SBAR	subordinating conjunction	IN	whether or not
VP	verb phrase	RB+MD+VB	was looking

Table A.2.: Penn Treebank II Tag Set: Chunk Tags.

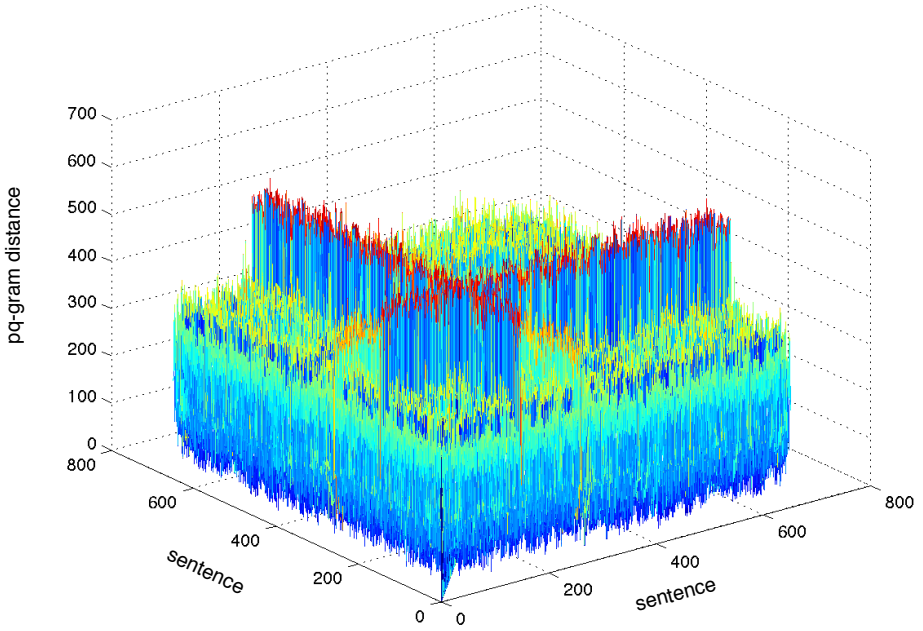
A.2. Plag-Inn: Examples of 3D Distance Matrix Visualizations



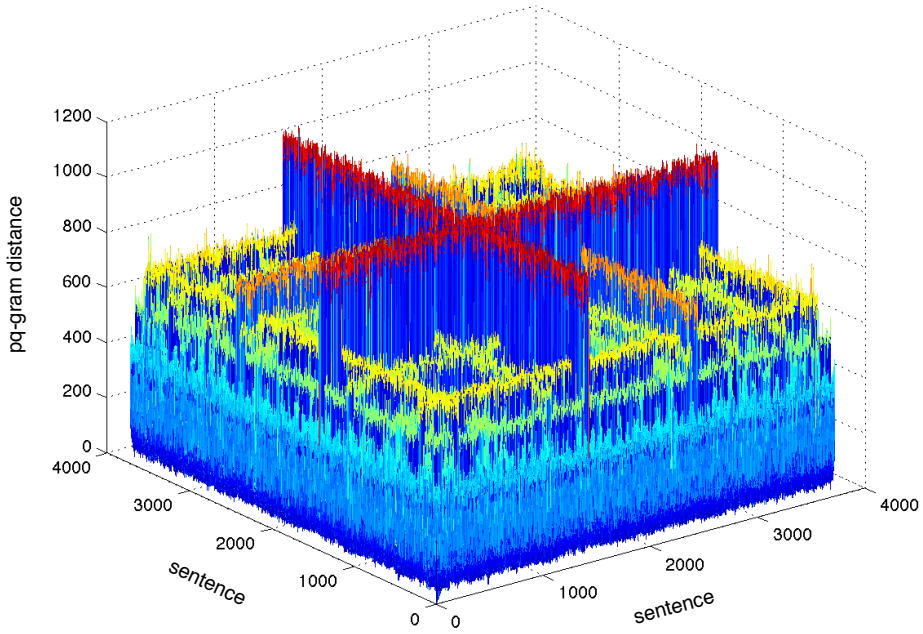
F-score: 72%



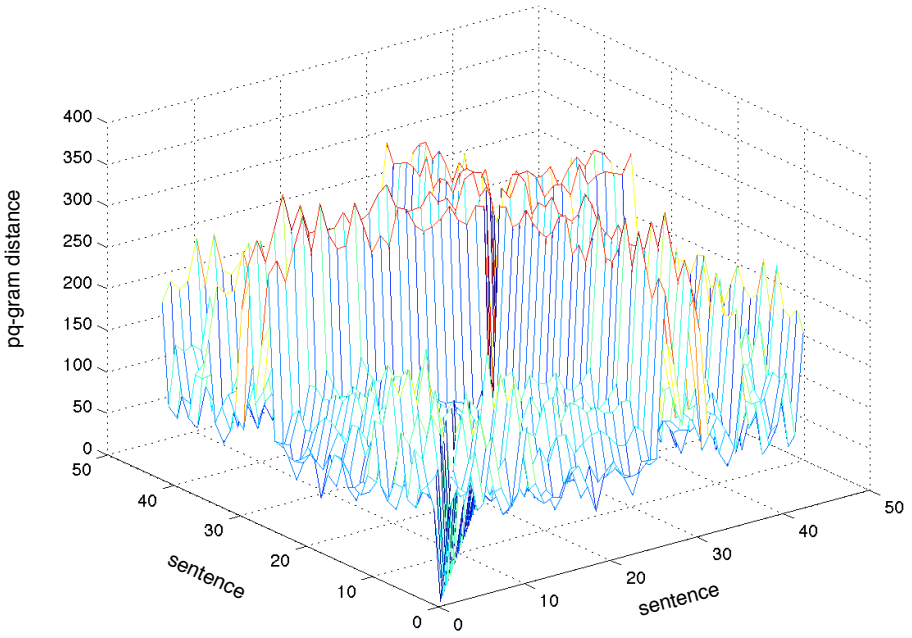
F-score: 49%



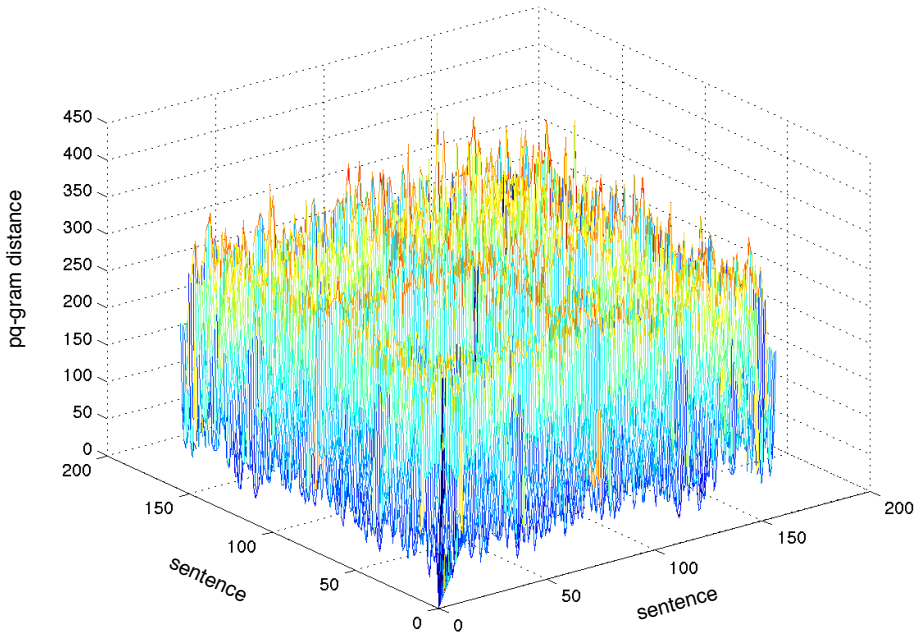
F-score: 27%



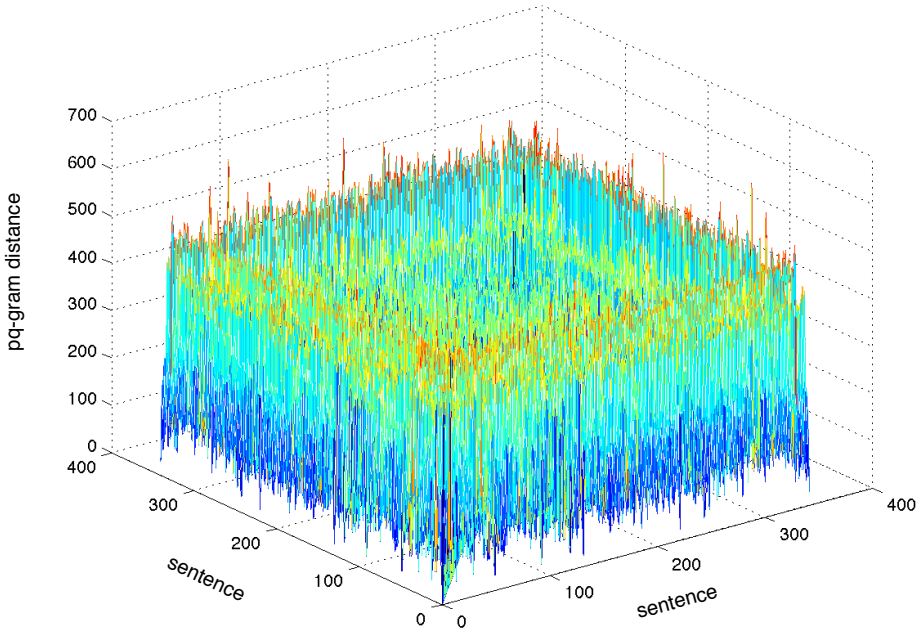
F-score: 12%



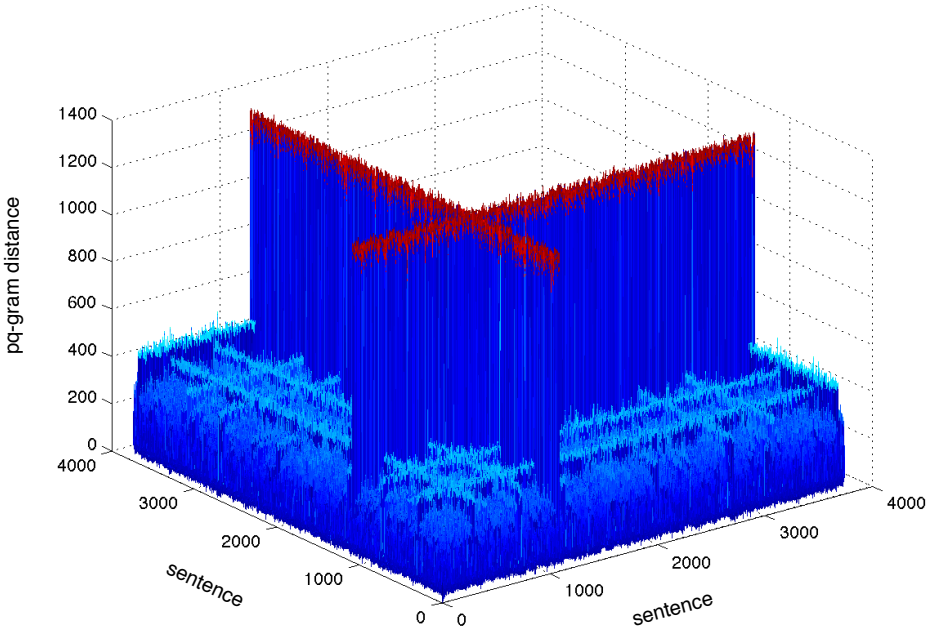
F-score: 100% (no plagiarism)



F-score: 100% (no plagiarism)



F-score: 0%
(contains no plagiarism, but the algorithm found two passages)



F-score: 0%
(contains no plagiarism, but the algorithm found one large passage, resulting from parsing/sentence splitting errors)

A.3. Plag-Inn: Sentence Selection

Using a maximum lookahead of $maxLookahead = 3$, the sentence selection procedure of the Plag-Inn algorithm creates the final result set of plagiarized sentences like it is shown in the following figures:

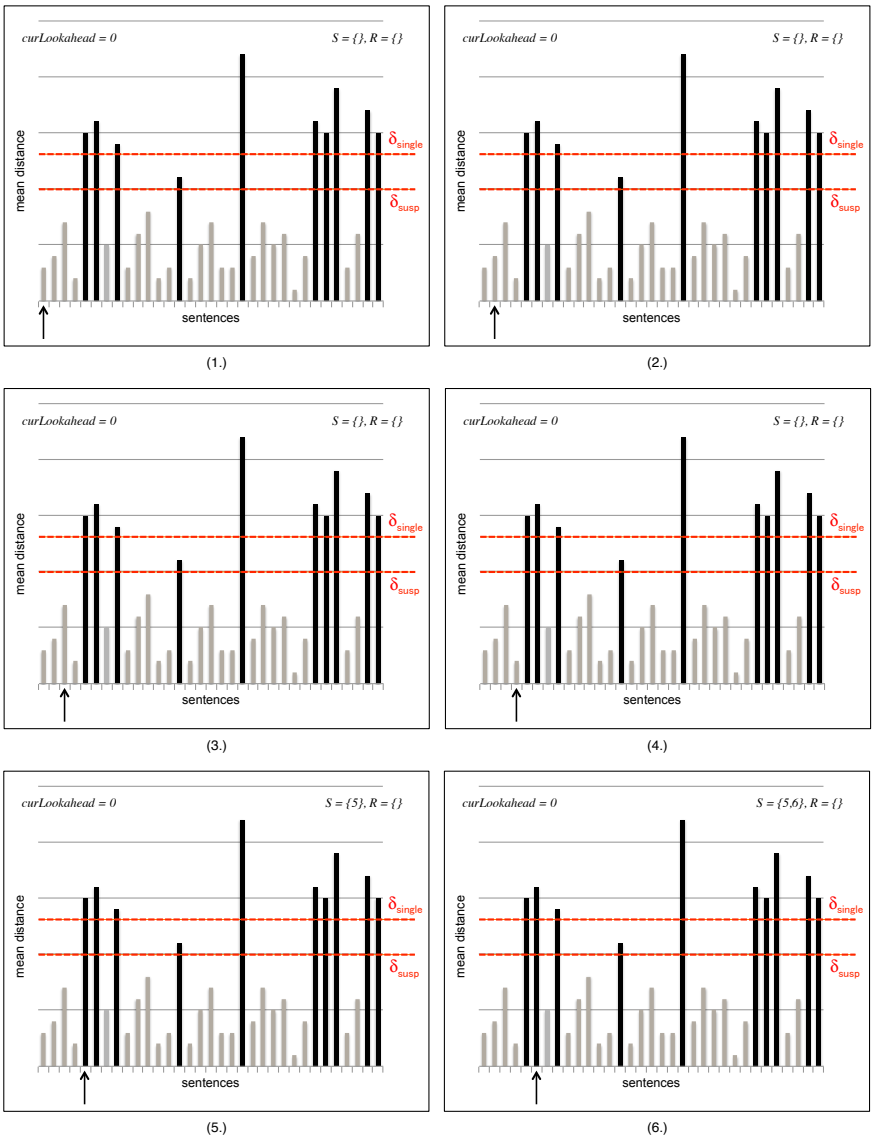


Figure A.1.: Example of the Sentence-Selection Algorithm (Steps 1-6).

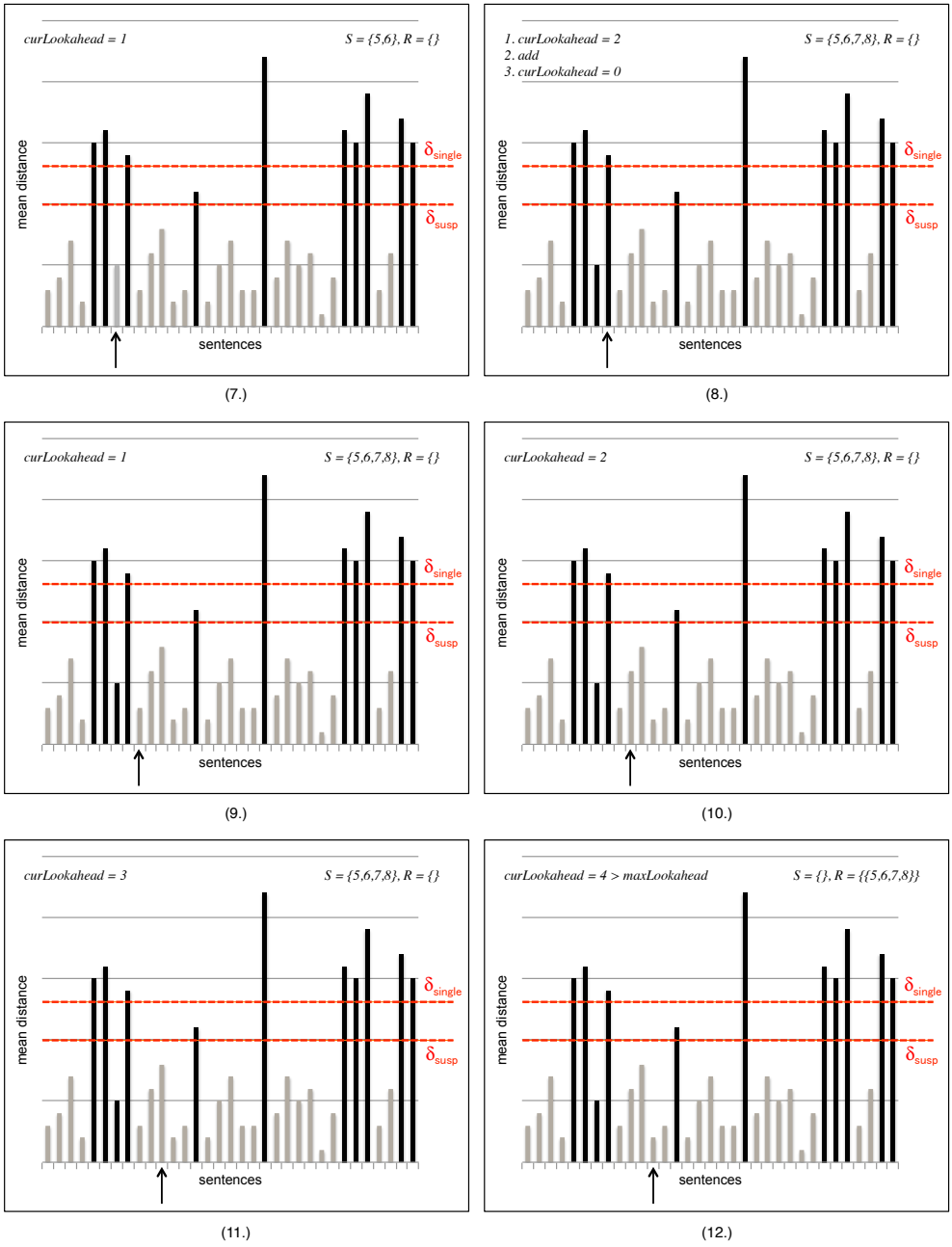


Figure A.2.: Example of the Sentence-Selection Algorithm (Steps 7-12).

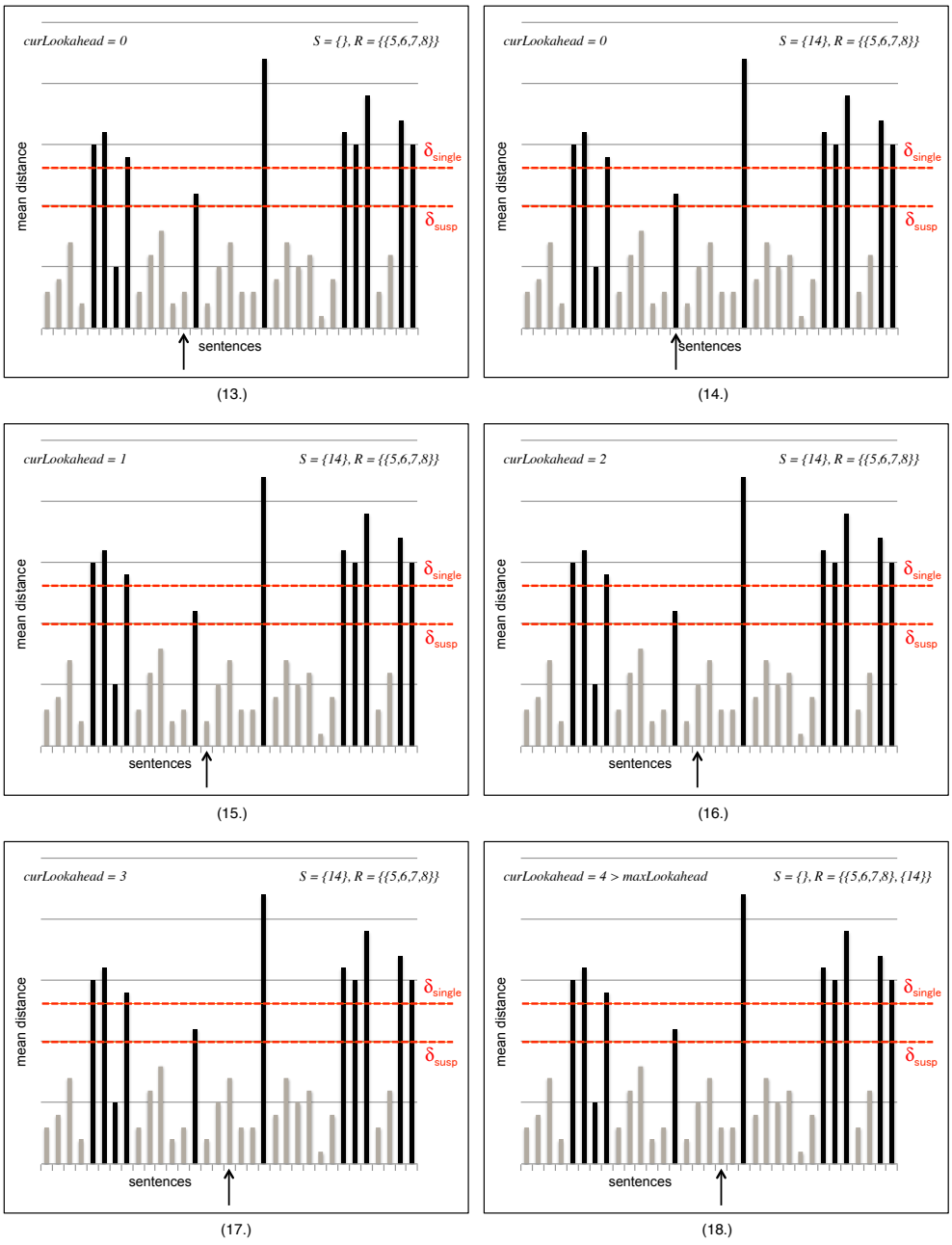


Figure A.3.: Example of the Sentence-Selection Algorithm (Steps 13-18).

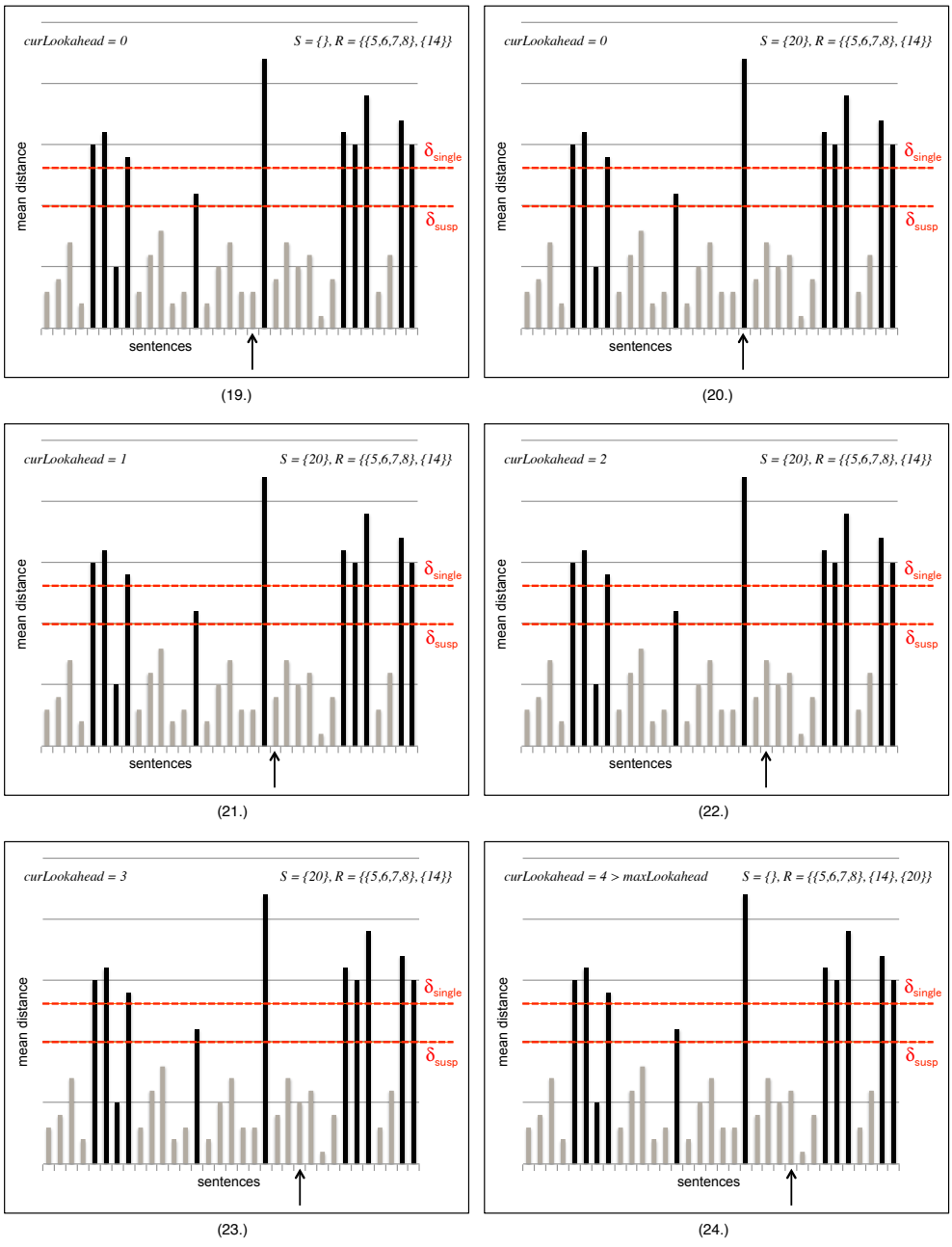


Figure A.4.: Example of the Sentence-Selection Algorithm (Steps 19-24).

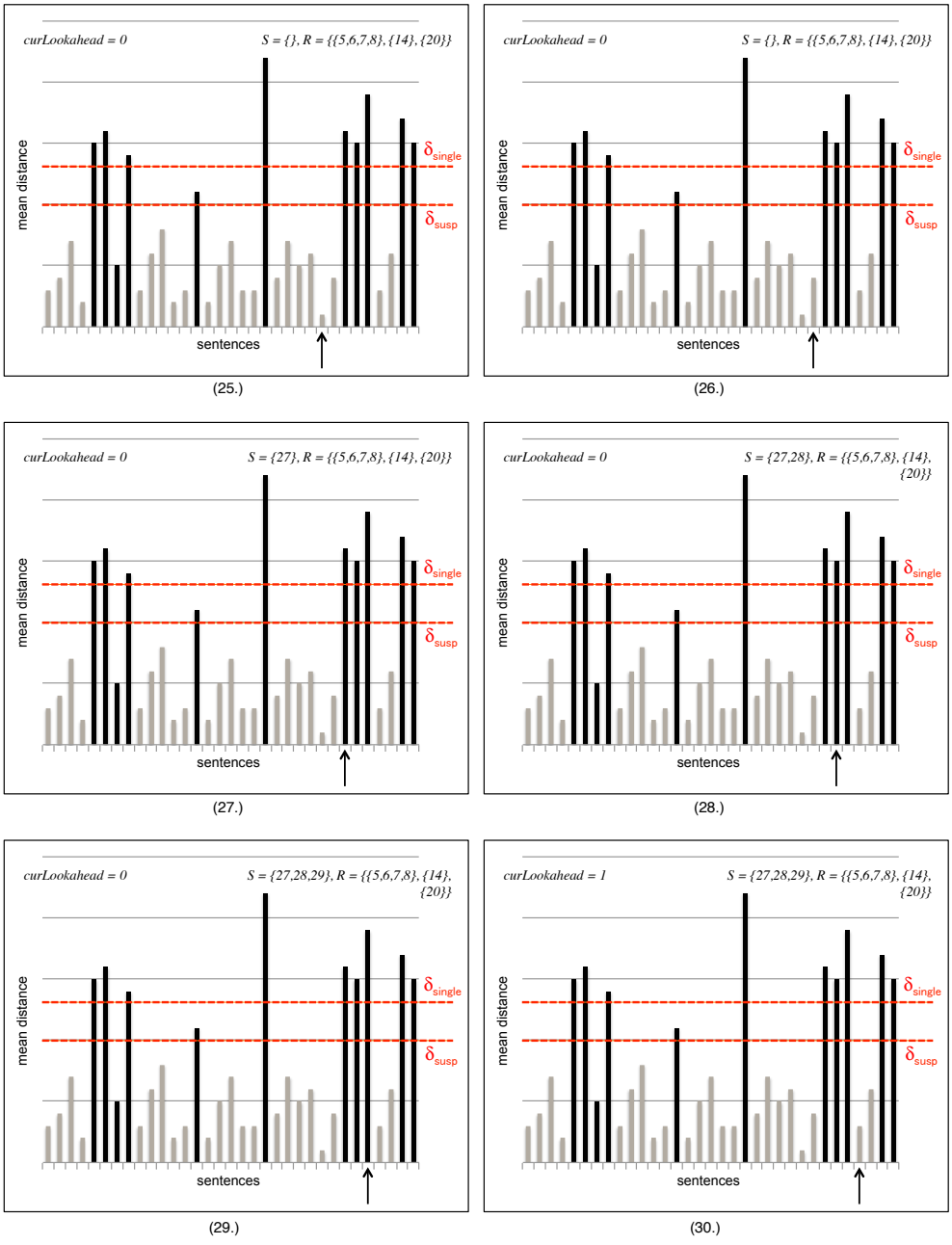


Figure A.5.: Example of the Sentence-Selection Algorithm (Steps 25-30).

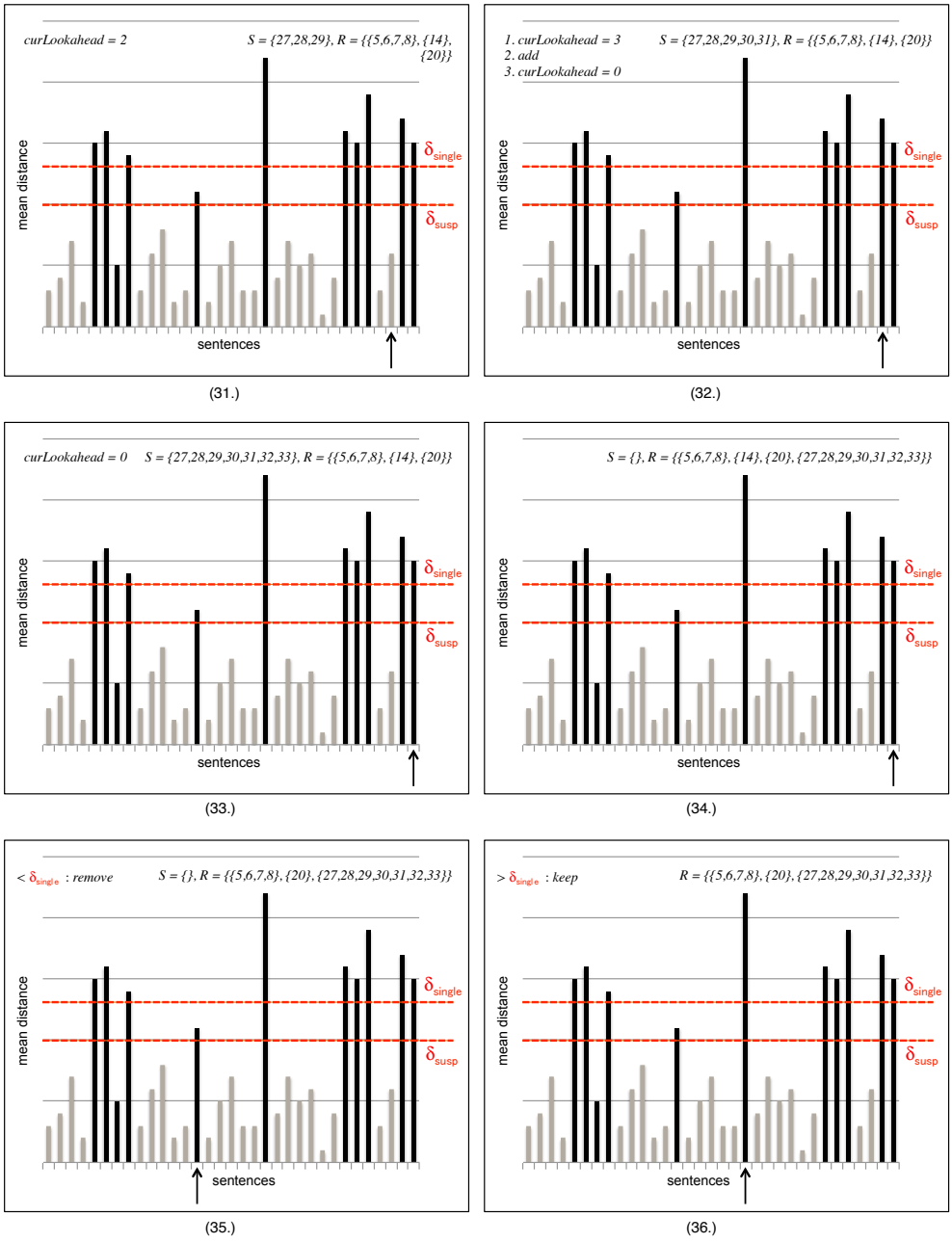


Figure A.6.: Example of the Sentence-Selection Algorithm (Steps 31-36).

List of Figures

Bibliography

[1] A. Abbasi and H. Chen. Applying Authorship Analysis to Extremist-group Web Forum Messages. *Intelligent Systems*, 20(5):67–75, 2005.

[2] E. E. Abdallah, A. E. Abdallah, M. Bsoul, A. F. Otoom, and E. Al-Daoud. Simplified Features for Email Authorship Identification. *International Journal of Security and Networks*, 8(2):72–81, 2013.

[3] S. Abney. Partial Parsing via Finite-State Cascades. *Natural Language Engineering*, 2(4):337–344, 1996.

[4] D. Aha and D. Kibler. Instance-Based Learning Algorithms. *Machine Learning*, 6:37–66, 1991.

[5] M. Anderka and B. Stein. Overview of the 1th International Competition on Quality Flaw Prediction in Wikipedia. In *Notebook Papers of the 7th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Rome, Italy, September 2012.

[6] C. Apte, S. M. Weiss, and B. F. White. Lightweight Document Clustering, November 2003. Google Patents, US Patent 6,654,739.

[7] S. Argamon, S. Dhawle, M. Koppel, and J. W. Pennebaker. Lexical Predictors Of Personality Type. In *Proceedings of the Joint Annual*

Meeting of the Interface and the Classification Society of North America, St. Louis, Missouri, USA, June 2005.

[8] S. Argamon, M. Koppel, J. W. Pennebaker, and J. Schler. Automatically Profiling the Author of an Anonymous Text. *Communications of the ACM*, 52(2):119–123, 2009.

[9] S. Argamon and S. Levitan. Measuring the Usefulness of Function Words for Authorship Attribution. In *Proceedings ot the 17th Joint International Conference on Humanities Computing and Digital Scholarship (ACH/ALLC)*, Victoria, BC, USA, June 2005.

[10] S. Argamon, M. Šarić, and S. S. Stein. Style Mining of Electronic Messages for Multiple Authorship Discrimination: First Results. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, Washington, DC, USA, August 2003. ACM, pages 475–480.

[11] D. Arthur and S. Vassilvitskii. K-means++: The Advantages of Careful Seeding. In *Proceedings of the 18th Annual Symposium on Discrete Algorithms (SODA)*, New Orleans, Louisiana, USA, 2007. ACM, Society for Industrial and Applied Mathematics, pages 1027–1035.

[12] N. Augsten, M. Böhlen, and J. Gamper. The pq-gram Distance Between Ordered Labeled Trees. *ACM Transactions on Database Systems (TODS)*, 35(1):4, 2010.

[13] H. Baayen, H. Van Halteren, and F. Tweedie. Outside the Cave of Shadows: Using Syntactic Annotation to Enhance Authorship Attribution. *Literary and Linguistic Computing*, 11(3):121–132, 1996.

[14] E. V. Balaguer. Putting Ourselves in SME's Shoes: Automatic Detection of Plagiarism by the WCopyFind tool. In *Proceedings of the 25th Conference of the Spanish Society for Natural Language Processing (SEPLN)*, 2009, pages 34–35.

[15] J.-P. Bao, J.-Y. Shen, X.-D. Liu, H.-Y. Liu, and X.-D. Zhang. Semantic Sequence Kin: A Method of Document Copy Detection. In *Advances in Knowledge Discovery and Data Mining*, volume 3056, pages 529–538. Springer Berlin, Heidelberg, 2004.

[16] K. Barker and N. Cornacchia. Using Noun Phrase Heads to Extract Document Keyphrases. In *Advances in Artificial Intelligence*, pages 40–52. Springer, 2000.

[17] L. Bergroth, H. Hakonen, and T. Raita. A Survey of Longest Common Subsequence Algorithms. In *Proceedings of the 7th International Symposium on String Processing and Information Retrieval (SPIRE)*, Santiago, Chile, October 2000. IEEE, pages 39–48.

[18] N. Besnier. Language and Affect. *Annual Review of Anthropology*, 19(1):419–451, 1990.

[19] Bibliographisches Institut GmbH. Zum Umfang des deutschen Wortschatzes. http://www.duden.de/sprachwissen/sprachratgeber/zum-umfang-des-deutschen-wortschatzes, visited March 2014.

[20] P. Bille. A Survey on Tree Edit Distance and Related Problems. *Theoretical Computer Science*, 337(1):217–239, 2005.

[21] J. N. G. Binongo and M. Smith. The Application of Principal Component Analysis to Stylometry. *Literary and Linguistic Computing*, 14(4):445–466, 1999.

[22] R. A. Bosch and J. A. Smith. Separating Hyperplanes and the Authorship of the Disputed Federalist Papers. *American Mathematical Monthly*, 105(7):601–608, 1998.

[23] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[24] J. D. Burger, J. Henderson, G. Kim, and G. Zarrella. Discriminating Gender on Twitter. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Edinburgh, UK, July 2011. Association for Computational Linguistics, pages 1301–1309.

[25] J. F. Burrows. Not Unless You Ask Nicely: The Interpretative Nexus Between Analysis and Information. *Literary and Linguistic Computing*, 7(2):91–109, 1992.

[26] T. Caliński and J. Harabasz. A Dendrite Method for Cluster Analysis. *Communications in Statistics - Theory and Methods*, 3(1):1–27, 1974.

[27] S. K. Card, J. D. Mackinlay, P. L. Pirolli, and J. E. Pitkow. Method and Apparatus for Clustering a Collection of Linked Documents Using Co-Citation Analysis, March 2000. Google Patents, US Patent 6,038,574.

[28] R. Caruana and D. Freitag. Greedy Attribute Selection. In *Proceedings of the 11th International Conference on Machine Learning (ICML)*, New Brunswick, NJ, USA, July 1994. pages 28–36.

[29] C.-C. Chang and C.-J. Lin. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[30] F. Y. Choi. Advances in Domain Independent Linear Text Segmentation. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference (NAACL)*, Seattle, Washington, USA, April 2000. Association for Computational Linguistics, pages 26–33.

[31] N. Chomsky. Three Models for the Description of Language. *IRE Transactions on Information Theory*, 2(3):113–124, 1956.

[32] G. G. Chowdhury. Natural Language Processing. *Annual Review of Information Science and Technology*, 37(1):51–89, 2003.

[33] J. Coates. *Women, Men, and Language: A Sociolinguistic Account of Gender Differences in Language*. Pearson Education, 2004.

[34] G. F. Cooper and E. Herskovits. A Bayesian Method for the Induction of Probabilistic Networks From Data. *Machine Learning*, 9(4):309–347, 1992.

[35] M. Dahllöf. Automatic Prediction of Gender, Political Affiliation, and Age in Swedish Politicians From the Wording of Their Speeches—A Comparative Study of Classifiability. *Literary and Linguistic Computing*, 27(2):139–153, 2012.

[36] S. Dasgupta. Performance Guarantees for Hierarchical Clustering. In *Computational Learning Theory*. Springer, 2002, pages 351–363.

[37] I. S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: Spectral Clustering and Normalized Cuts. In *Proceedings of the 10th International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, Seattle, Washington, USA, August 2004. ACM, pages 551–556.

[38] J. Diederich, J. Kindermann, E. Leopold, and G. Paass. Authorship Attribution With Support Vector Machines. *Applied Intelligence*, 19(1-2):109–123, 2003.

[39] M. Eder. Does Size Matter? Authorship Attribution, Small Samples, Big Problem. In *Proceedings of the Digital Humanities Conference*, London, UK, July 2010. ALLC.

[40] A. Eiselt, M. Potthast, B. Stein, P. Rosso, and A. Barrón-Cedeno. Overview of the 1st International Competition on Plagiarism Detection. In *Proceedings of the 3rd Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, 2009.

[41] S. R. El-Beltagy and A. Rafea. KP-Miner: A Keyphrase Extraction System for English and Arabic Documents. *Information Systems*, 34(1):132–144, 2009.

[42] V. Elizalde. Using Statistic and Semantic Analysis to Detect Plagiarism. *Notebook Papers of the 9th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN), Valencia, Spain, September*, September 2013.

[43] D. Estival, T. Gaustad, S. B. Pham, W. Radford, and B. Hutchinson. Author Profiling for English Emails. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, 2007, pages 263–272.

[44] A. Faktor and M. Irani. "Clustering by Composition"–Unsupervised Discovery of Image Categories. In *Proceedings of the 12th European Conference on Computer Vision (ECCV)*, pages 474–487. Springer, Florence, Italy, October 2012.

[45] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A Library for Large Linear Classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[46] D. Fisher. Knowledge Acquisition Via Incremental Conceptual Clustering. *Machine Learning*, 2(2):139–172, 1987.

[47] L. Flekova and I. Gurevych. Can We Hide in the Web? Large Scale Simultaneous Age and Gender Author Profiling in Social Media. In *Notebook Papers of the 7th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Rome, Italy, September 2012.

[48] G. Frantzeskou, E. Stamatatos, S. Gritzalis, and S. Katsikas. Effective Identification of Source Code Authors Using Byte-level Information. In

Proceedings of the 28th International Conference on Software Engineering, Shanghai, China, May 2006. ACM, pages 893–896.

[49] G. Fung. The Disputed Federalist Papers: SVM Feature Selection via Concave Minimization. In *Proceedings of the Conference on Diversity in Computing*, Atlanta, Georgia, USA, October 2003. ACM, pages 42–46.

[50] M. Gamon. Linguistic Correlates of Style: Authorship Classification With Deep Linguistic Analysis Features. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, Geneva, Switzerland, August 2004. Association for Computational Linguistics, page 611.

[51] W. Gassler, E. Zangerle, M. Tschuggnall, and G. Specht. SnoopyDB: Narrowing the Gap Between Structured and Unstructured Information Using Recommendations. In *Proceedings of the 21st Conference on Hypertext and Hypermedia (HT)*, Toronto, Canada, June 2010. ACM, pages 271–272.

[52] A. Genkin, D. D. Lewis, and D. Madigan. Large-Scale Bayesian Logistic Regression for Text Categorization. *Technometrics*, 49(3):291–304, 2007.

[53] C. Giannella. An Improved Algorithm for Unsupervised Decomposition of a Multi-Author Document. *Technical Papers, The MITRE Corporation*, February 2014.

[54] J. Gibbons. *Forensic Linguistics: An Introduction to Language in the Justice System*. Wiley-Blackwell, 2003.

[55] L. Gillam, N. Newbold, and N. Cooke. Educated Guesses and Equality Judgements: Using Search Engines and Pairwise Match for External Plagiarism Detection. In *Notebook Papers of the 7th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Rome, Italy, September 2012.

[56] D. Gillick. Sentence Boundary Detection and the Problem With the US. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), Companion Volume: Short Papers*, Boulder, Colorado, USA, June 2009. Association for Computational Linguistics, pages 241–244.

[57] A. Glover and G. Hirst. Detecting Stylistic Inconsistencies in Collaborative Writing. In *The New Writing Environment*, pages 147–168. Springer, 1996.

[58] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[59] G. H. Gonnet, R. A. Baeza-Yates, and T. Snider. New Indices for Text: Pat Trees and Pat Arrays. Published online: http://orion.lcg.ufrj.br/Dr.Dobbs/books/book5/chap05.htm, 1992.

[60] M. Goormachtigh. How Old is English? http://www.proto-english.org/sum1.html, visited March 2014.

[61] T. Gottron. External Plagiarism Detection Based on Standard IR Technology and Fast Recognition of Common Subsequences. In *Notebook Papers of the 4th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Padua, Italy, September 2010.

[62] T. Gottron and N. Lipka. A Comparison of Language Identification Approaches on Short, Query-Style Texts. In *Proceedings of the 32nd European Conference on Advances in Information Retrieval (ECIR)*, Milton Keynes, UK, March 2010. Springer, pages 611–614.

[63] N. Graham, G. Hirst, and B. Marthi. Segmenting Documents by Stylistic Character. *Natural Language Engineering*, 11(04):397–415, 2005.

[64] J. Grieve. Quantitative Authorship Attribution: An Evaluation of Techniques. *Literary and Linguistic Computing*, 22(3):251–270, 2007.

[65] I. Guyon and A. Elisseeff. An Introduction to Variable and Feature Selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[66] O. Haggag and S. El-Beltagy. Plagiarism Candidate Retrieval Using Selective Query Formulation and Discriminative Query Scoring. In *Notebook Papers of the 9th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Valencia, Spain, September 2013.

[67] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA Data Mining Software: an Update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[68] M. A. Hearst. TextTiling: Segmenting Text Into Multi-paragraph Subtopic Passages. *Computational Linguistics*, 23(1):33–64, 1997.

[69] S. Helmer, N. Augsten, and M. Böhlen. Measuring Structural Similarity of Semistructured Data Based on Information-theoretic Approaches. *The International Journal on Very Large Data Bases (VLDB-Journal)*, 21(5):677–702, 2012.

[70] F. Heylighen and J.-M. Dewaele. Variation in the Contextuality of Language: An Empirical Measure. *Foundations of Science*, 7(3):293–340, 2002.

[71] G. Hirst and O. Feiguina. Bigrams of Syntactic Labels for Authorship Discrimination of Short Texts. *Literary and Linguistic Computing*, 22(4):405–417, 2007.

[72] D. I. Holmes. Authorship Attribution. *Computers and the Humanities*, 28(2):87–106, 1994.

[73] D. I. Holmes. The Evolution of Stylometry in Humanities Scholarship. *Literary and Linguistic Computing*, 13(3):111–117, 1998.

[74] D. I. Holmes and R. S. Forsyth. The Federalist Revisited: New Directions in Authorship Attribution. *Literary and Linguistic Computing*, 10(2):111–127, 1995.

[75] G. Inches and F. Crestani. Overview of the International Sexual Predator Identification Competition. In *Notebook Papers of the 7th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Rome, Italy, September 2012.

[76] G. Jäger and J. Rogers. Formal Language Theory: Refining the Chomsky Hierarchy. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1598):1956–1970, 2012.

[77] A. Jayapal. Similarity Overlap Metric and Greedy String Tiling. In *Notebook Papers of the 9th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Valencia, Spain, September 2013.

[78] G. H. John and P. Langley. Estimating Continuous Distributions in Bayesian Classifiers. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, Montreal, Canada, August 1995. Morgan Kaufmann Publishers Inc., pages 338–345.

[79] A. Joshi and Y. Schabes. Tree-Adjoining Grammars. *Handbook of Formal Languages*, 3:69–123, 1997.

[80] P. Juola. Authorship Attribution. *Foundations and Trends in Information Retrieval*, 1(3):233–334, 2006.

[81] P. Juola. An Overview of the Traditional Authorship Attribution Subtask. In *Notebook Papers of the 7th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Rome, Italy, September 2012.

[82] P. Juola and J. Sofko. Proving and Improving Authorship Attribution Technologies. In *Proceedings of the Canadian Symposium for Text Analysis (CaSTA)*, Hamilton, ON, Canada, 2004. pages 1–5.

[83] P. Juola and E. Stamatatos. Overview of the Author Identification Task at PAN 2013. In *Notebook Papers of the 9th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Valencia, Spain, September 2013.

[84] D. A. Keim and D. Oelke. Literature Fingerprinting: A New Method for Visual Literary Analysis. In *Proceedings of the Symposium on Visual Analytics Science and Technology (VAST)*. IEEE, 2007, pages 115–122.

[85] J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. Springer New York, 1976.

[86] V. Kešelj, F. Peng, N. Cercone, and C. Thomas. N-gram-Based Author Profiles for Authorship Attribution. In *Proceedings of the Conference Pacific Association for Computational Linguistics (PACLING)*, volume 3, Harifax, Canada, 2003. pages 255–264.

[87] M. Kestemont, K. Luyckx, and W. Daelemans. Intrinsic Plagiarism Detection Using Character Trigram Distance Scores. In *Notebook Papers of the 5th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Amsterdam, The Netherlands, September 2011.

[88] D. V. Khmelev and F. J. Tweedie. Using Markov Chains for Identification of Writers. *Literary and Linguistic Computing*, 16(3):299–307, 2001.

[89] B. Kjell, W. Woods, and O. Frieder. Discrimination of Authorship Using Visualization. *Information Processing & Management*, 30(1):141–150, 1994.

[90] D. Klein and C. D. Manning. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics, Volume 1*, Sapporo, Japan, 2003. Association for Computational Linguistics, pages 423–430.

[91] R. Kohavi. The Power of Decision Tables. In *Proceedings of the 8th European Conference on Machine Learning (ECML)*, pages 174–189. Springer, Heraclion, Greece, April 1995.

[92] T. Kohonen. *Self-organizing Maps, 3rd Edition*. Springer, 2001.

[93] A. N. Kolmogorov. On Tables of Random Numbers. *Theoretical Computer Science - Special Issue: Kolmogorov Complexity*, 207(2):387–395, November 1998.

[94] M. Koppel, N. Akiva, I. Dershowitz, and N. Dershowitz. Unsupervised Decomposition of a Document Into Authorial Components. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (HLT), Volume 1*, Portland, Oregon, June 2011. Association for Computational Linguistics, pages 1356–1364.

[95] M. Koppel, S. Argamon, and A. R. Shimoni. Automatically Categorizing Written Texts by Author Gender. *Literary and Linguistic Computing*, 17(4):401–412, 2002.

[96] M. Koppel and J. Schler. Exploiting Stylistic Idiosyncrasies for Authorship Attribution. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, volume 69, Acapulco, Mexico, August 2003. pages 72–80.

[97] M. Koppel, J. Schler, and S. Argamon. Computational Methods in Authorship Attribution. *Journal of the American Society for Information Science and Technology*, 60(1):9–26, 2009.

[98] M. Koppel, J. Schler, and D. Mughaz. Text Categorization for Authorship Verification. In *Proceedings of the 8th International Symposium on Artificial Intelligence and Mathematics*, Fort Lauderdale, Florida, USA, January 2004.

[99] A. Kornai. Natural Languages and the Chomsky Hierarchy. In *Proceedings of the 2nd Conference on European Chapter of the Association for Computational Linguistics (EACL)*, Geneva, Switzerland, 1985. Association for Computational Linguistics, pages 1–7.

[100] B. Larsen and C. Aone. Fast and Effective Text Mining Using Linear-time Document Clustering. In *Proceedings of the 5th international Conference on Knowledge Discovery and Data Mining (SIGKDD)*, San Diego, CA, USA, August 1999. ACM, pages 16–22.

[101] T. Lee, J. Chae, K. Park, and S. Jung. CopyCaptor: Plagiarized Source Retrieval System using Global Word Frequency and Local Feedback. In *Notebook Papers of the 9th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Valencia, Spain, September 2013.

[102] K. Leilei, Q. Haoliang, W. M. Du Cuixia, and H. Zhongyuan. Approaches for Source Retrieval and Text Alignment of Plagiarism Detection. In *Notebook Papers of the 9th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Valencia, Spain, September 2013.

[103] K. Leilei, Q. Haoliang, W. Shuai, D. Cuixia, W. Suhong, and H. Yong. Approaches for Candidate Document Retrieval and Detailed Comparison of Plagiarism Detection. In *Notebook Papers of the 7th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Rome, Italy, September 2012.

[104] V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. In *Soviet Physics Doklady*, volume 10, 1966, page 707.

[105] B. Levin. *English Verb Classes and Alternations : A Preliminary Investigation*. University Of Chicago Press, September 1993.

[106] S.-W. Lin, Z.-J. Lee, S.-C. Chen, and T.-Y. Tseng. Parameter Determination of Support Vector Machine and Feature Selection Using Simulated Annealing Approach. *Applied Soft Computing*, 8(4):1505–1512, 2008.

[107] N. Littlestone. Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm. *Machine Learning*, 2(4):285–318, 1988.

[108] Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma. A Survey of Content-Based Image Retrieval with High-Level Semantics. *Pattern Recognition*, 40(1):262–282, 2007.

[109] F. Mairesse, M. A. Walker, M. R. Mehl, and R. K. Moore. Using Linguistic Cues for the Automatic Recognition of Personality in Conversation and Text. *Journal of Artificial Intelligence Research*, 30:457–500, 2007.

[110] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330, June 1993.

[111] Y. Marton, N. Wu, and L. Hellerstein. On Compression-Based Text Classification. In *Advances in Information Retrieval*, pages 300–314. Springer, 2005.

[112] S. Massung, C. Zhai, and J. Hockenmaier. Structural Parse Tree Features for Text Representation. In *Proceedings of the 7th International Conference on Semantic Computing (ICSC)*, Irvine, California, USA, September 2013. IEEE, pages 9–16.

[113] H. A. Maurer, F. Kappe, and B. Zaka. Plagiarism - A Survey. *Journal of Universal Computer Science*, 12(8):1050–1084, 2006.

[114] R. Mayer, T. Lidy, and A. Rauber. The Map of Mozart. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR)*, Victoria, Canada, October 2006. pages 351–352.

[115] A. McCallum and K. Nigam. A Comparison of Event Models for Naive Bayes Text Classification, 1998.

[116] M. Meina, K. Brodzinska, B. Celmer, M. Czoków, M. Patera, J. Pezacki, and M. Wilk. Ensemble-based Classification for Author Profiling Using Various Features. *Notebook Papers of the 9th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, September 2013.

[117] H. Misra, F. Yvon, O. Cappé, and J. Jose. Text Segmentation: A Topic Modeling Perspective. *Information Processing & Management*, 47(4):528–544, 2011.

[118] T. K. Moon. The Expectation-Maximization Algorithm. *Signal Processing Magazine*, 13(6):47–60, 1996.

[119] F. Mosteller and D. Wallace. *Inference and Disputed Authorship: The Federalist*. Addison-Wesley, 1964.

[120] A. Mukherjee and B. Liu. Improving Gender Classification of Blog Authors. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, MIT, Massachusetts, USA, October 2010. Association for Computational Linguistics, pages 207–217.

[121] F. Murtagh. A Survey of Recent Advances in Hierarchical Clustering Algorithms. *The Computer Journal*, 26(4):354–359, 1983.

[122] M. Nagao and S. Mori. A New Method of n-gram Statistics for Large Number of n and Automatic Extraction of Words and Phrases From Large Text Data of Japanese. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING), Volume 1*, Kyoto, Japan, August 1994. Association for Computational Linguistics, pages 611–615.

[123] Nativlang.com. X-Bar Theory, Parse Trees and Ambiguity. http://www.nativlang.com/linguistics/grammar-xbar-lessons.php, visited March 2014.

[124] S. B. Needleman and C. D. Wunsch. A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. *Journal of Molecular Biology*, 48(3):443–453, Mar. 1970.

[125] J.-Y. Nie, J. Gao, J. Zhang, and M. Zhou. On the Use of Words and n-grams for Chinese Information Retrieval. In *Proceedings of the 5th International Workshop on Information Retrieval with Asian Languages*, Hongkong, China, October 2000. ACM, pages 141–148.

[126] S. Niezgoda and T. P. Way. SNITCH: A Software Tool for Detecting Cut and Paste Plagiarism. In *Proceedings of the 37th Technical Symposium on Computer Science Education (SIGCSE)*, Houston, Texas, USA, March 2006. ACM, pages 51–55.

[127] S. Nirkhi and R. Dharaskar. Comparative Study of Authorship Identification Techniques for Cyber Forensics Analysis. *International Journal of Advanced Computer Science & Applications*, 4(5), 2013.

[128] J. Noecker, M. Ryan, and P. Juola. Psychological Profiling Through Textual Analysis. *Literary and Linguistic Computing*, 28(3):382–387, 2013.

[129] A. Nourian. Submission to the 5th International Competition on Plagiarism Detection. *Notebook Papers of the 9th Evaluation Lab on Uncover-*

ing Plagiarism, Authorship and Social Software Misuse (PAN), Valencia, Spain, September, September 2013.

[130] S. Nowson, J. Oberlander, and A. J. Gill. Weblogs, Genres and Individual Differences. In *Proceedings of the 27th Annual Conference of the Cognitive Science Society*, Stresa, Italy, July 2005. Citeseer.

[131] G. Oberreuter, G. L'Huillier, S. A. Ríos, and J. D. Velásquez. FastDocode: Finding Approximated Segments of N-Grams for Document Copy Detection. In *Notebook Papers of the 4th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Padua, Italy, September 2010.

[132] G. Oberreuter, G. L'Huillier, S. A. Ríos, and J. D. Velásquez. Approaches for Intrinsic and External Plagiarism Detection. In *Notebook Papers of the 5th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Amsterdam, The Netherlands, September 2011.

[133] Oxford Dictionaries. How Many Words Are There in the English Language? http://www.oxforddictionaries.com/words/how-many-words-are-there-in-the-english-language, visited March 2014.

[134] Y. Palkovskii and A. Belov. Applying Specific Clusterization and Fingerprint Density Distribution with Genetic Algorithm Overall Tuning in External Plagiarism Detection. In *Notebook Papers of the 7th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Rome, Italy, September 2012.

[135] Y. Palkovskii and A. Belov. Using Hybrid Similarity Methods for Plagiarism Detection. In *Notebook Papers of the 9th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Valencia, Spain, September 2013.

[136] PAN Workshop. Uncovering Plagiarism, Authorship, and Social Software Misuse. http://pan.webis.de, visited June 2014.

[137] C. Peersman, W. Daelemans, and L. Van Vaerenbergh. Predicting Age and Gender in Online Social Networks. In *Proceedings of the 3rd International Workshop on Search and Mining User-Generated Contents*. ACM, 2011, pages 37–44.

[138] D. Pelleg, A. W. Moore, et al. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In *Proceedings of the*

17th International Conference on Machine Learning (ICML), Stanford, CA, USA, June 2000. pages 727–734.

[139] F. Peng, D. Schuurmans, S. Wang, and V. Keselj. Language Independent Authorship Attribution Using Character Level Language Models. In *Proceedings of the 10th Conference on the European Chapter of the Association for Computational Linguistics (EACL), Volume 1*, Budapest, Hungary, April 2003. Association for Computational Linguistics, pages 267–274.

[140] J. W. Pennebaker and L. A. King. Linguistic Styles: Language Use as an Individual Difference. *Journal of Personality and Social Psychology*, 77(6):1296, 1999.

[141] J. M. Ponte and W. B. Croft. Text Segmentation by Topic. In *Proceedings of the European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, Pisa, Italy, 1997. Springer, pages 113–125.

[142] M. Potthast, A. Eiselt, A. Barrón-Cedeño, B. Stein, and P. Rosso. Overview of the 3rd International Competition on Plagiarism Detection. In *Notebook Papers of the 5th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Amsterdam, The Netherlands, September 2011.

[143] M. Potthast, T. Gollub, M. Hagen, J. Kiesel, M. Michel, A. Oberländer, M. Tippmann, A. Barrón-Cedeno, P. Gupta, P. Rosso, et al. Overview of the 4th International Competition on Plagiarism Detection. In *Notebook Papers of the 7th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Rome, Italy, September 2012.

[144] M. Potthast, T. Gollub, M. Hagen, J. Kiesel, M. Michel, A. Oberländer, M. Tippmann, A. Barrón-Cedeno, P. Gupta, P. Rosso, et al. Overview of the 5th International Competition on Plagiarism Detection. In *Notebook Papers of the 9th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Valencia, Spain, September 2013.

[145] M. Potthast, M. Hagen, B. Stein, J. Graßegger, M. Michel, M. Tippmann, and C. Welsch. ChatNoir: A Search Engine for the ClueWeb09 Corpus. In *Proceedings of the 35th International Conference on Research and Development in Information Retrieval (SIGIR)*, Portland, Oregon, USA, August 2012. ACM, pages 1004–1004.

[146] M. Potthast, M. Hagen, M. Völske, and B. Stein. Crowdsourcing Interaction Logs to Understand Text Reuse From the Web. In *Proceedings of*

the 51st Annual Meeting of the Association of Computational Linguistics (ACL), Sofia, Bulgaria, August 2013.

[147] M. Potthast, B. Stein, A. Barrón-Cedeño, and P. Rosso. An Evaluation Framework for Plagiarism Detection. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, Beijing, China, August 2010. Association for Computational Linguistics.

[148] J. R. Quinlan. *C4.5: Programs for Machine Learning*. The Morgan Kaufmann Series in Machine Learning, 1993.

[149] F. Rangel, P. Rosso, M. Koppel, E. Stamatatos, and G. Inches. Overview of the Author Profiling Task at PAN 2013. In *Notebook Papers of the 9th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Valencia, Spain, September 2013.

[150] J. C. Reynar. Statistical Models for Topic Segmentation. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, College Park, Maryland, USA, June 1999. Association for Computational Linguistics, pages 357–364.

[151] M. D. Riley. Some Applications of Tree-Based Modeling to Speech and Language. In *Proceedings of the Workshop on Speech and Natural Language*, Cape Cod, Massachusetts, USA, 1989. Association for Computational Linguistics, pages 339–352.

[152] J. Rudman. The State of Authorship Attribution Studies: Some Problems and Solutions. *Computers and the Humanities*, 31(4):351–365, 1997.

[153] J. Rudman. The Non-Traditional Case for the Authorship of the Twelve Disputed Federalist Papers: A Monument Built on Sand. In *Proceedings ot the 17th Joint International Conference on Humanities Computing and Digital Scholarship (ACH/ALLC)*, Victoria, BC, USA, June 2005. pages 187–190.

[154] C. Sanderson and S. Guenter. Short Text Authorship Attribution via Sequence Kernels, Markov Chains and Author Unmasking: an Investigation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Sydney, Australia, 2006. pages 482–491.

[155] N. Scaringella, G. Zoia, and D. Mlynek. Automatic Genre Classification of Music Content: A Survey. *Signal Processing Magazine*, 23(2):133–141, 2006.

[156] J. Schler, M. Koppel, S. Argamon, and J. W. Pennebaker. Effects of Age and Gender on Blogging. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, volume 6, 2006, pages 199–205.

[157] L. Seaward and S. Matwin. Intrinsic Plagiarism Detection Using Complexity Analysis. In *Proceedings of the 25th Conference of the Spanish Society for Natural Language Processing (SEPLN)*, 2009, page 56.

[158] F. Sebastiani. Machine learning in Automated Text Categorization. *Computing Surveys*, 34(1):1–47, 2002.

[159] L. Shen. *Statistical LTAG Parsing*. PhD thesis, University of Pennsylvania, 2006.

[160] P. Shrestha and T. Solorio. Using a Variety of n-Grams for the Detection of Different Kinds of Plagiarism. In *Notebook Papers of the 9th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Valencia, Spain, September 2013.

[161] T. F. Smith and M. S. Waterman. Identification of Common Molecular Subsequences. *Journal of Molecular Biology*, 147(1):195–197, March 1981.

[162] G. Specht and B. Freitag. AMOS: A Natural Language Parser Implemented as a Deductive Database in LOLA. In *Applications of Logic Databases, Chapter 9*, pages 197–215. Springer, 1995.

[163] L. Spracklin, D. Inkpen, and A. Nayak. Using the Complexity of the Distribution of Lexical Elements as a Feature in Authorship Attribution. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC)*, Marrakech, Morocco, May 2008.

[164] L. Spracklin and L. V. Saxton. Filtering Spam Using Kolmogorov Complexity Estimates. In *Proceedings of the 21st International Conference on Advanced Information Networking and Applications (AINA)*, volume 1, Niagara Falls, Canada, May 2007. IEEE, pages 321–328.

[165] E. Stamatatos. Authorship Attribution Based on Feature Set Subspacing Ensembles. *International Journal on Artificial Intelligence Tools*, 15(05):823–838, 2006.

[166] E. Stamatatos. Author Identification Using Imbalanced and Limited Training Texts. In *Proceedings of the 18th International Workshop on*

Database and Expert Systems Applications (DEXA), Regensburg, Germany, 2007. IEEE, pages 237–241.

[167] E. Stamatatos. Author Identification: Using Text Sampling to Handle the Class Imbalance Problem. *Information Processing and Management*, 44(2):790–799, 2008.

[168] E. Stamatatos. A Survey of Modern Authorship Attribution Methods. *Journal of the American Society for Information Science and Technology*, 60(3):538–556, March 2009.

[169] E. Stamatatos. Intrinsic Plagiarism Detection Using Character n-gram Profiles. In *Notebook Papers of the 5th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Amsterdam, The Netherlands, September 2011.

[170] E. Stamatatos. Plagiarism Detection Using Stopword n-grams. *Journal of the American Society for Information Science and Technology*, 62(12):2512–2527, 2011.

[171] Statistic Brain Research Institute. Social Network Statistics. http://www.statisticbrain.com/social-networking-statistics, visited February 2014.

[172] B. Stein, N. Lipka, and P. Prettenhofer. Intrinsic Plagiarism Analysis. *Language Resources and Evaluation*, 45(1):63–82, 2011.

[173] B. Stein and S. M. zu Eissen. Intrinsic Plagiarism Analysis with Meta Learning. In *Proceedings of the SIGIR Workshop on Plagiarism Analysis, Authorship Attribution, and Near-Duplicate Detection*, Amsterdam, The Netherlands, July 2007. pages 45–50.

[174] B. Stein, S. M. zu Eissen, and M. Potthast. Strategies for Retrieving Plagiarized Documents. In *Proceedings of the 30th Annual International Conference on Research and Development in Information Retrieval (SIGIR)*, Amsterdam, The Netherlands, July 2007. ACM, pages 825–826.

[175] M. Stevenson and R. Gaizauskas. Experiments on Sentence Boundary Detection. In *Proceedings of the 6th Conference on Applied Natural Language Processing (ANLC)*, Seattle, Washington, 2000. Association for Computational Linguistics, pages 84–89.

[176] S. Suchomel, J. Kasprzak, M. Brandejs, et al. Three Way Search Engine Queries with Multi-feature Document Comparison for Plagiarism

Detection. In *Notebook Papers of the 7th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Rome, Italy, September 2012.

[177] Š. Suchomel, J. Kasprzak, M. Brandejs, et al. Diverse Queries and Feature Type Selection for Plagiarism Discovery. In *Notebook Papers of the 9th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Valencia, Spain, September 2013.

[178] L. Tanguy, F. Sajous, B. Calderone, and N. Hathout. Authorship Attribution: Using Rich Linguistic Features When Training Data is Scarce. In *Notebook Papers of the 7th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Rome, Italy, September 2012.

[179] The Internet Archive. Open Library. https://openlibrary.org, visited March 2014.

[180] The Project Gutenberg Literary Archive Foundation. Project Gutenberg. http://www.gutenberg.org, visited March 2014.

[181] The Stanford Parser. A Statistical Parser. http://nlp.stanford.edu/software/lex-parser.shtml, visited January 2012.

[182] D. A. R. Torrejón and J. M. M. Ramos. Text Alignment Module in CoReMo 2.1 Plagiarism Detector. In *Notebook Papers of the 9th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Valencia, Spain, September 2013.

[183] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-Rich Part-of-speech Tagging With a Cyclic Dependency Network. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL), Volume 1*, Edmonton, Canada, 2003. Association for Computational Linguistics, pages 173–180.

[184] M. Tschuggnall. Plag-Inn: Uncovering Plagiarism by Examining Author's Grammar Syntax. In M. Barden and A. Ostermann, editors, *Scientific Computing@uibk*. Innsbruck University Press, 2013.

[185] M. Tschuggnall and G. Specht. Plag-Inn: Intrinsic Plagiarism Detection Using Grammar Trees. In *Proceedings of the 18th International Conference on Applications of Natural Language to Information Sys-*

tems (NLDB), Groningen, The Netherlands, June 2012. Springer, pages 284–289.

[186] M. Tschuggnall and G. Specht. Countering Plagiarism by Exposing Irregularities in Authors' Grammar. In *Proceedings of the European Intelligence and Security Informatics Conference (EISIC)*, Uppsala, Sweden, August 2013. IEEE, pages 15–22.

[187] M. Tschuggnall and G. Specht. Detecting Plagiarism in Text Documents Through Grammar-Analysis of Authors. In *Proceedings of the 15th Fachtagung des GI-Fachbereichs Datenbanksysteme für Business, Technologie und Web (BTW)*, LNI, Magdeburg, Germany, March 2013. GI, pages 241–259.

[188] M. Tschuggnall and G. Specht. Using Grammar-Profiles to Intrinsically Expose Plagiarism in Text Documents. In *Proceedings of the 18th International Conference on Applications of Natural Language to Information Systems (NLDB)*, volume 7934 of *Lecture Notes in Computer Science*, Salford, UK, June 2013. pages 297–302.

[189] M. Tschuggnall and G. Specht. Automatic Decomposition of Multi-Author Documents Using Grammar Analysis. In *Proceedings of the 26th GI-Workshop on Grundlagen von Datenbanken*, Bozen, Italy, October 2014. CEUR-WS.

[190] M. Tschuggnall and G. Specht. Enhancing Authorship Attribution By Utilizing Syntax Tree Profiles. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL), volume 2: Short Papers*, Gothenburg, Sweden, April 2014. Association for Computational Linguistics, pages 195–199.

[191] M. Tschuggnall and G. Specht. What Grammar Tells About Gender and Age of Authors. In *Proceedings of the 4th International Conference on Advances in Information Mining and Management (IMMM)*, Paris, France, July 2014. pages 30–35.

[192] G. Tsoumakas and I. Katakis. Multi-Label Classification: An Overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13, 2007.

[193] F. J. Tweedie and R. H. Baayen. How Variable May a Constant Be? Measures of Lexical Richness in Perspective. *Computers and the Humanities*, 32(5):323–352, 1998.

[194] F. J. Tweedie, S. Singh, and D. I. Holmes. Neural Network Applications in Stylometry: The Federalist Papers. *Computers and the Humanities*, 30(1):1–10, 1996.

[195] Ö. Uzuner, B. Katz, and T. Nahnsen. Using Syntactic Information to Identify Plagiarism. In *Proceedings of the 2nd Workshop on Building Educational Applications Using NLP*, Stroudsburg, PA, USA, 2005. pages 37–44.

[196] O. Veselý, T. Foltýnek, and J. Rybička. Source Retrieval via Naïve Approach and Passage Selection Heuristics. In *Notebook Papers of the 9th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Valencia, Spain, September 2013.

[197] D. Weber-Wulff. Mogelpackung: Was leisten Plagiatserkennungssysteme? *Forschung und Lehre, Karriere-Praxis*, 2:140–141, 2011.

[198] D. Weber-Wulff and K. Köhler. Kopienjäger: Cloud-Software vs. menschliche Crowd in der Plagiaterkennung. *iX: Magazin für professionelle Informationstechnik*, 6:78–82, 2011.

[199] F. L. Wellman. *The Art of Cross Examination, Fourth Edition.* Simon and Schuster, 1936.

[200] K. Williams, H.-H. Chen, S. R. Choudhury, and C. L. Giles. Unsupervised Ranking for Plagiarism Source Retrieval. In *Notebook Papers of the 9th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Valencia, Spain, September 2013.

[201] X. Yan and L. Yan. Gender Classification of Weblog Authors. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, 2006, pages 228–230.

[202] C. U. Yule. *The Statistical Study of Literary Vocabulary.* Cambridge University Press, 1943.

[203] O. Zamir and O. Etzioni. Web Document Clustering: A Feasibility Demonstration. In *Proceedings of the 21st Annual International Conference on Research and Development in Information Retrieval (SIGIR)*, Melbourne, Australia, August 1998. ACM, pages 46–54.

[204] Y. Zhao and J. Zobel. Effective and Scalable Authorship Attribution Using Function Words. In *Proceedings of the Information Retrieval*

Technology: Second Asia Information Retrieval Symposium (AIRS). Springer, October 2005, pages 174–189.

[205] R. Zheng, J. Li, H. Chen, and Z. Huang. A Framework for Authorship Identification of Online Messages: Writing-Style Features and Classification Techniques. *Journal of the American Society for Information Science and Technology*, 57(3):378–393, 2006.

[206] J. Ziv and A. Lempel. A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.

[207] D. Zou, W.-J. Long, and Z. Ling. A Cluster-Based Plagiarism Detection Method. In *Notebook Papers of the 4th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Padua, Italy, September 2010.

[208] S. M. Zu Eissen and B. Stein. Intrinsic Plagiarism Detection. In *Proceedings of the 28th European Conference on Advances in Information Retrieval (ECIR)*, London, UK, April 2006. Springer, pages 565–569.

Errata

page 15: instead of the stated pq-gram index

$\mathcal{I}(T_a)$ = {

`*A*BC, *ABC*, *AC**,`	*(root node is `*`)*
`AB**D, AB*DE, ABDEF, ABEF*, ABF**,`	*(root node is `A`)*
`AC**G, AC*G*, AC**G,`	*(root node is `A`)*
`BD***,BE***,BF***,`	*(root node is `B`)*
`CG**H, CG*H*, CG**H,`	*(root node is `C`)*
`GH***`	*(root node is `G`)*

}

the correct index for $\mathcal{I}(T_a)$ is

$\mathcal{I}(T_a)$ = {

`*A**B, *A*BC, *ABC*, *AC**,`	*(root node is `*`)*
`AB**D, AB*DE, ABDEF, ABEF*, ABF**,`	*(root node is `A`)*
`AC**G, AC*G*, ACG**,`	*(root node is `A`)*
`BD***,BE***,BF***,`	*(root node is `B`)*
`CG**H, CG*H*, CGH**,`	*(root node is `C`)*
`GH***`	*(root node is `G`)*

}

page 17: Accordingly, instead of the stated pq-gram index

$\mathcal{I}(T_a)$ = {`*A*BC`, `*ABC*`, `*AC**`, `AB**D`, `AB*DE`, `ABDEF`, `ABEF*`, `ABF**`, `AC**G`, `AC*G*`, `AC**G`, `BD***`, `BE***`, `BF***`, `CG**H`, `CG*H*`, `CG**H`, `GH***`}

$\mathcal{I}(T_b)$ = {`*A**H`, `*A*HC`, `*AHC*`, `*AC**`, `AH**B`, `AH*B*`, `AHB**`, `AC**G`, `AC*G*`, `ACG**`, `HB**D`, `HB*DE`, `HBDE*`, `HBE**`, `CG**F`, `CG*FH`, `CGFH*`, `CGH**`, `BD***`, `BE***`}

the correct index is

$$\mathcal{I}(T_a) = \{\texttt{*A**B}, \texttt{*A*BC}, \texttt{*ABC*}, \texttt{*AC**}, \texttt{AB**D}, \texttt{AB*DE}, \texttt{ABDEF}, \texttt{ABEF*}, \texttt{ABF**}, \texttt{AC**G}, \texttt{AC*G*}, \texttt{ACG**}, \texttt{BD***}, \texttt{BE***}, \texttt{BF***}, \texttt{CG**H}, \texttt{CG*H*}, \texttt{CGH**}, \texttt{GH***}\}$$

$$\mathcal{I}(T_b) = \{\texttt{*A**H}, \texttt{*A*HC}, \texttt{*AHC*}, \texttt{*AC**}, \texttt{AH**B}, \texttt{AH*B*}, \texttt{AHB**}, \texttt{AC**G}, \texttt{AC*G*}, \texttt{ACG**}, \texttt{HB**D}, \texttt{HB*DE}, \texttt{HBDE*}, \texttt{HBE**}, \texttt{CG**F}, \texttt{CG*FH}, \texttt{CGFH*}, \texttt{CGH**}, \texttt{BD***}, \texttt{BE***}, \texttt{GF***}, \texttt{GH***}\}$$

Finally, instead of

$\lvert\mathcal{I}(T_a)\rvert = 18$	number of pq-grams in $\mathcal{I}(T_a)$
$\lvert\mathcal{I}(T_b)\rvert = 20$	number of pq-grams in $\mathcal{I}(T_b)$
$\lvert\mathcal{I}(T_a) \uplus \mathcal{I}(T_b)\rvert = 18 + 20$	multi-set union of the two indices
$\lvert\mathcal{I}(T_a) \upuplus \mathcal{I}(T_b)\rvert = 5$	multi-set intersection of the two indices (the number of pq-grams occurring in both indices)

$$dist^{pq}(T_a, T_b) = \lvert\mathcal{I}(T_a) \uplus \mathcal{I}(T_b)\rvert - 2 \cdot \lvert\mathcal{I}(T_a) \upuplus \mathcal{I}(T_b)\rvert = 18 + 20 - 2 \cdot 5 = 28$$

the correct pq-gram distance between the trees T_a and T_b is:

$\lvert\mathcal{I}(T_a)\rvert = 19$	number of pq-grams in $\mathcal{I}(T_a)$
$\lvert\mathcal{I}(T_b)\rvert = 22$	number of pq-grams in $\mathcal{I}(T_b)$
$\lvert\mathcal{I}(T_a) \uplus \mathcal{I}(T_b)\rvert = 19 + 22$	multi-set union of the two indices
$\lvert\mathcal{I}(T_a) \upuplus \mathcal{I}(T_b)\rvert = 7$	multi-set intersection of the two indices (the number of pq-grams occurring in both indices)

$$dist^{pq}(T_a, T_b) = \lvert\mathcal{I}(T_a) \uplus \mathcal{I}(T_b)\rvert - 2 \cdot \lvert\mathcal{I}(T_a) \upuplus \mathcal{I}(T_b)\rvert = 19 + 22 - 2 \cdot 7 = 27$$