Author identification in short texts

Marcia Fissette
0647721

Supervisor:
dr. F.A. Grootjen.

2010

Abstract

Most research on author identification considers large texts. Not many research is done on author identification for short texts, while short texts are commonly used since the rise of digital media. The anonymous nature of internet applications offers possibilities to use the internet for illegitimate purposes. In these cases, it can be very useful to be able to predict who the author of a message is. Van der Knaap and Grootjen [28] showed that authors of short texts can be identified using single words (word unigrams) with Formal Concept Analysis.

In theory, grammatical information can also be used as an indication of the author of the text. Grammatical information can be captured by word bigrams. Word bigrams are pairs of successive words, so they reveal some information on the sentence structure the author used. For this thesis I performed experiments using word bigrams as features for author identification to determine whether performance increases compared to using word unigrams as features. In most languages many grammatical relations within a sentence are between words that are not successive. The DUPIRA parser, a natural language parser for Dutch, produces dependency triplets that represent relations between non successive words, based on the Dutch grammar. I used these triplets as features, either alone or in combination with unigrams or bigrams. People often use smileys when communicating with someone using digital media. Therefore, I also examined the influence of smileys on author identification.

The messages used for the experiments are obtained from the subsection 'Eurovision Songfestival 2010' of the fok.nl message board. With these messages the data files for 7 feature sets were constructed: word unigrams excluding smileys, word unigrams including smileys, word bigrams excluding smileys, word bigrams including smileys, only dependency triplets, triplets+word unigrams, triplets+word bigrams. A support vector machine algorithm (SVM) was used as the classification method. This is a commonly used algorithm for author identification. There are different implementations of SVM. In this thesis SMO, LibSVM and LibLINEAR are compared. The LibLINEAR algorithm gave the best results.

The results revealed that in all conditions the performance is above chance level. So all reveal some information about the author. The performance for the word unigrams including smileys showed the best results, while the performance using the dependency triplets is the lowest. Results also revealed that when smileys are considered the performance increases, so smileys provide additional information about the author.

Contents

Introduction

People rely on the internet in their working environment as well as in their private life. They post messages anonymously on the internet, using e-mail, message boards and websites like Facebook and Twitter. These forms of contact are highly integrated in everyday life. The anonymous nature of internet applications offers possibilities to use the internet for illegitimate purposes. The authors of messages sent via email or submitted on message boards cannot always be identified by for example simply tracking an e-mail address. Messages may be routed via anonymous e-mail servers. All that is left for author identification is the message itself. For messages of which one knows that they were used for illegitimate purposes, knowing who the author was can contribute to solving the crime. Therefore, it would be very useful if one could predict with a predetermined level of certainty who the author of a message is, given a list of suspected authors.

Author identification is not a research area that emerged out of the increased use of internet. It was used for determining which author wrote a chapter or passage of a book, the bible being the most famous example. Author identification research makes use of the structure of the text and the words that are used. A subdivision of this is stylometric research in which linguistic characteristics are used to identify the author of a text. Actually, most of the features used for author identification are stylometric, especially in literary authorship. In stylometry research it is generally accepted that authors have unconsciously writing habits [4, 6, 7, 22]. These habits become evident in for example their use of words and grammar. The more unconscious a process is, the less controllable it is. Therefore words and grammar could be a reliable indicator of the author. These individual differences in use of language is referred to as idiolect. The unconscious use of syntax gives rise to the opportunity to perform author identification based on stylometric features.

A commonly used stylometric features is based on n-grams [1] of characters [5, 7, 22]. An example of a character n-gram is the character 2-gram, which is a sequence of two characters. The name 'bigram' is used to indicate a 2-gram. When using character bigrams, the text is split in all possible sequences of two characters. The (relative) frequency of each bigram can be used for classification. Experiments that use n-grams of characters have shown to be successful in determining the author of the text [5, 7, 22]. Also structural information is relevant for determining authorship, successful classifications are reported when using bigrams of syntactic labels [15, 22]. These studies focused on long texts. But most messages on the internet are short, with a maximum of about 200 words. Van der Knaap and Grootjen [28] showed that authors of messages in chat logs can be recognized using single words, also called word unigrams, and Formal Concept Analysis. These results indicate that authorship identification based on short texts is possible.

The aim of this project is to determine which types of information make it possible to identify the author of a short digital text. In particular, is it possible to identify the author of a short text based on the words and grammar used? This is actually a classification task. The features of the text decide to which author (category) the text belongs.

As said before Van der Knaap and Grootjen [28] showed that authors can be identified by the words the authors used with the method Formal Concept Analysis. For 2 out of 5 classification tests the method correctly classified the author (first place). On average the correct author achieved a second place. So the words used in a text are an indication of who wrote it. Because the use of grammar is subject to unconscious processes, grammatical information can also be used as an indication of the author of the text. Word bigrams are pairs of successive words, so they reveal some information on

[1]An n-gram is a sequence of n items. There are different types of n-grams: character n-grams, word n-grams and n-grams of syntactical labels. A word n-gram is a sequence of n words. For example a word 2-gram consists of 2 successive words. A 1-gram is often referred to as a unigram. A 2-gram is often called a bigram.

the sentence structure the author used. Words bigrams capture both the words and the grammar used. Classification performance may increase when the classification is based on bigrams of words instead of single words. However, many grammatical relations within a sentence are between words that are not successive. Another type of grammatical information is needed to capture these relations. Such relations can be captured by natural language processors (NLP). The classification performance may increase when grammatical structures extracted from a NLP is used compared to classification based on word unigrams or word bigrams. Does taking the grammatical structures into account contribute to identifying the author of the text?

In digital media people often communicate with smileys. This affects the messages communicated digitally. Since people have unconscious writing habits, their use of smileys may also be part of the unconscious process. So how many and which smileys are used in a message can be indicators of who the author is.

These ideas led to the following research questions:

- Does the classification performance increase when the classification is based on bigrams of words instead of single words?
- Does the classification performance increase when grammatical structures are used compared to classification based on word unigrams or word bigrams?
- Does the classification performance increase when smileys are taken into account?

The answers to these questions have scientific and societal relevance. From a scientific point of view it is relevant to know whether author identification based on short texts is possible. And, if this is possible, which factors contribute to a good classification performance. The answers to the research questions show whether the relationships between words, either because they are positioned next to each other or because they have a grammatical relation, are relevant for identifying who wrote them, and whether smileys influence the performance. More research is necessary to successfully use automatic author identification, especially for identifying the authors of short texts. The social relevance becomes evident when author identification can contribute to solving criminal activities. As discussed before, internet technologies can be used anonymously which makes it a suitable technique for criminal activities.

It should be noted that author identification techniques can not be used as scientific evidence. Short texts will not provide enough evidence. Chaski [4] explains why author identification results do not yet meet the criteria "empirical testing, known or potential rate of error, standard procedures for performing a technique, peer review and publication, as well as general acceptance in the scientific community" [4] to be considered as scientific evidence. A lot of research still needs to be done, so in the future author identification may become a reliable tool contributing to solving (cyber)crimes. Good author identification techniques can provide a top 5 list of possible authors. Such a list is useful for further investigation of the crime. It remains questionable whether a computer based tool can be the main source of evidence for solving a crime.

In the first chapter I will give an overview of previous research relevant to these topics and provide a background for the choices made for answering the research questions. In the second chapter I describe the methods used for executing the experiments. The third chapter gives an overview of the results. Chapter four, the conclusion, provides the answers to the questions. Finally, in the discussion I propose several interesting questions for future research.

Chapter 1

Background

In this chapter I will give an overview of previous research relevant for this thesis and explanations of choices made for examining the research questions. In the first section I describe the data and features that are used in previous author identification research and what results were achieved with those data and features. Because the focus of the research question is on the use of grammar, I will elaborate on previous author identification research in which grammatical features were used. For the experiments in this thesis a natural language parser for Dutch, DUPIRA, will be used. Therefore the output of this parser is also described in the first section. In Section 1.2 the features used in author identification research are summarized.

To identify the author of a text machine learning techniques are used. Section 1.3 contains descriptions of machine learning techniques that were used in the previous author identification research. I will use this examination to decide which algorithm to use for the experiments in this thesis.

The shortcomings and choices made in the previous author identification research influences the choices I made for executing the experiments. These choices are described in Section 1.4

1.1 Previous research

In author identification research different aspects can influence the performance of the author classification task. These aspects are the language of the messages used, the length of these messages, the number of authors and messages, the types of features and the classification method. Before executing experiments the researchers make choices on these aspects. I will describe previous research to illustrate the choices that were made and how they influence the result of the author identification task. Table 1.1 gives an overview of the characteristics of the data used in previous author identification research.

The number of features is most often varied to determine the influence of certain types of features. Corney et al. [6] indicate that the most successful features are function words and character n-grams. Their tests on non-email data showed that function words gave good results independent of topic, while the character n-grams seem to depend on topic. Corney et al. [6] conducted a baseline experiment using the PhD theses of three people to determine the amount of data necessary for author identification. The results revealed that 20 text samples of 100 words are sufficient for successful author identification. They performed the experiments each with a different set of stylometric features. In the other experiments a combination of different types of features was used. In total, they used up to 184 features, from 5 different categories: ‘character-based’, ‘word-based’, ‘document-based’, ‘function word frequency’ and ‘word length frequency distribution’ [6].

For the tests on the e-mail data they used between 122 and 211 features (the previous 184 plus 27 additional features specific to e-mail). The e-mail data consists of 253 messages obtained from 4 authors. Each of these messages contains between 0 and 964 words, with an average length of 92 words. The classification performance increased when an e-mail structural feature set and an HTML tag feature set were added to the stylometric feature set, but only the addition of the email structural feature set or only the HTML tag feature set does not improve classification performance.

De Vel et al. [29] also executed experiments with e-mail messages. They used 156 messages from three native English authors. Each author contributed e-mails on three topics (about 12,000 words for each author for all topics). The classification was performed using 170 stylistic features and 21 features

	Nr of authors	Nr of messages	Message length	Messages p. author
Corney et al. [6]	4	253	0-964 (avg. 92)	-
Diederich et al. [9]	300	2652	>200	1-100
Hirst & Feiguina [15]	2	2232/942/480	200/500/1000	2232/942/480
Hoorn [16]	3	90	-	30
Houvardas [17]	50	5000	2-8 kB	100
Kjell [19]	2	65	237,000 chars	14 & 51
Stamatatos et al. [25]	10	300	avg. 1,112	30
Tsuboi [27]	3	4961	avg. 112	335-1675
de Vel [29]	3	156	190-400	30-63
v.d. Knaap & Grootjen [28]	28	25,000	1 sentence	-
Zheng et al. [30] EN	20	-	84-346 (avg. 169)	30-92 (avg. 48)
Zheng et al. [30] CH	20	532	avg. 807	30-40 (avg. 37)
Zheng et al. [31] EN mail	3	70	-	20-28
Zheng et al. [31] EN news	9	153	-	8-30
Zheng et al. [31] CH news	3	70	-	20-28

Table 1.1: An overview of the number of authors, number of messages and message length used in the previous author identification research. The message length is the number of words, unless otherwise stated. A dash (-) is used when a measure is unknown.

describing the structure of the e-mail. These experiments indicated that the category of style markers contributes most to the author identification, compared to the category of structural features. However, because the experiments are executed with messages from only three authors it can not be concluded that in all author identification tasks the style markers contribute more compared to the structural features.

McCombe [22] executed experiments to determine which features can successfully be used for author identification. She performed tests using word unigrams as classification feature. She showed that the results using this method are promising. But no method she used was successful in classification based on word bigrams. Which seems contradictory because word bigrams capture more information about the sentence structure used by the author. Grammar tags indicate the type of word, for example 'verb' or 'noun'. Such tags provide information about the grammatical structure the author used. McCombe found that tag bigrams as features for classification gave encouraging results for further investigation, for short texts. Hirst and Feiguina [15] used tag bigrams to discriminate between the work of Anne and Charlotte Brönte with three experiments using tag bigrams. Details on this research will be described in the subsection 'Grammar in author identification research'.

Because for this research Hirst and Feiguina [15] only have to distinguish between two authors, Anne and Charlotte Brönte, and an equal number of words are taken from each of these authors, the chance of classifying correctly without using any features is already 50 percent.

The research described so far all used English data sets. Author identification is also performed with messages of other languages. For identifying authors of Greek texts, published in a newspaper, Stamatatos et al. [25] used 300 texts from 10 authors, so there are 30 texts per author. Only three authors had an average text length less than 1000 words. With a combination of lexical measures and style markers 87% of the text was classified correctly. Tsuboi and Matsumoto [27] obtained successful results with a data set consisting of 4961 Japanese messages (about 4000 for training and 1000 for testing), with an average message length of 112 words, written by three authors, from a mailing list of a computer programming language development community. Zheng et al. [31] conducted experiments using two English and one Chinese data sets. The English e-mail data sets contained 70 messages of 3 authors, the English newsgroup data set 153 messages of 9 authors and the Chinese data set 70 messages of 3 authors. The English data sets contain messages from several topics, while all Chinese messages were not categorized by topic. The best results were obtained with the English data sets, probably because 205 style markers were used, while there were only 67 style markers for the Chinese data set. The Dutch language has more similarities with English than with Greek, Japanese and Chinese.

The previous research discussed all have one thing in common: they all use less than 10 authors. Zheng et al. [30] executed an experiment that revealed that performance increases when the number of authors decreases. This holds for several machine learning algorithms and for English as well as Chinese. The results for the English newsgroup data set are shown in Figure 1.1.

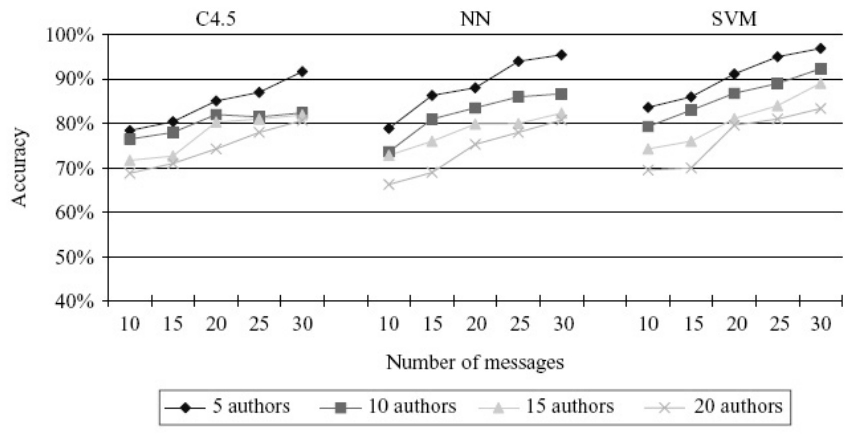


Figure 1.1: The classification performance (accuracy) increases when the number of messages per author increases. The classification performance decreases when the number of possible authors increases. The figure is taken from Zheng et al. [30].

In many real life situations there are usually more possible authors. Therefore I will experiment with messages of more than 20 authors. The research of Houvardas and Stamatatos [17] and the research of Van Der Knaap and Grootjen [28] show that successful results can be obtained when many texts and more than 25 authors are used. Houvardas and Stamatatos [17] used a data set consisting of a training set containing 2500 texts from 50 authors, so there are 50 messages per author. The non-overlapping test set also contains 2500 texts from 50 authors. As features for classification they used the most frequently occurring character n-grams of variable length (3-grams, 4-grams and 5-grams). An accuracy of 73.08% was achieved. The addition of words longer than 5 characters did not significantly improve this result.

The short texts used in previous research of Corney et al. [6], Stamatatos et al. [25] and de Vel [29] contain about 100 to 200 words. However many e-mail and other digital messages are even shorter. The length of the texts used in the experiments for this thesis will be less than 100 words. Van Der Knaap and Grootjen [28] used even shorter messages, these messages are no longer than one sentence. The results of their experiment are encouraging for classification with such short texts. Their data set contained 25.000 lines written by 28 authors, obtained from chat logs. For 2 out of 5 classification tests the text was correctly classified, which means that the the author appeared on the first place. On average the correct author achieved a second place.

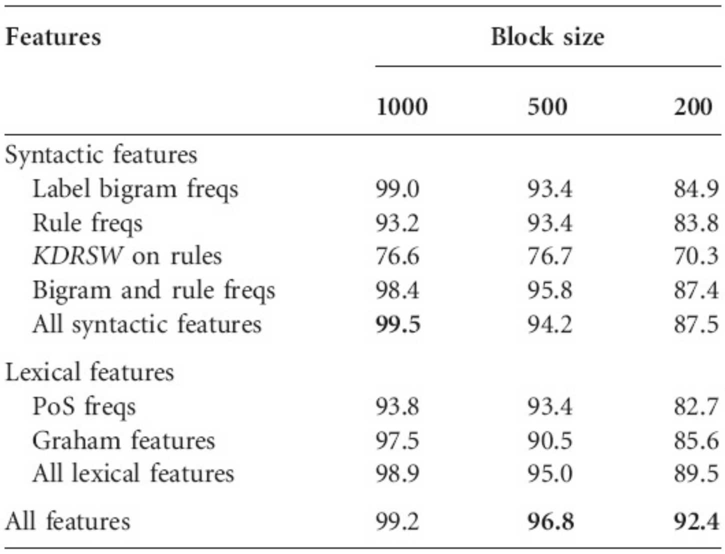
Grammar in author identification research

In some of the previous research features are based on grammatical information. These features are created by part-of-speech tagging, this means that every word gets a grammatical label, e.g. noun, verb, adjective. With these labels new features can be created. A feature that is often used is the 'function word frequency'. Function words have little lexical meaning, they express relationships between words and can not be used in isolation. Examples of English function words are, 'a', 'with', 'in' and 'but'. Function words are an indicator of the syntactic usage of an author. Besides these word based approaches researchers have used other approaches that are more directly linked to the grammatical structure of the text. Diederich et al. [9] performed experiments with German texts that combined function words with tag names. They used the frequency of bigrams of tag names as features. For that purpose they replaced all nouns, verbs and adjectives with their tag names, subsequently they constructed the bigrams and calculated the frequencies of these bigrams. The performance in this condition is better than when using the words themselves. These results are promising since almost all content information is absent. The classification is not biased by content, so apparently grammar information provides signs of the author. Baayen et al. [2] used the frequency of rewrite rules as classification feature. An example of a rewrite rule is NP: DTP + N, where NP stands for ‘noun phrase’, DTP for ‘Determiner phrase’ and N for ‘noun’. Two authors could successfully be distinguished based on the frequency of the rewrite rules. Stamatatos et al. [25] used a Natural Language Processor (NLP), Sentence and Chunk Boundaries Detector (SCBD) for Greek. They used the result of the parser to extract as many features as possible, which resulted in three feature categories. The first category is called the token-level that indicates sentence boundaries. The phrase-level comprises phrase based features like the frequency of noun-phrases. The third category, the analysis-level, is specific to the NLP used. This level contains the stylistic information that could not be captured by the previous two levels. The features of all the three levels reach a performance of 81%, which is above chance level since they discriminate between 10 authors.

Hirst and Feiguina [15] noted that when using types of features as described above, the ordering information that exists in a text is lost. Therefore they parsed the text and used the frequencies of bigrams of the grammatical labels as features. For example after parsing a text the resulting stream of syntactic labels could be:

vp vx vb c c0 nx prp vx be nx prp infp inf to vb ng nx dt
nn of nx nn vnp vnx vb n ax rb jj in pp of nx jj nn cma ng
nx dt nn of nx nn pp in nx nn cma nx dt jj nn pp in nx nn
cma ng nx dt nn of nx nn per

The bigrams are then vp-vx, vx-vb, vb-c, c-c0 etc. The frequencies of these bigrams are the features. Besides these features they also used the frequencies of the rewrite-rules that were used by the parser. Hirst and Feiguina [15] did not restrict themselves to these syntactical features, they also experimented with lexical features like the part-of-speech tag frequencies and features used by Graham [13], which include average word length and hapax legomena. The percentages that were correctly classified for the different feature sets are summarized in Table 1.2 [2], taken from Hirst and Feiguina [15].



Features	**Block size**		
	1000	**500**	**200**
Syntactic features			
Label bigram freqs	99.0	93.4	84.9
Rule freqs	93.2	93.4	83.8
KDRSW on rules	76.6	76.7	70.3
Bigram and rule freqs	98.4	95.8	87.4
All syntactic features	**99.5**	94.2	87.5
Lexical features			
PoS freqs	93.8	93.4	82.7
Graham features	97.5	90.5	85.6
All lexical features	98.9	95.0	89.5
All features	99.2	**96.8**	**92.4**

Figure 1.2: The results of the experiments of Hirst and Feiguina [15], in which they discriminate between texts of Anne and Charlotte Brönte. They experimented with syntactical and lexical features to identify the authors of short texts.

Remember that they only distinguish between two authors, Anne and Charlotte Brönte, and use an equal number of messages from these authors, so the results are above chance level (50%). The performance when using syntactic features is not worse than performance when using lexical features. They concluded that the syntactic label bigram features are the most promising syntactic feature set for short texts. They conducted follow-up experiments with only the label bigram feature set and all lexical

[2]KDRSW represents five measures of vocabulary richness: “Yules measure K and Simpsons measure D of the lexical repetition rate; Honore’s measure R and Sichel’s measure S of hapax legomena and dislegomena, respectively; and Brunets measure W based on the type/token ratio.”[15].

features. The results of these experiments reveal that performance increases if the 75 label bigrams with the highest frequency are used instead of the 150 label bigrams with the highest frequency.

McCombe [22] also concluded that tag bigrams as features for classification gave encouraging results for further investigation, for short texts.

The experiments of McCombe [22] and Hirst and Feiguina [15] capture the ordering information in the text, but do not capture the relationships between words that are not positioned next to each other in the text. As said before, function words indicate relationships between words. Usually this comprises relationships between words that are not positioned next to each other in the text. However these relationships reveal even more about the actual grammatical structure of the text. I will illustrate this with an example. Take the Dutch sentence 'De man zit op het bankje' ('The man is sitting on the couch'). The approach of Hirst and Feiguina will capture the relationships: 'de-man', 'man-zit', 'zit-op', 'op-het' and 'het-bankje'. But it will not capture the relationship 'zit-bankje', which are related by the function word 'op'.

So, to determine the influence of grammatical information on author identification the texts will be parsed using a natural language parser, that produces dependency triplets that describe the grammatical relationship between words, even if these are non-adjacent. The parser that will be used for this thesis is the DUPIRA parser for Dutch developed by Koster [20]. This parser is described next.

The DUPIRA parser

The DUPIRA parser is a natural language parser (NLP) for Dutch and produces a dependency tree. This tree consists of dependency triplets, so unnesting the tree results in these dependency triplets. Dependency triplets have the form [word, relation, word] [20]. The triplet also contains the word category the word belongs to. An example of a triplet is '[V:leest,OBJ,N:boek]' is obtained from the sentence 'De man op het bankje leest een boek' The triplet indicates that 'leest' is of the category 'Verb' (V) and that the word 'boek' is of the category 'Noun' (N). The words 'leest' and 'boek' have an object relation, were 'boek' is an object of 'leest'. Word bigrams do not capture this relation because 'leest' and 'boek' do not occur subsequently in the sentence.

An example produced by the parser is shown in Table 1.2:

Input	De man op het bankje leest een boek.
Dependency graph	[N:man MODop [N:bankje DET het] DET de] SUBJ [V:leest OBJ [N:boek DET een]]
Result of unnesting	[N:bankje,DET,het] [N:boek,DET,een] [N:man,DET,de] [N:man,MODop,N:bankje] [N:man,SUBJ,V:leest] [V:leest,OBJ,N:boek]

Table 1.2: An example of the output of the DUPIRA parser [20]. A dependency graph is constructed from the input line, which after unnesting results in dependency triplets.

The N stands for 'noun', Det for 'determiner', Mod for 'modifier', SUBJ for 'subject' and OBJ for 'object'. The dependency tree of this example is pictured in Figure 1.3.

There are different types of possible dependency triplets because of the different types of relations that exist between words. The relationships that exist depend on the language. I will now give an overview of the relationships of the DUPIRA parser for Dutch. Possible dependency triplets are summarized in Table 1.3.

There are different types of modifier and conjunctive relationships, therefore in the triplets of these types is stated which type it is. This is illustrated with the next examples taken from the DUPIRA website.

In dat geval kun jij de rest opeten: [V:opeten,MODin,N:geval]
Toen gaf ik hem een klap: [V:gaf,MODaan,hem]

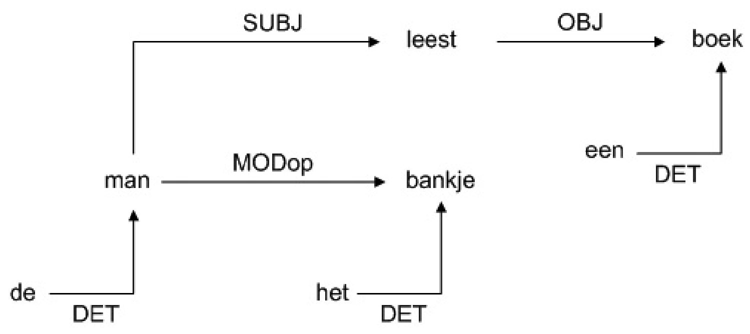


Figure 1.3: An example of a dependency tree.

Relation type	Notation	Example (dutch)
Subject relation	[noun,SUBJ,verb]	[N:man,SUBJ,V:leest]
Subject relation	[personal pronoun,SUBJ,verb]	[jij,SUBJ,V:leest]
Object relation	[verb,OBJ,noun]	[V:leest,OBJ,N:boek]
Object relation	[verb,OBJ,personal pronoun]	[V:ziet,OBJ,jou]
Predicate relation	[verb,PRED,noun]	[V:heet,PRED,N:Piet]
Attribute relation	[noun,ATTR,adjective]	[N:rover,ATTR,dertiende]
Modifier relation	[verb,MOD,adverb]	[V:piepen,MODadv,wel]
Determiner relation	[noun,DET,article]	[N:man,DET,de]
Determiner relation	[noun,DET,possessive pronoun]	[N:boek,DET,zijn]
Auxiliary relation	[verb,AUX,verb]	[V:gewezen,AUX,V:had]
Conjunctive relation	[verb,CON,personal pronoun]	[V:piepen,CONals,jij]

Table 1.3: An overview of dependency triplets which can be produced by the DUPIRA parser.

De man wiens brood ik eet: [N:brood,MODvan,N:man]

The words from an open category, like nouns and verbs, contribute most to the classification of a text. Words from these categories contribute most to the aboutness of the text [1]. Therefore triplets that only contain words from open categories probably contribute most to the classification. The determiner, auxiliary and conjunctive relations do not contribute to the aboutness of the text. In English, the triplet types that contribute most to aboutness are the attribute, preposition, object and subject relations [21].

The dependency triplets can be used to extract features. Possible features are the frequency of the type of triplet, for example Noun-Determinant, or the frequency of that triplet, for example the frequency of [N:boek, DET, een]. Building the features like this results in a high number of features and a sparse feature space. The machine learning algorithm used for classification needs to handle this type of feature space.

1.2 Possible features

There are many possible features, from different categories, that can be used for author identification. The number and types of features is often varied in author identification research to determine the influence of certain types of features. In this section the features used or mentioned in previous research are summarized.

In literary authorship stylometric features are commonly used [4, 6, 7, 22]. Examples of stylometric features are summarized in the list below, taken from Corney et al. [7].

- Letter frequencies
- N-gram frequencies (overlapping n-character frequencies)

- Function word usage (short structure-determining words: common adverbs, auxiliary verbs, conjunctions, determiners, numbers, prepositions and pronouns)
- Vocabulary richness (number of different words used)
- Lexical richness (word frequency as a function of full text)
- Distribution of syllables per word
- Word frequencies
- Hapax legomena (words used once only)
- Hapax dislegomena (words used twice only)
- Word length distribution
- Word collocations (words frequently used together)
- Sentence length
- Preferred word positions
- Prepositional phrase structure
- Distribution parts of speech
- Phrasal composition grammar

Note that some of these features are content biased, like the hapax legomena and hapax dislegomena [29]. There is no consensus on which stylometric features are the best for authorship identification.

Corney et al. [7] only mention the N-gram frequencies feature for characters. But other types of N-grams were used in the previous research. One of these types are the word N-grams, which are constructed from overlapping N words. Word frequencies are the same as the frequencies of word 1-grams, also called word unigrams. A word 2-gram consists of 2 successive words and is often called a word bigram. Hirst and Feiquina [15] used bigrams of syntactic labels. So the N-gram feature can also exist for overlapping n-label frequencies.

De Vel et al. [29] and Zheng et al. [31] speak of style markers instead of stylometric features. They also add the category of structural features, which capture the structure of an e-mail.

- Has a greeting acknowledgment
- Uses a farewell acknowledgment
- Contains signature text
- Number of attachments
- Position of re-quoted text within e-mail body
- HTML tag frequency distribution/total number of HTML tags

The use of the results of the DUPIRA parser for author identification introduces a new type of feature:

- Dependency triplets frequencies

1.3 Machine learning algorithms

Machine learning algorithms learn the characteristics of training data samples. This information is often used to create a model. In essence this is a classification model, each combination of different feature values for the characteristics is labeled with a predefined class. The model is then used to generalize over unseen data. The model uses the characteristics of the unseen data to predict the class label for this unseen data sample. The unseen data sample receives the class label predicted by the model. There are different types of machine learning algorithms that achieve this in a different way [7, 26].

Different machine learning algorithms provide different classification results. For author identification different methods are used, like support vector machines and neural networks. There is no consensus on which is the best classification method to be used for authorship identification, however support vector machines are widely used. An overview of which machine learning algorithms are used in the previous author identification research are provided in Table 1.4

Van der Knaap and Grootjen [28] showed that Formal Concept Analysis (FCA) can be used for author identification in short texts. In this section I will describe the Decision Tree, Nearest Neighbor, Neural Network and Support Vector Machine algorithms that may be suitable for the classification problem of identifying an author, given a list of possible authors. The results are compared to determine which algorithm is most suitable for author identification.

	FCA	Decision Tree	Nearest Neighbor	Neural Network	SVM
Corney et al. [6]					X
Diederich et al. [9]					X
Graham et al. [13]				X	
Hirst & Feiguina [15]					X
Hoorn [16]			X	X	
Houvardas [17]					X
Joachims [18]					X
Kjell [19]				X	
Tsuboi [27]					X
de Vel [29]					X
van der Knaap & Grootjen [28]	X				
Zheng et al. [30, 31]		X		X	X

Table 1.4: An overview of which machine learning algorithms are used in the previously discussed author identification research.

1.3.1 Decision Tree

In decision trees the characteristics of the data are modeled as a tree structure. The root node contains a feature test that separates data samples that have a different value for the feature being tested. Each test should result in subsets of possible categories. The terminal nodes contain the class label. In the case of author identification this is the name or identification number of the author. A very simple example of how a decision tree might look for an author identification task is given in Figure 1.4.

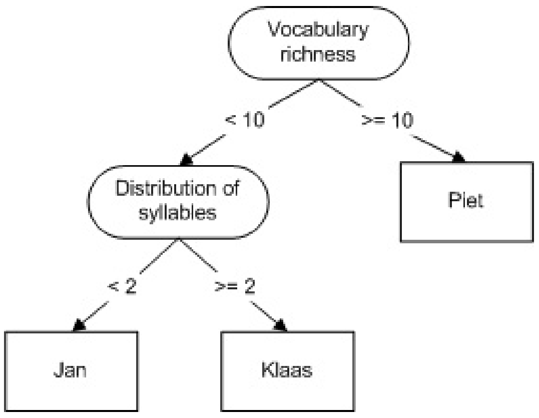


Figure 1.4: An example of a decision tree for an author identification task.

The number of decision trees that can be constructed is exponential in the number of attributes. Therefore an algorithm building decision trees needs to use a strategy that produces a tree within a reasonable amount of time. A commonly used strategy is a greedy approach, which creates the nodes of a decision tree by choosing locally the most optimal test. There are several measures to decide what the most optimal test is. Possible measures are the 'Gini index' and the 'Classification Error'. Studies have shown that the measure used does not have a large effect on the performance since they measure similar information. For each of these measures holds that the test condition is better when the difference in the value of the measure before the split and value after the split is larger. So the decision tree is build based on the information that is captured in the data samples, not by the order in which examples are presented to the algorithm [24, 26].

An advantage of decision trees is that once the tree is constructed, classification of unseen data is very fast. Another advantage is that when two features are highly correlated, when one is chosen as a test, the other one will not be used anymore. A disadvantage of decision trees is that when the data contains irrelevant features these might be used in the decision tree, resulting in a three that is larger than necessary for the classification. This problem can be resolved by eliminating the irrelevant features. However this is difficult when it is unclear which features are relevant.

In the domain of author identification, the authors are the categories and the messages are the data samples provided to the decision tree learning algorithm. A disadvantage of decision trees in the domain of author identification is that most features are numeric. As can be seen in the picture continue values have to be split in categories. How these are divided is crucial for the performance of a decision tree. Finding the best value for splitting is computationally expensive, hence decision trees are not suitable for classification tasks of which many features are continuous. Because the features used for the experiments in this thesis will be continuous, decision trees are not a suitable machine learning technique.

Most of the previously discussed studies do not use decision tree learning. Only Zheng et al. [30] used C4.5, a decision tree algorithm developed by Quinlan [24], with the Weka software [14]. They compared the performance of the C4.5 algorithm with the performances of a neural network and a support vector machine for different feature sets for English and Chinese data. For all feature sets in both languages the performance of the C4.5 algorithm was lowest. As can be seen in Figure 1.5, taken form Zheng et al. [30].

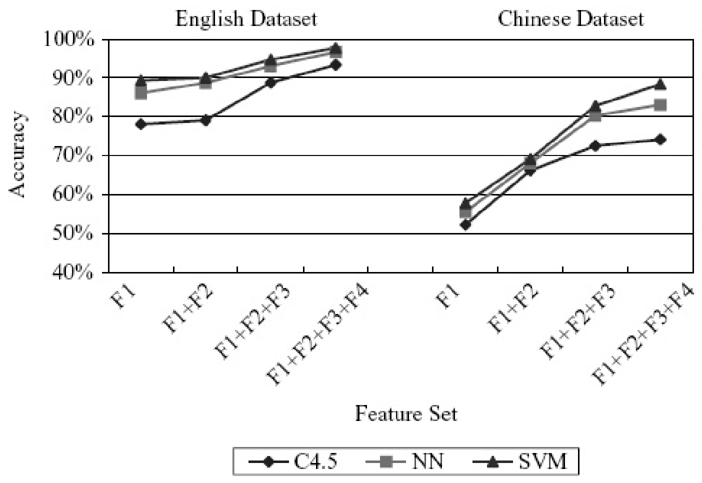


Figure 1.5: Comparison of performance of the C4.5 decision tree, neural network and support vector machine algorithms in the experiments of Zheng et al. [30].

1.3.2 Nearest Neighbor algorithm

A nearest neighbor algorithm does not create a model from the training data. This algorithm is a so called ‘lazy learner’, it retrieves the information from the test data when needed to classify an unseen sample. Each sample from the training data is represented as an n-dimensional data point. The ‘n’ represents the number of features that is used to describe the data. When an unseen sample is presented to the algorithm it will retrieve the k-nearest neighbors of that sample calculated with a proximity measure. The ‘k’ is the number of nearest neighbors that should be retrieved. The unseen data sample gets the same class label as its k neighbors. If these neighbors have more than one class the unseen data receives the label that the majority of its neighbors has. If there is a tie between class labels, a random class label is given to the unseen sample [26].

A disadvantage of the nearest neighbor algorithm is that when there are a lot of features many examples are needed to perform the classification. For the domain of author identification this will be a problem when many messages are used, which results in a lot of features.

None of the previously described studies on author identification use the nearest neighbor algorithm for author identification. Joachims [18] compared the Support Vector machine, which I will describe later, with the nearest neighbor, C4.5, naïve Bayes and Rocchio algorithm for text categorization. In this domain the nearest neighbor algorithm performed better then the C4.5, naïve Bayes and Rocchio algorithms. However the support vector machine outperformed the nearest neighbor algorithm. Hoorn et al. [16] compared a neural network with Bayes classification and a nearest neighbor algorithm in identifying the author of a poem. When using character bigrams the performance of the neural network and nearest neighbor algorithm did not differ significantly. However when character trigrams were used, the neural network performed better than the other two methods.

1.3.3 Neural Networks

Another machine learning technique are neural networks. A neural network is made up of nodes with directed weighted links between them. The network has an input layer representing the input features, an output layer to give the output of the model, and possibly several hidden layers. The weighted sum of the input of a node is used as an input for an activation function, which determines the output of that node. The activation function makes it possible to produce an output that is a nonlinear function of the inputs. During the learning phase the weights of the network are adjusted until an error rate is minimized. A widely used method to minimize this error is the gradient descent. For training the hidden units a commonly used method is back-propagation. A training example is presented to the network and the network produces an output. This output is compared to the actual class the example belongs to. The weights are adjusted so that the output of the network is closer to the actual output. These methods can only be used in supervised learning, the class labels need to be known in order to learn. The gradient descent method is a greedy approach, therefore the network may only find locally optimal solutions. After learning, unseen data can be put into the network which will predict the output class.

To use a neural network a lot of parameters have to be set: the number of input nodes which depends on the number and type of features, the number of output nodes which depends on the number of classes, number of hidden layers, number of nodes in the hidden layers, the activation function, and the initial weights. Improperly setting these parameters may result in under-fitting so the network can not fully describe the data or in over-fitting so the network can not generalize well to unseen data [26, 7].

Kjell [19] used a neural network with character bigrams as features to identify the authors of articles in the Federalist Papers. These articles were written by James Madison or Alexander Hamilton. He found several network configurations that successfully classified the papers. However there were only 14 papers of Madison compared to 51 of Hamilton, so all the papers of Madison had to be in the training set. As mentioned before Hoorn [16] performed author identification experiments in which the neural network outperformed the nearest neighbor and a Bayes algorithm. Zheng et al. [30] compared the C4.5, neural network and a support vector machine. Figure 1.1 shows these results. The neural network performed better than C4.5 algorithm, but the support vector machine outperformed the neural network.

1.3.4 Support Vector Machines

Almost every previously described study in author identification successfully used support vector machines (SVM). This technique is based on finding the maximal margin hyper-plane which separates the data in two sets. Finding this hyper-plane is based on structural risk minimization, a principle that tries to minimize the generalization error while minimizing the training error and avoiding a model that is too complex. The earlier discussed machine learning techniques only minimized the training error, but this does not necessarily mean that the generalization error is minimized. So theoretically this means that SVM can better generalize over unseen data. And in contrast with decision trees and neural networks, SVM do not use a greedy approach, therefore it can find the globally optimal solution.

A SVM tries to find the hyper-plane with the largest margin because this improves the generalization error. A small margin is prone to over-fitting. In the Figure 1.6 illustrates an example of the largest margin. The hyper-plane is positioned so that the margin between the classes is as large as possible. Only the data points that are necessary to determine the largest margin are considered, these are called the support vectors. In Figure 1.6, the support vectors are the points that touch the line were the margin stops (dotted) [7, 26].

Note that there are other possible hyper-planes that could separate this data, but for these hyper-planes the margins are smaller. In cases were the data is not linearly separable a soft margin approach can be used. This approach makes a trade off between the width of the margin and the number of training errors. There are also cases in which classes are separated by a nonlinear boundary. For these

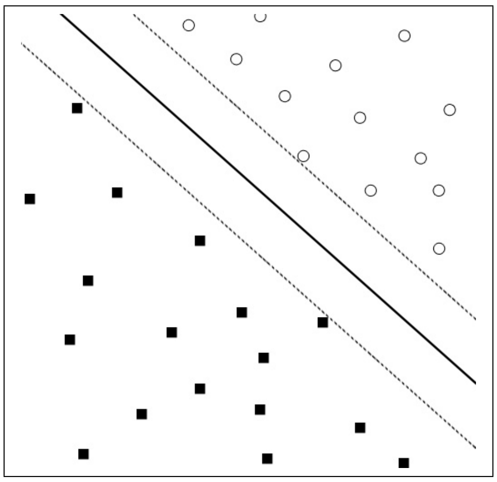


Figure 1.6: A support vector machine algorithm tries to find a hyper-plane with the largest margin. This figure is an example of such a hyper-plane for a 2-dimensional classification task.

cases the Kernel trick can be used. With a Kernel trick data is mapped into a new space in which a linear hyper-plane can be found [15, 26].

The SVM as described only separates two classes. But in many real life situation, like author identification, one wants to distinguish between more classes. This can be performed using pair wise classification. This classification method constructs classifiers for each pair of classes, while ignoring the data that does not belong to one of these two classes. So, for C classes C(C-1)/2 binary classifiers need to be constructed. The unseen data sample gets the class label that is predicted most by the classifiers. As with the nearest neighbor algorithm, there can be a tie between class labels.

Support vector machines are successfully used in previous author identification research, all using different features and texts of varying lengths [6, 7, 9, 15, 17, 27, 29, 30, 31]. Corney et al. [6] used support vector machine with stylometric features known to be successful for discrimination in longer texts. Diederich et al. [9] achieved successful results when classifying texts from a German newspaper, even if nouns, verbs and adjectives are replaced with their tag names. Hirst and Feiguina [15] successfully used SVM in combination with syntactic bigrams, even for texts containing only 200 words, to discriminate between the work of Anne and Charlotte Brönte. Also e-mails can correctly be classified using SVM, as Tsuboi and Matsumoto [27] showed in their experiments. Finally, Zheng et al. [31] concluded that SVM outperformed the decision tree algorithm C4.5 and a neural network.

1.4 Choices

As mentioned at the beginning of this chapter, when defining the experiment choices regarding several aspects have to be made. Recall that these aspects are the language of the messages used, the length of these messages, the number of authors and messages, the types of features and the classification method. By the examination of previous research the influence of these aspects became apparent, which guided me in making choices for the experiments in this thesis. Because the focus of the research question is on grammatical information I will select Dutch messages. Dutch is my native language, hence I will best be able to inspect the grammar of Dutch messages. The focus is also on the possibilities of author identification of short texts, as opposed to previously discussed research which mainly focuses on large texts. Therefore messages shorter than 100 words will be used. In most of the previously described research the number of authors is relatively small, which makes classification easier. Internet applications like message boards, Facebook and Twitter are used by many people, so when identifying the author of texts in real life situations an author identification method should be able to distinguish between many possible authors. Therefore I will use messages from at least 30 authors in the experiments for this thesis.

In most previous research many different types of features are used, which makes it difficult to indicate which features actually are relevant for determining the author of the text. For the experiments in this thesis only specific types of features will be used. The research questions do not focus on identifying the authors of e-mails, so no structural features specific to e-mail will be used. To determine whether classification is possible for short texts based on single words, the relevant feature is word frequency or lexical richness. For the experiment using word bigrams, a suitable feature is ‘word bigram frequency’, this feature can also be calculated as a function of the text. Note that classification based on word bigrams includes the word collocations feature. To capture grammatical relations between two words that are not successive in a sentence I will use the dependency triplets produced by the DUPIRA parser. These triplets will be used as features for classification based on grammatical structures. To answer the third research question, whether the classification results increases when smileys are considered, smileys are added to unigrams and bigrams features. Chapter 2 describes the data and the construction of the features in more detail.

As mentioned in Section 1.3.4, support vector machines are successfully used in previous author identification research, all using different features and texts of varying lengths [6, 7, 9, 15, 17, 27, 29, 30, 31]. These results indicate that SVM is a suitable and promising technique for author identification. Therefore I will use this machine learning technique for the experiments in this thesis.

Chapter 2

Methods

To provide answers for the research questions, several classification tasks will be executed. In this chapter I describe the conditions for the experiment, the data, the feature extraction for each condition and the classification methods that will be used.

2.1 Conditions

There are seven conditions that use different features to identify the author of a short text.

- Word unigrams excluding smileys
- Word unigrams including smileys
- Word bigrams excluding smileys
- Word bigrams including smileys
- Dependency triplets
- Word unigrams and dependency triplets
- Word bigrams and dependency triplets

The results of these conditions will be compared to determine the influence of adding grammatical structure and smileys as features for author identification. The influence of smileys can be determined by comparing the results of the word unigrams and bigrams data excluding smileys with the word unigrams and bigrams data including smileys. In the dependency triplets condition smileys can not be included, because the parser cannot interpret these smileys since they are not part of the grammatical structure of a sentence. To determine the influence of adding grammatical structure more comparisons will be made. First, the results of the word unigrams data will be compared with the results obtained with the word bigram data. Secondly, the results of dependency triplets will be compared to the unigrams data and to the results of the bigrams data. Finally the influence of adding the triplet features to the word unigrams and word bigrams will be assessed.

2.2 The data

The data is extracted from the message board fok.nl. It is the largest Dutch message board with about 300,000 members who together posted about 80,000,000 messages [12]. The message board consists of several sub boards each representing a different main topic. Because of these characteristics people with different backgrounds are actively posting messages on this board. Only messages from the section 'Eurovisie songfestival 2010' are used in this thesis. In this section people talk about the performers that participate in the Eurovision song contest. There are discussions about the songs, performances and outfits of the performers. The messages are all about the same main topic and it is likely that people with different backgrounds participate in this sub board.

Figure 2.1 shows the top 50 of authors who posted in this subsection and the number of messages they posted. Even among only 50 authors there is a big difference in the available messages of each author. Using a different amount of messages for each author has an influence on the classification. If all messages of author 'w163650' are included, many messages of this author will be classified correctly only

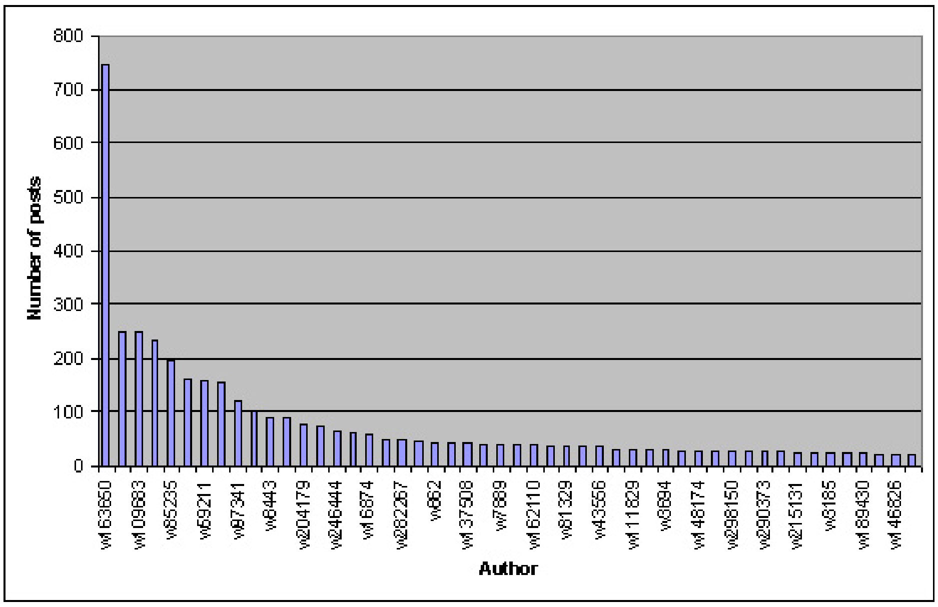


Figure 2.1: The number of messages of the top 50 authors who posted in the 'Eurovisie songfestival 2010' section of the fok.nl message board.

because the chance that a message is written by this author is high. Therefore I will select 25 messages per author, for 40 authors.

To verify whether such a distribution is specific to this section I looked at the message distributions of other sections as well. All sections show an unequal distribution of messages among the authors. Figure 2.2 contains the distribution of four other sections.

A Java program build by Jan Broos and Franc Grootjen extracts data from the fok.nl forum, including a writer ID and the messages. Quotes, links and signatures are removed from the messages. The program saves all data in an XML file. I let the program generate two XML files for the Eurovision Songfestival subsection. In one XML file the smileys are included in the messages, in the other XML file the smileys are removed. The former is used to create the unigrams and bigrams classification files excluding smileys, the latter to create classification files for the other conditions.

2.3 Extracting features

The data from the xml file is used to create features and instances suitable for classification. To generate a classification file I build a Java program. This program extracts the features for each condition. Because each condition requires different features, the feature extraction process differs for each condition. In the following sections I will describe these processes of feature extraction.

2.3.1 Unigrams data

In the unigrams condition the words are the features, so the words from the text have to be isolated. A list of all single words occurring in all texts is constructed. This is the list of features. For each text the occurrence of a feature is counted. This frequency is then normalized for text length. Normalization is necessary because the frequency of a word in a text might be higher compared to other text, not because it is an indication of the author of that text, but just because the text is longer. If a text consists of more words, then the chance on occurrence of a word in the text increases.

When constructing the features punctuation marks, like full stops and comma's are removed. Only two punctuation marks are not removed: the apostrophe (') and the dash (-). Since these are used within words, e.g. d'r and Sha-la-lie (Song title).

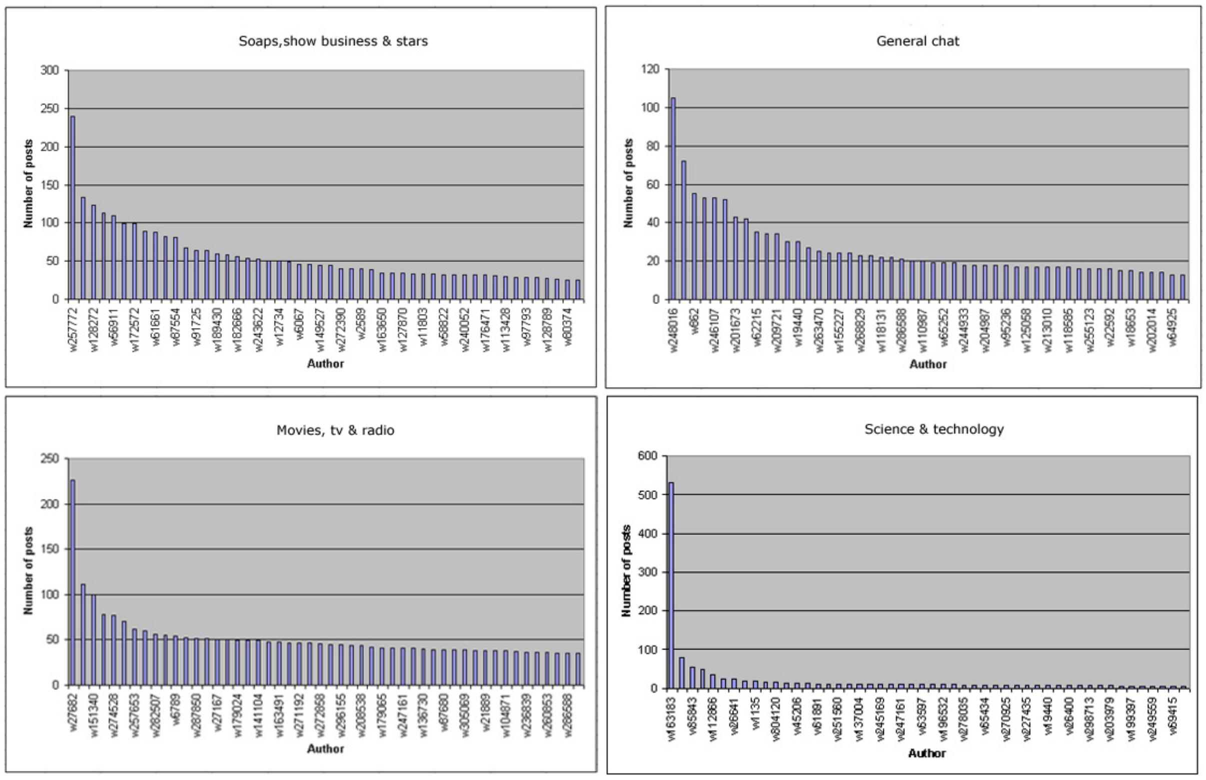


Figure 2.2: The number of messages of the top 50 authors from different subsection of the fok.nl message board.

There are two options for constructing unigrams data: include smileys as features or not. In digital messages people often use smileys, sometimes a message only consists of a smiley. So, smileys can give information about who wrote the text. Smileys are added to the unigrams data the same way as words, but with the extension '.gif', to differentiate between smileys and words. For example: 'laugh.gif' is added as a feature, not just 'laugh' because the latter can also occur as a word. Examples of smileys of the fok.nl message board are: 'wapper.gif', 'sorry_roos.gif' and 'very-grumpy.gif'. These are added as features.

I will now give examples of the unigrams that are extracted from the message De man op het bankje leest een boek. smile.gif.
The unigrams excluding smileys are: 'De', 'man', 'op', 'het', 'bankje', 'leest', 'een', 'boek'.
The unigrams including smileys are: 'De', 'man', 'op', 'het', 'bankje', 'leest', 'een', 'boek', 'smile.gif'.

2.3.2 Bigrams data

For this condition word bigrams have to be constructed. Word bigrams consist of two successive words. For punctuation marks the same rules apply as for constructing the unigrams: exclude all marks except the apostrophe and dash. For each text the frequency of occurrence of a bigram in the text is counted and normalized for text length. Also in this condition smileys can be included.

The bigrams extracted from the example message De man op het bankje leest een boek. smile.gif are as follows:
The bigrams excluding smileys are: 'De man', 'man op', 'op het', 'het bankje', 'bankje leest', 'leest een', 'een boek'.
The bigrams including smileys are: 'De man', 'man op', 'op het', 'het bankje', 'bankje leest', 'leest een', 'een boek', 'boek smile.gif'.

2.3.3 Dependency triplets data

For the dependency triplets data, the features are triplets produced by the DUPIRA parser. So the texts first have to be parsed by the DUPIRA parser. Each text is sent to the parser sentence by sentence. These calls to the DUPIRA parser are automated with the Java program. For each text the occurrence of each triplet in the text is determined. The frequency of occurrence is normalized for text length, just as in the unigram and bigram conditions.

The triplet data do not contain smileys because the parser cannot interpret them. Smileys are not part of the grammatical structure of a sentence.

For testing whether different conditions complement each other I created two additional conditions: word unigrams combined with the dependency triplets and word bigrams combined with dependency triplets. I build the Java program in a way that it could easily combine such combination. The word unigrams and bigrams in these conditions do not include smileys.

The triplets extracted from the example message De man op het bankje leest een boek. smile.gif are given in Table 1.2.

2.3.4 Number of features

Each previously described condition results in a different number of features. The number of features created for the Euro-vision Song festival data are summarized in Table 2.1.

Condition	Nr of features
Unigrams excl smileys	4008
Unigrams incl smileys	4053
Bigrams excl smileys	12692
Bigrams incl smileys	13454
Triplets	5362
Unigrams + triplets	9370
Bigrams + triplets	18054

Table 2.1: The number of features created for the Euro-vision Song festival data for each condition.

2.4 Classification

The Java program I build to extract the features also constructs the classification file. This is a data file in .arff format, the format preferred by Weka. See appendix A for more information on the .arff format and an example of a (small) .arff file.

The classification is performed by a support vector machine algorithm (SVM). There are different implementations of this algorithm. The standard SVM algorithm implemented in Weka is SMO. Weka also provides wrapper classes for LibSVM and LibLINEAR, so these can be used in the Weka environment as well. I will shortly describe the differences in the following section [3, 11, 10, 14].

2.4.1 Support Vector Machine algorithms

There are different implementations of support vector machines, the three I consider are:

- SMO
- LibSVM
- LibLINEAR

SMO stands for Sequential Minimal Optimization. A support vector machine has to solve a quadratic programming problem. SMO breaks the problem into several smaller quadratic programming problems [23]. LibSVM is a faster implementation of the SMO algorithm. LibSVM reduces the computational time by applying two techniques: caching and shrinking. Caching means that earlier computed values are stored in memory so that re-computation is unnecessary. The shrinking technique temporarily eliminates variables that have reached a predefined lower or upper bound. So these values are not used in subsequent computations [3, 10].

SMO and LibSVM solve the multi-class problem by combining several binary classifiers, using the so called one-vs-one strategy. This strategy constructs a classifier for each pair of classes.

When training a support vector machine several parameters can be set. One of these is the type of Kernel to use. A linear kernel is preferred when the number of features is large. Mapping the data to a higher dimension does not improve performance. LibLINEAR is an implementation of a support vector machine which solves classification problem linearly without the use of kernels. This can significantly reduce the training time while preserving similar or better performance. LibLINEAR is faster on large sparse data sets [11]. The LibLINEAR implementation has two options to handle the multi-class problem. The first option is the one-vs-rest strategy, the other is a method developed by Crammer and Singer [8]. This method solves the multi-class problem directly so there is no combination of binary classifiers necessary [11].

In the remainder of this thesis I will use abbreviations to indicate the different algorithms and options. An overview of the abbreviations is given in Table 2.2.[3]

Abbreviation	Algorithm	Options
SMO	Weka's SVM	normalization
LibSVM	LibSVM	
LibSVM Z	LibSVM	normalization
LibLIN	LibLINEAR	
LibLIN Z	LibLINEAR	normalization
LibLIN S4	LibLINEAR	Crammer and Singer method for handling multi-class problems
LibLIN S4 Z	LibLINEAR	Crammer and Singer method for handling multi-class problems, normalization

Table 2.2: An overview of the abbreviations used in the remainder of this thesis to indicate the different support vector machine algorithms and options that are used.

2.4.2 Cross-validation

For the classification I will use 10-fold cross-validation. This means that the data is randomly partitioned into 10 equal-sized parts. The classification task is performed 10 times, each time a different part is used as test data while the remaining 9 parts are used as training data. So each part is used as test data exactly once. The result of these 10 classification tasks is are averaged. this method reduces the variability of the classification.

2.4.3 Classification results

The Weka output includes several measures that indicate the performance of the classification. These measures can also be used to compare the classifiers and the results of the different conditions. The most intuïtive measure is the percentage correctly classified instances. I will mainly use this measure to determine the best classifier for the classification tasks discussed in this thesis.

The Weka output also includes a confusion matrix which shows how many instances of a category are classified by the support vector machine as the same or a different category.

These outputs are give for the training and the test data. See appendix B for an example of the complete Weka output. Appendix C contains a short description of all performance measures Weka produces.

[3]The abbreviation 'Z' is used to indicate normalization, 'S4' indicates the Crammer and Singer method for handling multi-class problems. This abbreviations are uses because they are the commands for using these options in the Linux Command line.

Chapter 3

Results

This chapter shows the results of author identification in the conditions described earlier: single words, word bigrams and dependency triplets produced by the DUPIRA parser. The first section, classification results, describes all the results of the classification task obtained with the different support vector machine algorithms. First an overview of the performance of all SVM's in all conditions is given. Then the results for each condition are given in the subsequent subsections. In the second section, Comparison, I first compare the different algorithms using their performance as well as speed results. These results are used to determine which algorithm is most suitable for the classification task: author identification in short text. In the subsequent subsections I will compare the performance of this classifier in the different condition. This will give insight into which type of information is most useful for identifying the author of a short text. In the final section of this chapter I will also provide results of a short examination of the probability distribution obtained from Weka's SMO algorithm.

3.1 Classification results

There are several classification tasks performed, which can be divided in several categories: Word unigrams, Word bigrams and Grammatical structure. The word unigrams and bigrams conditions are performed including and excluding smileys. For the grammatical structure condition there are three options: only use the dependency triplets as features, use triplets and unigrams or use bigrams and triplets as the features for author identification.

3.1.1 Overall results

Table 3.1 contains the performance results, the percentage of correctly classified instances, of all support vector machine algorithms for each condition. Note that because from 40 authors an equal amount of messages were used the chance of predicting the correct author without knowledge is 2.5%.

	SMO	LibSVM	LibSVM Z	LibLIN	LibLIN Z	LibLIN S4	LibLIN S4 Z
Unigrams excl smileys	9.33	11.43	11.63	10.83	14.74	12.44	14.24
Unigrams incl smileys	13.24	15.55	13.64	14.74	15.85	16.05	15.65
Bigrams excl smileys	3.11	5.82	3.41	8.33	10.43	8.12	9.63
Bigrams incl smileys	3.11	6.62	3.51	9.63	11.03	9.53	11.13
Triplets	4.21	4.61	3.71	6.42	7.02	6.12	7.42
Unigrams + triplets	5.52	10.92	5.72	10.13	13.94	11.84	13.34
Bigrams + triplets	3.21	6.22	3.51	8.73	10.33	9.33	10.13
Average	5.96	8.74	6.45	9.83	11.91	10.49	11.65

Table 3.1: The percentages of correctly classified instances for the different support vector machine algorithms for all conditions.

Figure 3.1 is a visual reproduction of Table 3.1.

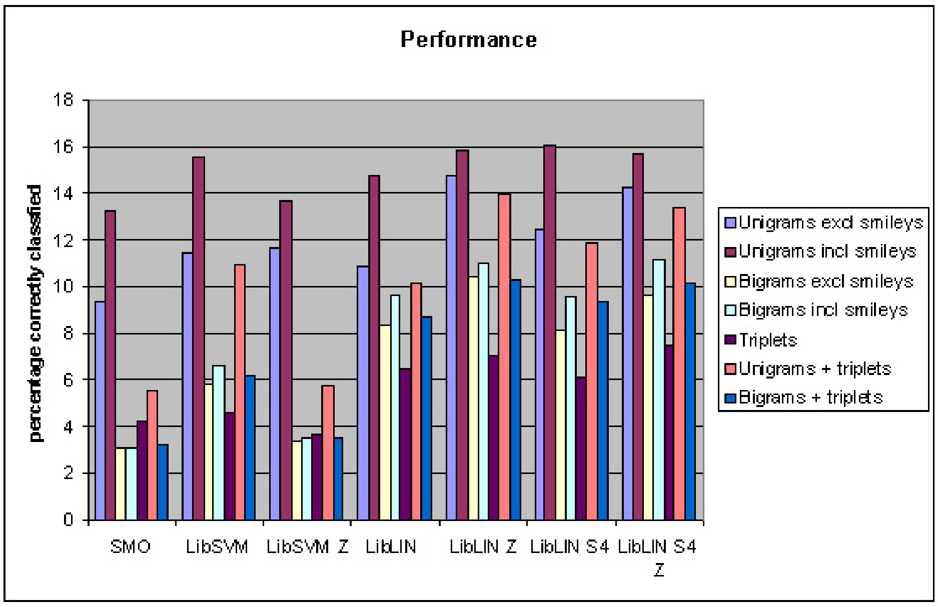


Figure 3.1: The percentages of correctly classified instances for the different support vector machine algorithms for all conditions. This actually is a visual reproduction of Table 3.1.

For all support vector machine types, the unigrams including smileys has the best classification performance. This is also visible in Figure 3.2. This figure shows the percentages correctly classified instanced for the LibLINEAR algorithm with normalization.

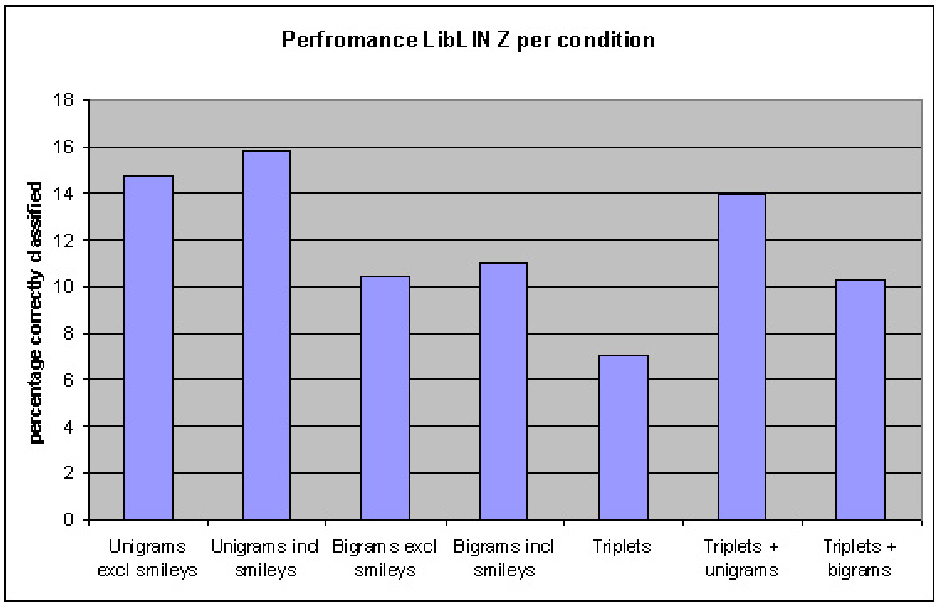


Figure 3.2: The performance of the LibLINEAR algorithm with normalization (Z) for each condition.

3.1.2 Word unigrams

To determine the influence of smileys on author identification, classification is performed without smileys and including smileys combined with word unigrams as features for classification. The results of these two conditions are summarized below.

Unigrams excluding smileys

When using word unigrams as features, excluding smileys, the percentage of correctly classified instances is highest for the LibLINEAR algorithm using the one-vs-rest method and normalization. For 14.74% of the messages the classification algorithm predicted the author correctly.

Unigrams including smileys

The best performance result for the word unigrams including smileys as features is achieved by LibLINEAR using the Crammer and Singer method for handling multi-class problems without normalization.. The percentage of correctly classified instances in this case is: 16.05%. The second best performance results were attained by the LibLINEAR algorithm using the one-vs rest method for multi-class problems and normalization. The LibLINEAR achieved a percentage of 15.85%, which is only 0.2% less than when the Crammer and Singer method is used.

3.1.3 Word bigrams

Also in the condition of word bigrams smileys might influence the classification performance, so again classification is performed on word bigrams excluding and word bigrams including smileys.

Bigrams excluding smileys

The highest performance result using word bigrams as features for classification is achieved by the LibLIN Z method. This method classifies 10.4313% of the instances correctly.

Bigrams including smileys

The best classification performance was achieved by the LibLIN algorithm using the Crammer and Singer method for handling multi-class problems and normalization. The percentage of correctly classified instances was 11.13%. The LibLIN algorithm using the one-vs-rest method for handling multi-class problems and normalization classified 11.03% of the instances correctly, which is only 0.10% less than using the Crammer and Singer method.

3.1.4 Dependency triplets

This section includes the results of using grammatical structure as clues for identifying the author of a short text. First the results of only using dependency triplets is given, then also the results of the combination of word unigrams & triplets and word bigrams & triplets is presented.

Only triplets

The highest percentage of correctly classified instances reached for the triplets data is 7.42%. This results was achieved by the LibLINEAR algorithm with the Crammer and Singer method for multi-class problems and normalization. The second highest result, 7.02% was achieved by the LibLINEAR algorithm using the one-vs rest method and normalization.

Unigrams and triplets

The percentage of correctly classified instances using word unigrams and dependency triplets as features and LibLIN Z as classifier is 13.04%.

Bigrams and triplets

Combining word bigrams and dependency triplets results in correct classification for 10.33% of all instances. This percentage was achieved using the LibLIN Z classification method.

3.2 Comparison

Using the results from the previous section I will first describe which type of support vector machine (classifier) is most suitable for author identification in short texts. Then I will compare the different conditions to determine the influence of using smileys and using more grammatical structure as features for classification.

3.2.1 Comparing classifiers

Compare performance

The classification result of all support vector machine algorithms was highest for the unigrams data including smileys. Figure 3.3 shows the these results.

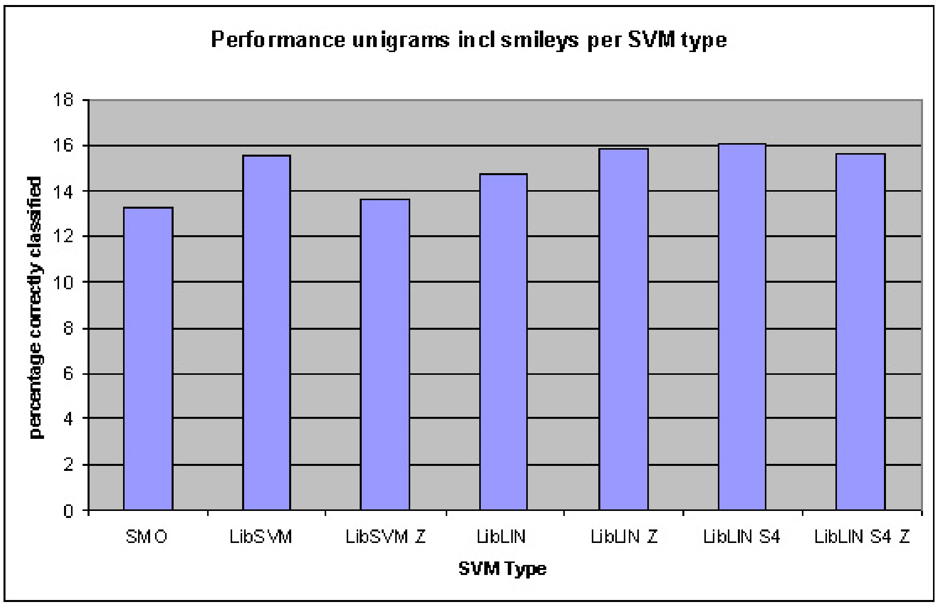


Figure 3.3: The performance of each type of support vector machine algorithm for the unigrams data including smileys.

As can be seen in Figure 3.3, the LibLINEAR with the Crammer and Singer method for handling multi class problems performs best, followed by LibLINEAR with normalization. For the other data sets the LibLINEAR with normalization performs best. Figure 3.2 shows the performance of this algorithm for all conditions.

Compare time to train and test

I mentioned before that LibSVM and LibLIN handle large problems more efficiently than the SMO algorithm. In this section I compare the time[4] needed to train and test for each algorithm. First of all, as can be seen in the correlation graph in Figure 3.6, the time to train and test for one fold increases

[4]The times were measured from the following machine: AMD Athlon (tm) 64 x2 Dual Core Processor 3800+, with 1 GB memory and the operating system 'GNU/LINUX'

when the number of features grows. Figure 3.6 is based on the train and test times for the LibLINEAR algorithm with normalization. The train and test time for the unigrams data including smileys is the lowest.

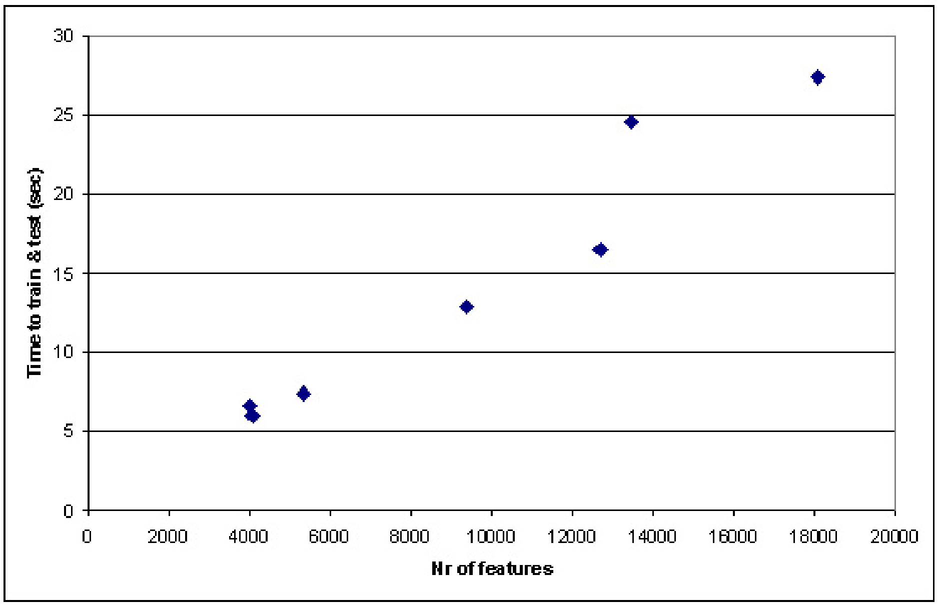


Figure 3.4: The correlation between the number of features and the time to train plus the time to test needed by the LibLINEAR algorithm with normalization for one fold.

Figure 3.7 shows for each algorithm the average time to train and test per fold in seconds. Note that scale of the y-axis is logarithmic.

Clearly the SMO algorithm is very inefficient. For the smallest data set, the unigrams excluding smileys, it needed about 30 minutes to perform the 10-fold cross-validation. SMO took almost 4 hours to classify the bigrams+triplets data using 10-fold cross-validation, which is the largest data set. LibSVM and LibLIN only needed about 10 minutes for the largest datasets.

3.2.2 Word unigrams vs. word bigrams

There are three types of conditions in which word unigrams are used and also three similar types of conditions in which word bigrams are used as features. Both the unigrams and bigrams are used as the only type of features in a condition, I defined these conditions as ‘excluding smileys’. In other conditions smileys are added to either the unigrams or bigrams, the so-called ‘including smileys’ conditions. Finally the unigrams and the bigrams are added to the dependency triplet data, which are the ‘with triplets’ conditions. A graphical overview of the results is given in Figure 3.8.

In every type of condition the word unigrams show the best classification results compared to the word bigrams. The difference is largest when smileys are included. The differences in each type of condition are:

Excluding smileys	4.31
Including smileys	4.81
With triplets	3.61

Also the values for precision, the F-measure and ROC area are smaller for the bigrams data compared to the unigrams data. Actually the values for all measures are, although some very little, better for the unigrams data compared to the values for the bigrams data.

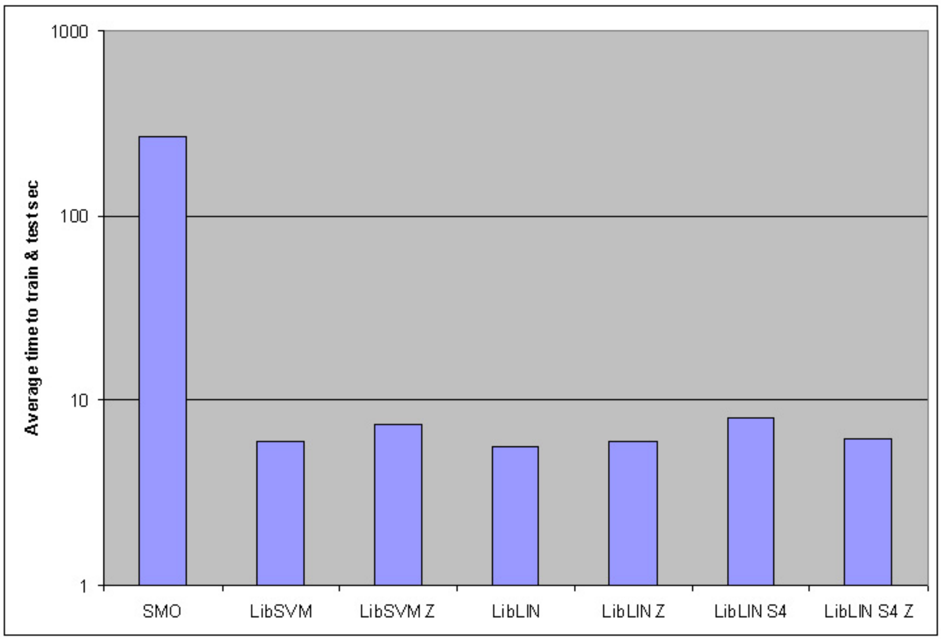


Figure 3.5: The time to train and test needed by the LibLINEAR algorithm with normalization per fold in seconds for the unigrams data including smileys. Note that the y-axis is logarithmic.

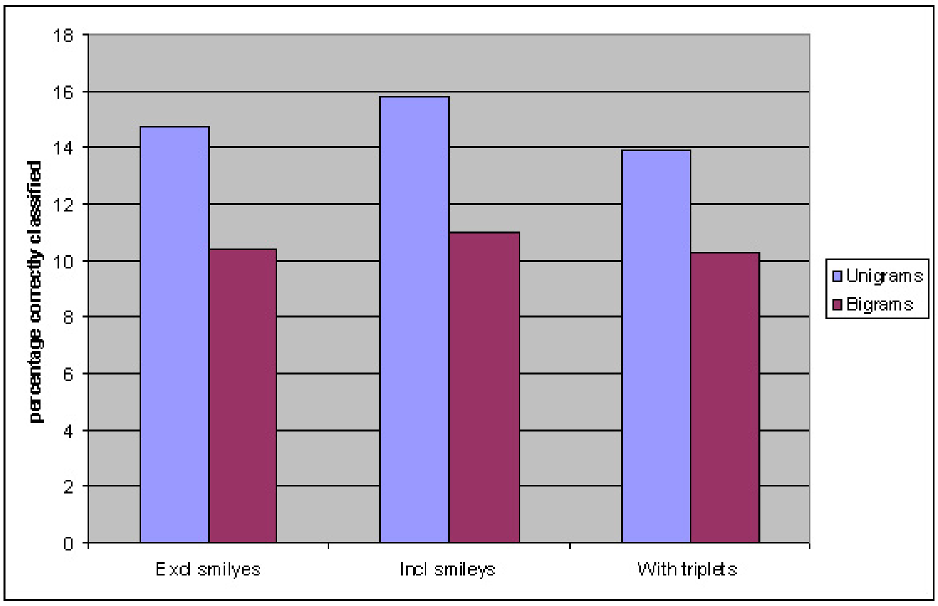


Figure 3.6: Performance results for all conditions that contain unigrams or bigrams, to compare the influence of unigrams and bigrams.

3.2.3 Including vs. excluding smileys

Figure 3.9 shows the performance for the LibLIN classifier with the one-vs rest strategy and normalization (LibLIN Z) for the unigrams and bigrams conditions including and excluding smileys.

For unigrams and bigrams the performance improves when smileys are included. The improvement for the unigrams data is 1.10%, and 0.60% for the bigrams data. For all types of support vector machine

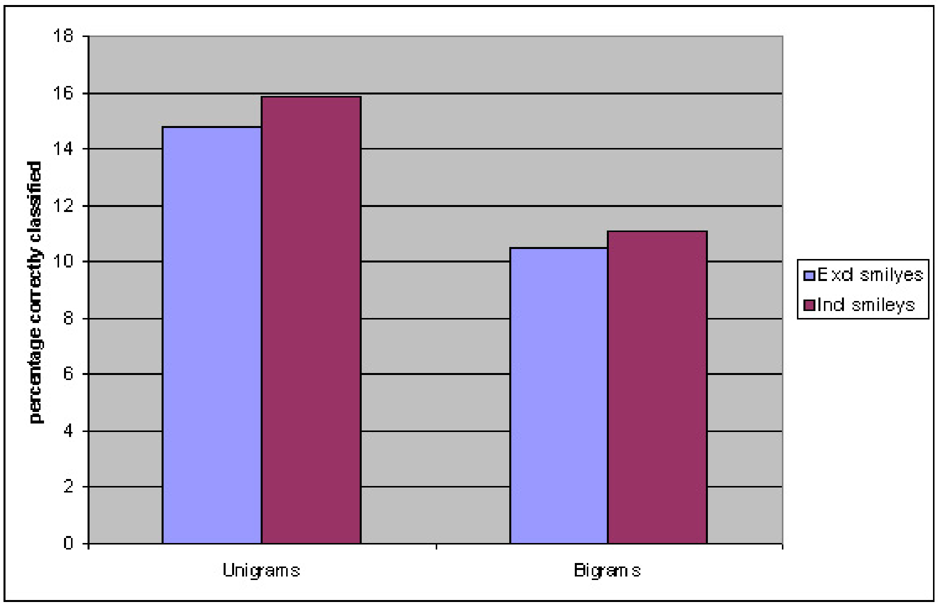


Figure 3.7: Performance for the LibLIN algorithm with normalization, for the unigrams and bigrams conditions including and excluding smileys.

types used in this thesis the performance improves when smileys are included. Only when the bigrams data is classied using the SMO algorithm the performances excluding and including smileys are equal. But how much the performance increases with the addition of smileys depends on the support vector machine type that is used.

The increase in performance due to the addition of smileys to the word unigrams is the least for the LibLIN Z algorithm, and the most for the LibSVM condition without normalization. In that case the increase in performance is 4.11%. For the bigrams the increase in performance is biggest for LibLINEAR with the Crammer and Singer method for handling multi-class problems without normalization (LibLIN S4). The increase is 1.40%.

Also precision, the F-measure and ROC Area increase when smileys are included in the generation of features. All measures improve by the addition of smileys.

3.2.4 Word unigrams vs. dependency triplets

In this section the performance of the word unigrams data is compared to the performance of the dependency triplet data. Remember that the dependency triplets smileys are not included because the parser cannot interpret smileys since they do not belong to the grammatical structure of a sentence. Therefore the performance of the triplets is only compared to the unigrams data excluding smileys. I also combined the word unigram features with the dependency triplet features to perform the classification. The results of this data are also included in Figure 3.10.

The performance when using dependency triplets as features is much lower compared to the performance when using word unigrams. The performance of the triplets data is 7.72% lower. When the triplet features are added to the unigrams data the performance decreases with 0.80%.

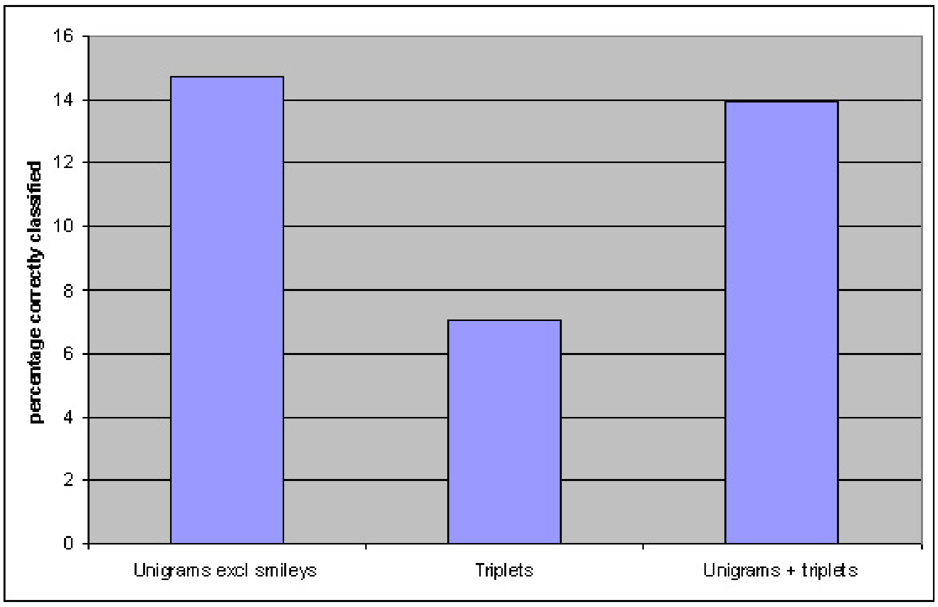


Figure 3.8: Comparison between the performances of the unigrams, triplets and unigrams+triplets conditions.

3.2.5 Word bigrams vs. dependency triplets

In this section I compare the performance when word bigrams are used with the classification result when dependency triplets are used as features. Just like with the comparison between unigrams and triplets data, the bigrams data that will be compared to the triplets data does not include smileys.

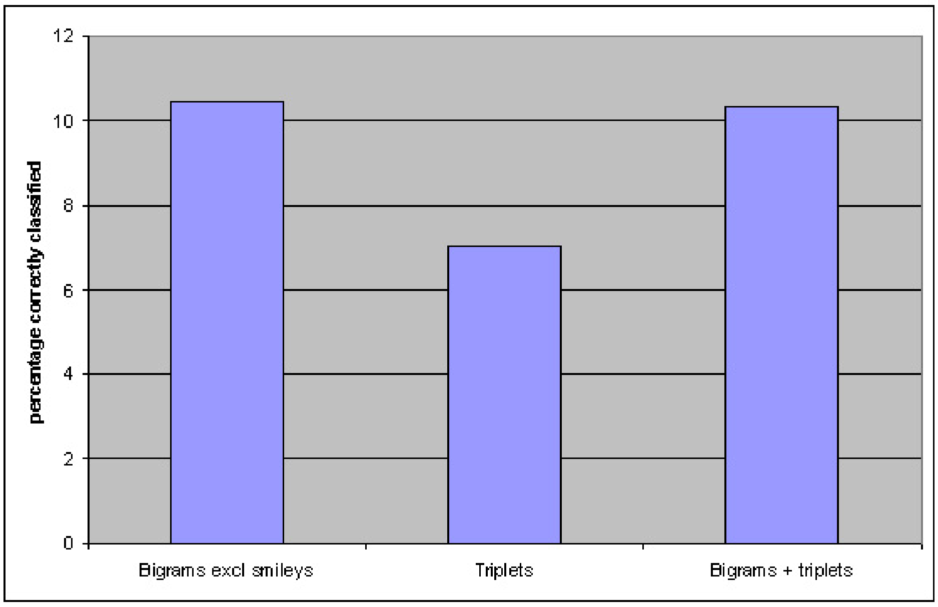


Figure 3.9: Comparison between the performances of the bigrams, triplets and bigrams+triplets conditions.

The pattern of the results is the same as for the unigrams data. The performance for the triplets data is lower than for the bigrams data. And when the triplet features are added to the bigram performance decreases. But because the performance for the bigrams data is lower than the performance of the unigrams data, the difference in performance between the bigrams and triplet data is 3.41%. This is smaller than the difference between the unigrams and triplet data. And the decrease in performance when adding the triplet features is 0.10%.

3.3 Performance of DUPIRA

Now the data from the fok.nl forum is available I tested the performance of the DUPIRA parser on this data. To do this I reviewed the results of 100 sentences. The first remark that has to be made is that messages containing only one word cannot produce a dependency triplet. From the 100 sentences 31 were not parsed, of which 11 were single words. 5 of the 31 sentences are in English. Since the DUPIRA parser is a parser for Dutch it can not interpret the English words.

For the other 69 sentences the DUPIRA parser produced some results. For only 15 of these sentences all words occur in the triplets and the triplets are correct. So I also reviewed the grammatical correctness of the resulting triplets. From the 69 sentences the parser extracted 228 dependency triplets. From these 228 triplets, 24 are incorrect, which is 10.5% of the produced triplets. The results of this review are graphically illustrated in Figure 3.11.

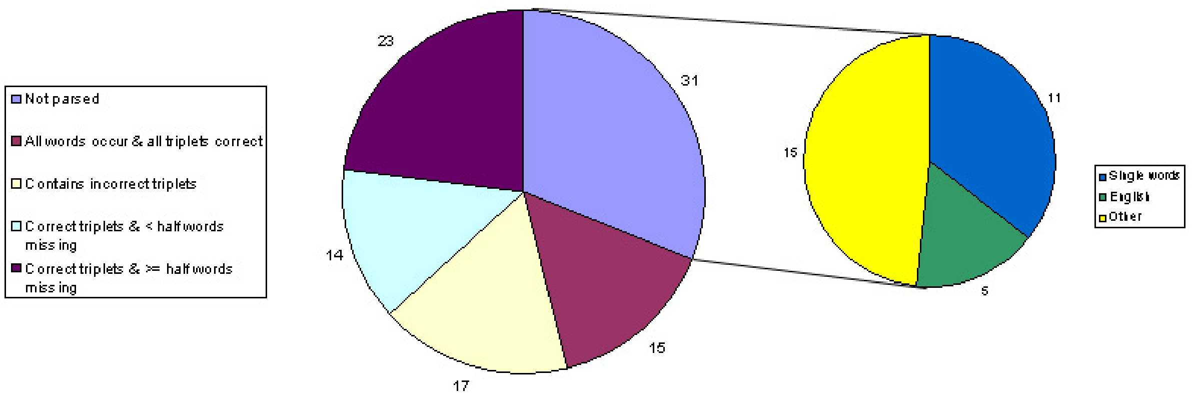


Figure 3.10: The performance of DUPIRA on 100 sentences taken form the Eurovision songfestival section of the fok.nl message board.

These low performance results are probably due to the fact that most people writing on message boards do not make grammatical well defined sentences. I will give two examples of sentences and their triplets produced by the DUPIRA parser.

Example of a correctly parsed sentence:
En dan hoop ik op uitgebreide voorrondes
1. [N:voorronde, ATTR, uitgebreid]
2. [V:hopen, MOD, dan]
3. [V:hopen, PREPop, N:voorronde]
4. [ik, SUBJ, V:hopen]

Example of a sentence which is not completely parsed and includes incorrect triplets:
Oeps, per ongeluk een was op 80 graden ipv 40 gedraaid
1. [N:Q:80, ATTR, op]
2. [N:Q:80, SUBJ, V:zijn]
3. [N:ongeluk, PRED, N:A:een]
4. [V:zijn, OBJ, N:graden]
5. [V:zijn, PREPper, N:ongeluk]
6. [N:A:V:gedraaid, QUANT, 40]

Only the first triplet is correct. This sentence is probably parsed incorrectly because a verb is missing. In Dutch the word ‘was’ can be a verb and a noun. So the parser interpreted this as a verb instead of a noun. The latter example shows the type of sentence the parser has difficulties with. People have no problems understanding this sentence, despite the incorrect use of grammar.

3.4 Class probability distribution

The class label assigned to an unseen sample is the label that was predicted with the highest probability by the classifiers. So other class labels received votes as well. When a sample is classified incorrectly, the correct label might be in the top 3 when looking at the probability scores. For the SMO algorithm, a probability distribution for all classes for all instances can be outputted by Weka.

The output for one instance can look like:

```
1 36:aw18272 30:aw16245 +
0, 0.001, 0, 0, 0, 0, 0.025, 0, 0, 0, 0, 0, 0, 0, 0.009, 0, 0, 0.001, 0.001, 0.001, 0.001,
0.057, 0, 0, 0, 0, 0.002, 0, 0, *0.383, 0, 0, 0, 0, 0, 0, 0, 0.002, 0.287, 0.228
```

The first number (1 in this example) is the number of the test sample. When using cross validation this is the number of the sample in that particular fold. The second number is the number of the actual class label of that sample, followed by the class name. So this sample has class label 36, which is author 18272. The class label predicted by the classifier was 30 which is author 16245. The plus indicates that the predicted class label is incorrect. When the prediction is correct the plus is omitted. The numbers that follow are the probabilities for all classes. In this example there are 40 classes, so there are 40 probabilities. The probability of the class predicted by the classifier is preceded by an asterisk. In this example it is the 30th probability, 0.383.

I wrote a small Java program that, given the probability distributions produced by Weka, determines the top 3 of classes that received the highest probability scores. Then this program counts in how many of the incorrectly classified instances the correct class label does appear in the top 3.

The results for the unigrams data including smileys are:
Of 54 instances the correct class label was the second in the distribution.
Of 50 instances the correct class label was the third in the distribution. So for 104 incorrectly classified instances the correct label appears in the top 3, which is about 12% of the incorrectly classified instances.

The results for the triplets data are:
Of 50 instances the correct class label was the second in the distribution.
Of 47 instances the correct class label was the third in the distribution. So for 97 incorrectly classified instances the correct label appears in the top 3, which is about 10% of the incorrectly classified instances.

Table 3.2 contains the percentages of instances that occur in the top 1, top 3 or top 5 for the unigrams data including smileys and the triplets data, when classified using the SMO implementation.

	Uni+smileys	Triplets
Top 1	13 %	4 %
Top 3	25 %	14 %
Top 5	33 %	20 %

Table 3.2: The percentages of instances from the unigrams including smileys condition and the triplet condition that occur in the top 1, top 3 or top 5 for the SMO implementation.

Chapter 4

Conclusions

The aim of this project was to determine which types of information make it possible to identify the author of a short digital text. In particular, the aim was to determine whether it is possible to identify the author of a short text based on the words and grammar used. I examined this with messages extracted from the 'Eurovision Songfestival 2010' subsection of the fok.nl message board. From these messages three main conditions were constructed: word unigrams, word bigrams and dependency triplets. These triplets were produced by the DUPIRA parser. Because the use of smileys is common in digital messages the word unigrams and bigrams data were constructed twice: once excluding smileys and once including smileys. Three different types of support vector machine algorithms were used to execute the classification task with each of these data files.

All conditions show a classification performance above chance level. This means that all conditions (word unigrams, word bigrams and dependency triplets) provide some information about the author of the text. Because from 40 authors and equal amount of messages per author were used, the chance of predicting the correct author without knowledge is 2.5%. For all conditions a performance above this level is achieved. The word unigrams condition including smileys showed the best classification results. Therefore the words are indicative of the author. The percentage correctly classified instances is 15.85%, so this method can not be used to be certain that a text was written by the author the predicted by the support vector machine. Word unigrams are insufficient as the only type of feature for author identification in short texts.

The first research question was: Does the classification performance increase when the classification is based on bigrams of words instead of single words? To answer this question, the results of the two unigrams conditions are compared with the results of the bigrams conditions. The classification performance does not increase when the classification is based on bigrams of words compared to single words, it even decreases. Maybe this is because the texts are too short for regular re-occurrence of bigrams. This is supported by the data. In the bigrams data there are more features with a frequency of zero compared to the unigrams data. In theory, authors often may use the same word bigrams because of their unconscious use of grammar. But this might not be evident when considering only a low number of words. However McCombe [22] used larger sized texts and her methods also were not successful for word bigrams.

To answer the second research question, whether the classification performance increases when grammatical structures are used compared to classification based on word unigrams or word bigrams, extra conditions were created. I constructed three data files containing the following features: only triplets, triplets+unigrams and triplets+bigrams. The classification performance decreases when grammatical information in the form of dependency triplets are used, compared to word unigrams and word bigrams. I believe that this poor classification result is due to the incorrect grammatical usage of people. My examination of 100 sentences obtained from the fok.nl message board showed that the DUPIRA parser has problems handling grammatically incorrect sentences. Which is logical since it is based on the correct Dutch grammar. Therefore it makes sense that it can not produce correct triplets. This result does not mean that grammatical information in general is unsuitable for author identification. Hirst and Feguina [15] achieved successful results using bigrams of syntactic labels. Also Diederich [9] reported successful results using bigrams of tags for German texts. These type of features are based on grammatical information, but as opposed to the results of DUPIRA they do not specify relations between words. The results of the DUPIRA parser for texts from a message board are insufficient for author identification of short texts.

To answer the final research question, whether the classification performance increases when smileys are taken into account, I compared the results of the word unigrams and bigrams conditions excluding smileys with the results of these conditions including smileys. The results reveal that the use of smileys in the construction of features improves classification performance. Apparently smileys provide additional information about the author. This is possible because some people use a lot of smileys while others never use smileys. People often use the same smileys. The increase in performance is the largest when smileys are added to the word unigrams, compared to adding smileys to the word bigrams condition. When the smileys are combined with the bigrams they become a part of the word bigram, the bigram does not solely exist of words but may consist of a word and a smiley. For example a bigram can consist of the word 'yesterday' and the smiley 'laugh.gif', resulting in the bigram 'yesterday_laugh.gif'. The author might use the smiley 'laugh.gif' regularly, which in the unigrams condition results in a high frequency of the unigram 'laugh.gif'. In the bigrams condition the regular use of 'laugh.gif' is not apparent because this smiley is used before and after different words, which means that the smiley occurs in different bigrams. In one bigram it may occur with the word 'yesterday' while in another bigram it occurs with another word. The frequency of these bigrams does not necessarily increase because of the regular use of the smiley. So the regular use of the smiley 'laugh.gif' might not become evident. Therefore the influence of smileys is less when they are added to the construction of bigrams than when the smileys are added to the unigrams. The dependency triplet condition lacks the additional information provided by smileys.

The classifiers collect evidence for each class, in the form of probabilities. The instance gets the label of the class for which the classifier gathered the most evidence. For Weka's SMO algorithm the probabilities of each instance can be retrieved. My examination of these probability distributions showed that for about 12% of the incorrectly classified instances, the correct label appeared on the second or third place of the distribution. These results are not sufficient to conclude with a high amount of certainty that the correct author occurs in the top 3 that was predicted by the support vector machine. Only in 25% of the cases the correct author occurs in the top 3, when the classification is based on word unigrams including smileys.

Three different support vector machine implementations were used: SMO, LibSVM and LibLINEAR. LibLINEAR with normalization showed the best results. Probably because libLINEAR was designed to handle sparse data with many instances and features, even more than were used in the experiments of this thesis.

To summarize, for all conditions the performance was above chance level, so all provide some information about who the author of the text is. The classification performance does not increase when the classification is based on word bigrams compared to classification based on word unigrams as features for author identification. The performance also does not increase when the classification is based on grammatical information compared to the performance of the word unigrams and word bigrams. The classification performance does increase when smileys are considered. Overall the unigrams data including smileys gave the best result.

Chapter 5

Discussion

The results and conclusions of the experiments raise remarks and new questions. A first remark to be made is that all experiments are performed with Dutch texts. Therefore results can not straightforwardly be generalized to other languages. It is possible that the results in other languages are different, but it is also conceivable that there are some underlying mechanisms for writing texts that are applicable to different languages. It also depends on the language to what extent results can be generalized to that language. Dutch has more similarities with the English language compared to the Chinese language. Zheng et al. [30] compared results of English and Chinese author identification tasks. In both languages the performance increases when more features are used, but the percentage correctly classified instances was much higher for the English texts. The experiments performed for this thesis can also be performed using English texts, of course in combination with a natural language parser for English.

For the experiments in this thesis I made the choice to use 40 authors and 25 messages per author. This raises the following questions: do the results change when more or less authors are used and when more or less messages per author are used? And, what are the differences if the results change? This can be assessed by creating new data files based on more texts and/or authors and again using a support vector machine for classification, preferably the LibLINEAR algorithm with normalization.

Another interesting question is whether similar results are obtained when the experiments are conducted on data from a different source. The answer to this question can be found by generating the same type of data files using the same number or authors and text but for data from a different source. This source could be a different subsection of the fok.nl message board or another message board.

In the experiments, I did not replace capital characters with the lower case characters. As a result 'Word' and 'word' are different unigrams. Maybe this influences the classification performance. When capital characters are replaced by lower case characters there would only be the unigram 'word'. If an author used both 'Word' and 'word', after substituting the capital characters with lower case characters, the frequency of the unigram 'word' increases. Therefore it is possible that this influences the classification performance. To determine whether this is the case all capital characters could be replaced by lower case characters before the construction of the features and data files.

Also the results of the experiments raised new questions. By extending the unigrams to bigrams the performance dropped. This leads to the expectancy that when word 3-grams, also called word trigrams, are used as features the performance also drops or reaches the same level of performance as the word bigrams data. Only tests can reveal what the influence is of using n-grams with a higher value for n. However it is still possible that bigrams add information about the author to the information obtained from the word unigrams. To determine whether this is the case, the classification task can be performed with word unigrams and word bigrams as features.

A shortcoming of the experiments in this thesis is that there no feature selection was applied. Adding more features can increase the noise and hence decrease the classification performance. It is well possible that there are bigrams that are characteristic of the author of the text, while other bigrams are not. Applying feature selection before performing the classification can improve the classification performance, because the most characteristic features remain after feature selection.

The classification performance decreased when the classification was based on dependency triplets produced by the DUPIRA parser. As my examination of 100 sentences and their triplets showed, the DUPIRA parser has problems interpreting the grammatically incorrect sentences. This has a great impact on the classification performance using these dependency triplets. Another natural language parser for Dutch can be used to examine whether the same difficulties arise. However, another parser for Dutch will be based on the same grammar rules, so it is likely that another parser will also have difficulties parsing grammatically incorrect sentences. But it is conceivable that a parser handles such sentences differently than the DUPIRA parser, which may lead to other results. Note that a different parser might not produce dependency triplets but another type of output. So in that case, for a different parser the type of features are also different, which makes comparison less straightforward.

A final question that arose after examining the results of the experiments concerns the influence of smileys on author identification. The increase in performance is larger when smileys are added to word unigrams compared to word bigrams. This may be due to the fact that in the word unigrams the smileys occur in isolation, while in the bigrams data the smileys are part of the bigram. In the unigrams condition the smiley 'laugh.gif' occurs as a unigrams, for example 'laugh.gif' is a unigram. In the bigrams condition the smiley 'laugh.gif' becomes a part of the bigram, for example when the smiley is used after the word 'yesterday' the bigram becomes: 'yesterday_laugh.gif'. Maybe the increase in performance is higher when smileys are added to the bigrams without being part of the bigram. This hypothesis can be tested by adding the smileys as separate features, instead of as part of the word bigrams. In this case the bigrams are constructed from all the words and the smileys are added separately as unigrams. This gives more insight into the influence of smileys on author identification in short text. What specifically can be assessed is what the increase in performance is when the smileys are added as unigrams to the word bigrams, compared to only using the word bigrams (excluding smileys). The increase can be compared with the increase in performance when smileys were added to the word unigrams, which is 1.10% for the LibLIN Z algorithm. The increase in performance after adding smileys to the bigrams, as part of the bigrams, is 0.60%, for the LibLIN Z algorithm. It is conceivale that the increase after adding smileys as unigrams to the word bigrams instead of as part of the bigram approaches the 1.10% for the LibLIN Z condition. The smileys can also be added as separate features to the dependency triplets. It is possible that also in this case the classification performance increases.

Appendix A

ARFF format

ARFF stands for Attribute-Relation File Format, and was developed by the the Department of Computer Science of The University of Waikato to use with the their machine learning software Weka [14]. An arff file is composed of two main sections, the Header section and the Data section. The header section contains the name of the relation and a list of the attributes with their types. The name of the relation is defined using: @relation <relation-nameThe template for an attribute is: @attribute <attribute-name><data type>. If a name includes spaces the name must be surrounded by quotation marks. The type of attributes can be real, integer, nominal, date and string. For nominal attributes all possible values need to be stated explicitly. The template for a nominal attribute: @attribute <relation-name>{<value1>, <value2>, <value3>, ...}. The data section starts with '@data', followed by the declaration of the instances, each on a new line. The attribute values for an instance are separated by comma's and need to be in the same order as they were declared in the header section. Comments are proceeded by %.

In the example below the name of the relation is 'author', and the attribute names correspond to the word unigrams. All attributes have type 'real', except for the Author attribute which is a nominal attribute. The data section of this example contains 8 instances.

```
% This is an example
%
@relation author

@attribute Author {aw257772, aw147511, aw161975, aw211002, aw132428}
@attribute 'Greys' real
@attribute 'EurovisieSongFestival' real
@attribute 'uil' real
@attribute 'te' real
@attribute 'of' real
@attribute 'over' real
@attribute 'Uiteraard' real
@attribute 'alles' real
@attribute 'het' real
@attribute 'on' real

@data
aw257772,0.017,0.017,0.017,0.017,0.017,0.017,0.017,0.017,0.08, 0.0
aw147511,0.33,0.33,0.08,0.09,0.0,0.0,0.0,0.0,0.0,0.0
aw161975,0.0,0.0,0.27,0.0,0.45,0.45,0.45,0.0,0.0,0.0,
aw211002,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
aw132428,0.0,0.0,0.0,0.0,0.0,0.0,0.083,0.083,0.083,0.083
aw161975,0.0,0.0,0.27,0.0,0.45,0.45,0.45,0.0,0.0,0.0,
aw132428,0.0,0.0,0.0,0.0,0.0,0.0,0.083,0.083,0.083,0.083
aw211002,0.7,0.017,0.017,0.09,0.0,0.0,0.0,0.0,0.0,0.0
```

Appendix B

Weka output example

This appendix contains an example of the Weka output for one classification task. First, the time to build and test the model for one fold is outputted. The remainder of the output is divided in six sections. I will explain its contents before each section in italics.

Time taken to build model: 3.18 seconds
Time taken to test model on training data: 4.2 seconds

This section gives the performance measures of the training data.
=== Error on training data ===

```
Correctly Classified Instances        955          95.7874 %
Incorrectly Classified Instances      42           4.2126 %
Kappa statistic                       0.9568
Mean absolute error                   0.0021
Root mean squared error               0.0459
Relative absolute error               4.3207 %
Root relative squared error           29.3962 %
Coverage of cases (0.95 level)        95.7874 %
Mean rel. region size (0.95 level)    2.5 %
Total Number of Instances             997
```

This section contains detailed information about the performance on the training data.
=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.92	0.004	0.852	0.92	0.885	0.958	aw257772
	0.88	0.002	0.917	0.88	0.898	0.939	aw147511
	1	0.001	0.962	1	0.98	0.999	aw215495
	0.96	0	1	0.96	0.98	0.98	aw92054
	0.92	0	1	0.92	0.958	0.96	aw163650
	0.96	0	1	0.96	0.98	0.98	aw3694
	0.92	0	1	0.92	0.958	0.96	aw85235
	1	0	1	1	1	1	aw128272
	0.96	0	1	0.96	0.98	0.98	aw285129
	0.92	0	1	0.92	0.958	0.96	aw59211
	0.92	0	1	0.92	0.958	0.96	aw239202
	1	0	1	1	1	1	aw82877
	1	0	1	1	1	1	aw249756
	0.96	0	1	0.96	0.98	0.98	aw43556
	0.96	0	1	0.96	0.98	0.98	aw8443
	0.96	0	1	0.96	0.98	0.98	aw68576
	0.96	0	1	0.96	0.98	0.98	aw97341
	0.88	0	1	0.88	0.936	0.94	aw235382
	0.96	0	1	0.96	0.98	0.98	aw246444
	1	0.001	0.962	1	0.98	0.999	aw202014
	0.88	0	1	0.88	0.936	0.94	aw81329
	0.92	0	1	0.92	0.958	0.96	aw109683
	0.96	0	1	0.96	0.98	0.98	aw6941
	0.96	0	1	0.96	0.98	0.98	aw22592
	0.92	0.003	0.885	0.92	0.902	0.958	aw862
	0.96	0	1	0.96	0.98	0.98	aw162110
	0.96	0	1	0.96	0.98	0.98	aw16874
	1	0	1	1	1	1	aw204179
	1	0	1	1	1	1	aw28226
	0.96	0	1	0.96	0.98	0.98	aw16245
	0.96	0.001	0.96	0.96	0.96	0.979	aw268228
	0.96	0	1	0.96	0.98	0.98	aw223701
	1	0	1	1	1	1	aw298150
	1	0.001	0.962	1	0.98	0.999	aw137508
	1	0.029	0.472	1	0.641	0.986	aw80791
	1	0	1	1	1	1	aw182725
	0.96	0	1	0.96	0.98	0.98	aw148174
	0.96	0.001	0.96	0.96	0.96	0.979	aw7889
	1	0	1	1	1	1	aw214634
	0.92	0	1	0.92	0.958	0.96	aw111829
Weighted Avg.	0.958	0.001	0.973	0.958	0.962	0.978	

This is the confusion matrix of the training data.

=== Confusion Matrix ===

```
a b c d e f g h i j k l m n o p q r s t u v w x y z aa ab ac ad ae af ag ah ai aj ak al am an - classified as
23 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 — a = aw257772
0 22 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 — b = aw147511
0 0 25 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 — c = aw215495
0 1 0 24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 — d = aw92054
1 0 0 0 23 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 — e = aw163650
0 0 0 0 0 24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 — f = aw3694
1 0 0 0 0 0 23 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 — g = aw85235
0 0 0 0 0 0 0 25 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 — h = aw128272
0 0 0 0 0 0 0 0 24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 — i = aw285129
0 0 0 0 0 0 0 0 0 23 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 — j = aw59211
1 0 0 0 0 0 0 0 0 0 23 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 — k = aw239202
0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 — l = aw82877
0 0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 — m = aw249756
0 0 0 0 0 0 0 0 0 0 0 0 0 24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 — n = aw43556
0 0 0 0 0 0 0 0 0 0 0 0 0 0 24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 — o = aw8443
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 — p = aw68576
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 — q = aw97341
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 22 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 — r = aw235382
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 — s = aw246444
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 — t = aw202014
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 22 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 — u = aw81329
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 — v = aw109683
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 — w = aw6941
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 24 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 — x = aw22592
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 23 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 — y = aw862
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 24 0 0 0 0 0 0 0 1 0 0 0 0 0 0 — z = aw162110
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 24 0 0 0 0 0 0 0 1 0 0 0 0 0 — aa = aw16874
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 0 0 0 0 0 0 0 0 0 — ab = aw204179
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 0 0 0 0 0 0 0 0 — ac = aw282267
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 24 0 0 0 0 1 0 0 0 0 0 — ad = aw16245
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 24 0 0 0 0 0 0 0 0 0 — ae = aw268228
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 24 0 0 0 0 0 0 0 0 — af = aw223701
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 0 0 0 0 — ag = aw298150
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 0 0 0 — ah = aw137508
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 0 0 — ai = aw80791
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 0 0 0 0 — aj = aw182725
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 24 0 0 0 — ak = aw148174
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 24 0 0 — al = aw7889
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 22 0 — am = aw214634
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 23 — an = aw111829
```

This section contains the performance measures of the test data.

=== Stratified cross-validation ===

```
Correctly Classified Instances        136          13.6409 %
Incorrectly Classified Instances      861          86.3591 %
Kappa statistic                       0.1143
Mean absolute error                   0.0432
Root mean squared error               0.2078
Relative absolute error               88.5646 %
Root relative squared error           133.0819 %
Coverage of cases (0.95 level)        13.6409 %
Mean rel. region size (0.95 level)    2.5 %
Total Number of Instances             997
```

This section gives the detailed information about the performance on the test data.
=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.28	0.037	0.163	0.28	0.206	0.621	aw257772
	0	0.044	0	0	0	0.478	aw147511
	0.12	0.008	0.273	0.12	0.167	0.556	aw215495
	0	0.006	0	0	0	0.497	aw92054
	0.36	0.012	0.429	0.36	0.391	0.674	aw163650
	0.16	0.017	0.19	0.16	0.174	0.571	aw3694
	0.04	0.039	0.026	0.04	0.031	0.5	aw85235
	0.2	0.01	0.333	0.2	0.25	0.595	aw128272
	0	0.006	0	0	0	0.497	aw285129
	0.2	0.009	0.357	0.2	0.256	0.595	aw59211
	0.2	0.013	0.278	0.2	0.233	0.593	aw239202
	0.12	0.025	0.111	0.12	0.115	0.548	aw82877
	0.12	0.019	0.143	0.12	0.13	0.551	aw249756
	0	0.019	0	0	0	0.491	aw43556
	0.12	0.011	0.214	0.12	0.154	0.554	aw8443
	0.04	0.015	0.063	0.04	0.049	0.512	aw68576
	0.16	0.015	0.211	0.16	0.182	0.572	aw97341
	0.04	0.029	0.034	0.04	0.037	0.506	aw235382
	0.16	0.01	0.286	0.16	0.205	0.575	aw246444
	0.08	0.032	0.061	0.08	0.069	0.524	aw202014
	0.04	0.013	0.071	0.04	0.051	0.513	aw81329
	0.04	0.026	0.038	0.04	0.039	0.507	aw109683
	0.12	0.012	0.2	0.12	0.15	0.554	aw6941
	0.2	0.075	0.064	0.2	0.097	0.562	aw22592
	0.24	0.036	0.146	0.24	0.182	0.602	aw862
	0.12	0.003	0.5	0.12	0.194	0.558	aw162110
	0	0.033	0	0	0	0.484	aw16874
	0.08	0.009	0.182	0.08	0.111	0.535	aw204179
	0	0.013	0	0	0	0.493	aw282267
	0.16	0.026	0.138	0.16	0.148	0.567	aw16245
	0.28	0.015	0.318	0.28	0.298	0.632	aw268228
	0.08	0.012	0.143	0.08	0.103	0.534	aw223701
	0.24	0.013	0.316	0.24	0.273	0.613	aw298150
	0	0.027	0	0	0	0.487	aw137508
	0.32	0.11	0.07	0.32	0.114	0.605	aw80791
	0.24	0.008	0.429	0.24	0.308	0.616	aw182725
	0.24	0.014	0.3	0.24	0.267	0.613	aw148174
	0.16	0.001	0.8	0.16	0.267	0.579	aw7889
	0.318	0.021	0.259	0.318	0.286	0.649	aw214634
	0.2	0.038	0.119	0.2	0.149	0.581	aw111829
Weighted Avg.	0.136	0.022	0.181	0.136	0.142	0.557	

This is the confusion matrix of the test data.

```
=== Confusion Matrix ===
a b c d e f g h i j k l m n o p q r s t u v w x y z aa ab ac ad ae af ag ah ai aj ak al am an ¡– classified as
7 1 0 1 1 0 1 1 0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 1 6 0 0 0 0 0 — a = aw257772
1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 2 1 1 0 2 0 0 2 0 1 1 0 0 0 2 6 0 2 0 0 1 — b = aw147511
0 1 3 0 0 1 0 0 0 1 0 2 1 1 0 0 1 1 0 1 1 0 0 2 1 0 1 0 0 1 0 2 0 1 2 0 0 0 1 0 — c = aw215495
1 1 0 0 0 0 3 1 1 1 1 1 0 0 1 1 0 2 0 0 0 0 0 3 0 0 0 1 0 1 0 1 1 0 1 1 0 0 1 1 — d = aw92054
3 0 0 0 9 0 2 0 0 0 0 1 0 0 0 1 1 0 0 1 0 0 0 2 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 2 — e = aw163650
3 0 0 0 1 4 1 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0 0 3 2 0 0 0 0 0 0 0 1 0 1 3 0 2 0 0 0 — f = aw3694
2 0 0 1 1 1 1 0 0 0 1 1 0 0 2 0 1 0 0 1 0 2 0 4 0 0 2 0 0 0 1 0 0 1 0 0 0 0 0 3 — g = aw85235
0 0 0 0 2 0 0 5 1 0 1 0 0 0 0 1 0 1 0 0 0 2 1 1 2 0 1 0 0 1 2 1 0 0 1 0 0 0 2 0 — h = aw128272
0 2 0 0 0 0 1 0 0 0 1 1 0 1 0 0 0 0 2 1 0 1 0 5 0 0 1 1 1 0 0 0 0 0 4 0 1 0 1 1 — i = aw285129
0 0 0 1 1 1 1 0 0 5 1 1 1 0 0 1 0 1 0 0 1 0 1 1 3 0 0 0 0 0 0 0 0 2 2 0 0 0 0 1 — j = aw59211
2 1 0 0 0 0 1 0 0 0 5 2 0 0 0 1 0 0 1 1 0 0 1 3 2 0 1 1 0 0 0 1 0 1 0 0 0 0 1 0 — k = aw239202
0 0 1 0 0 0 2 0 1 0 1 3 1 1 0 0 1 1 0 0 0 1 0 4 0 0 1 1 0 0 1 1 0 0 1 1 0 0 2 0 — l = aw82877
1 2 0 1 0 1 1 0 0 0 0 0 3 1 1 0 1 0 1 1 1 1 0 1 0 0 0 0 0 1 0 0 2 1 2 1 0 0 1 0 — m = aw249756
1 0 0 0 0 0 2 1 0 0 0 0 1 0 0 1 0 0 0 0 0 2 1 3 1 1 1 0 0 2 0 0 4 1 1 0 1 0 0 1 — n = aw43556
1 0 0 1 1 1 0 0 0 0 1 1 0 0 3 1 0 0 0 0 0 0 1 0 0 0 3 2 2 0 0 0 1 1 3 1 1 0 0 0 — o = aw8443
0 1 0 0 0 1 4 0 0 0 0 1 1 0 0 1 0 0 0 2 0 0 2 4 1 0 3 0 0 1 1 0 0 0 1 0 1 0 0 0 — p = aw68576
0 2 1 0 1 0 2 0 0 0 0 0 2 1 0 0 4 0 1 2 1 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 2 0 2 0 — q = aw97341
2 1 0 0 1 0 1 1 0 1 0 1 0 0 0 0 0 1 0 1 0 0 1 1 2 0 0 0 0 0 1 0 0 0 7 0 0 0 0 3 — r = aw235382
0 3 0 0 1 1 3 0 0 0 0 0 0 0 0 1 1 1 4 1 0 2 0 1 0 0 1 0 0 1 0 0 0 1 2 0 0 0 0 1 — s = aw246444
2 3 0 0 0 1 1 0 0 0 1 1 0 0 0 0 0 2 0 2 0 0 0 1 3 0 1 1 0 1 1 0 0 0 2 0 0 0 0 2 — t = aw202014
0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 2 1 0 0 4 3 0 0 0 1 2 0 0 0 1 5 0 0 0 0 1 — u = aw81329
1 3 0 0 0 0 1 1 0 0 0 0 0 0 0 2 1 2 2 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 3 0 1 0 0 1 — v = aw109683
0 3 1 1 0 0 1 0 0 1 1 2 0 1 0 0 0 0 0 0 1 0 3 0 1 0 1 0 2 1 0 0 0 0 3 0 0 0 1 1 — w = aw6941
0 1 1 0 1 0 1 0 0 0 0 0 0 2 0 0 0 0 0 2 1 0 0 5 1 0 0 0 0 0 0 0 0 1 8 0 0 0 0 1 — x = aw22592
1 1 1 0 0 0 0 0 0 2 0 1 0 1 0 0 1 0 0 1 0 3 0 2 6 0 0 0 1 0 1 0 0 0 2 0 0 0 0 1 — y = aw862
0 3 0 0 0 0 1 1 0 0 0 0 0 0 1 0 2 0 0 0 1 0 1 3 2 3 1 0 0 0 0 1 0 0 2 0 0 0 3 0 — z = aw162110
0 1 0 0 0 0 2 0 0 0 1 0 0 0 2 1 0 2 0 2 1 1 0 2 0 0 0 0 0 0 1 0 1 2 4 0 0 0 2 0 — aa = aw16874
2 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 4 2 0 1 2 0 2 1 1 0 0 3 1 0 0 0 2 — ab = aw204179
2 2 0 0 0 0 0 0 0 1 1 1 0 0 1 2 0 0 2 0 0 1 2 0 0 2 0 0 0 1 0 0 1 3 1 1 0 0 1 — ac = aw282267
1 1 0 0 0 0 1 0 0 0 0 0 2 1 0 1 0 1 1 0 0 1 0 1 0 0 0 0 0 4 0 0 0 0 7 0 1 0 0 2 — ad = aw16245
1 1 0 0 0 0 1 2 0 0 0 1 0 0 0 1 0 0 0 0 0 1 1 0 2 0 1 1 0 2 7 0 0 0 0 0 1 0 1 1 — ae = aw268228
3 0 0 0 0 3 0 0 0 0 0 0 1 0 0 0 0 1 0 0 2 2 0 1 0 0 2 0 0 0 0 2 0 3 3 0 0 0 0 2 — af = aw223701
1 3 0 0 0 0 0 1 0 0 0 0 3 1 1 0 0 1 1 0 0 0 0 1 0 0 2 1 0 1 1 0 6 0 1 0 0 0 0 0 — ag = aw298150
0 1 0 0 0 0 0 0 0 2 0 1 0 2 1 0 0 0 1 2 1 1 0 1 2 0 1 0 0 0 0 1 1 0 4 1 0 0 0 2 — ah = aw137508
3 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 3 0 0 0 0 1 0 1 0 0 0 1 2 1 1 0 2 8 0 0 0 0 0 — ai = aw80791
0 0 1 0 0 1 0 0 0 0 0 1 2 3 0 0 1 1 0 0 1 0 0 2 0 0 0 0 0 1 0 0 0 2 0 6 0 0 1 2 — aj = aw182725
0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 2 0 2 0 2 0 4 0 0 0 0 0 0 1 0 0 0 5 0 6 0 0 1 — ak = aw148174
2 1 0 0 0 1 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 2 0 0 0 0 2 1 0 0 1 1 4 1 0 4 0 2 — al = aw7889
0 1 1 0 0 1 0 0 0 1 0 3 0 0 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 0 0 0 2 0 0 0 7 1 — am = aw214634
0 2 0 0 0 2 1 1 0 0 0 0 0 0 0 0 0 2 0 2 0 0 0 0 2 0 1 0 0 1 0 1 1 0 2 0 0 1 1 5 — an = aw111829
```

Appendix C

Performance measures

An overview of the performance measures produced by Weka:

- Correctly Classified Instances
- Incorrectly Classified Instances
- Kappa statistic
- Mean absolute error
- Root mean squared error
- Relative absolute error
- Root relative squared error
- TP rate
- FP rate
- Precision
- Recall
- F-measure
- ROC Area

Correctly Classified Instances gives the number and the percentage of instances classified correctly. Logically follows that the percentage of instances classified incorrectly, is 100 - 'Percentage correctly classified instances'.
The *Kappa statistic* measures the amount of agreement between the predictions of a classifier and the actual class label. The Kappa statistic takes into account the chance level, so it indicates whether this agreement is above chance level. The Kappa statistic gives a number between 0 and 1. Where 0 means that the predictions and actual classes are not correlated and 1 complete agreement. This measure will make the same decisions as the percentage of correctly classified instances. So when a classifier has a higher percentage of correctly classified instances the Kappa statistic will also give a higher value. Compare Figure C.1 with Figure 3.4, the Kappa statistic makes the same decision as the percentage correctly classified instances.
The *mean absolute error* also measures how close the prediction of the classifier are to the actual class. Because the data is nominal this distance is measured using the probability distributions. The probability distribution of the classifier is compared to the actual distribution. In this actual distribution the probability of the actual class is 1 and 0 for the other classes. The root mean squared error also results in a number between 0 and 1, but uses the squared differences.
The relative errors, *relative absolute error and root relative squared error*, take into account the chance level. The root relative squared error is calculated as 100*(root mean squared error of classifier/root mean squared error when classifying by chance). Lower relative measures indicate a better performance.
The TP rate, FP rate, Precision, Recall and F-measure can best be explained with a binary classification problem. Remember that the multi-class classification problem is solved by constructing classifiers for each class combination. For each binary classification problem a confusion matrix can be constructed. An example of how such a matrix looks is given in Table C.1.

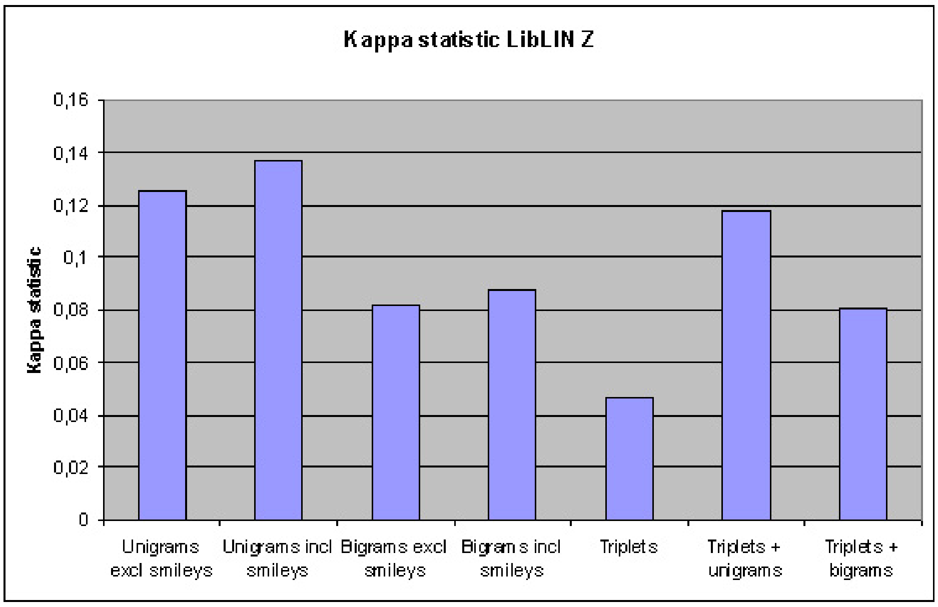


Figure C.1: The Kappa statistics for the LibLIN algorithm with normalization (Z).

		Predicted class label	
		+	-
Actual class label	+	True positive (TP)	False negative (FN)
	-	False positive (FP)	True negative (TN)

Table C.1: A confusion matrix for a binary classification problem.

True positive rate = TP/(TP+FN), where
TP = the number of positive samples correctly predicted by the classification method.
FN = the number of positive samples incorrectly predicted by the classification method.
So, the TP rate is the fraction of positive samples predicted correctly. The TP rate is a number between 0 and 1, 0 meaning that all samples are incorrectly classified and 1 all instances are classified correctly. So this gives the same results as the percentage correctly classified instances, but divided by 100.

False positive rate = FP/(TN+FP), where
FP = the number of negative samples incorrectly predicted as the positive class.
TN = the number of negative samples correctly predicted by the classification method.
So, the FP rate is the fraction of negative samples predicted as the positive class. The FP rate is a number between 0 and 1, where a lower rate is better.

Precision = TP/(TP+FP), where
TP = the number of positive samples correctly predicted by the classification method.
FP = the number of negative samples incorrectly predicted as the positive class.
Precision is the fraction of samples that actually belong to the positive class of the group of all samples the classifier labeled as positive. The higher the precision measure the lower the number false positives.

Recall = TP/(TP+FN), where
TP = the number of positive samples correctly predicted by the classification method.
FN = the number of positive samples incorrectly predicted by the classification method.
Recall is the fraction of positive examples correctly predicted by the classifier. If the recall is high the classification algorithm incorrectly classified only a few positive samples. The values of the true positive rate and recall are the same.

You want a classifier that maximizes precision and recall. The *F-measure* is a measure that combines precision and recall. So a high value for the F-measure means that precision and recall are high.
ROC stands for 'Receiver Operating Characteristic'. *ROC Area* refers to the area below the ROC curve. This curve is created by plotting the FP rate along the x-axis and the TP rate along the y-axis. The curve connects the point where TP rate = 0 and FP rate = 0 with the point where TP rate = 1 and FP rate = 1. If the classifier performs perfectly the area is 1. If the performance of the classifier is equal to random guessing the area under the curve is 0.5 [6, 26, 31].

Appendix D

Detailed classification results

This appendix contains the detailed classification results for all conditions: unigrams excluding and including smileys, bigrams excluding and including smileys, dependency triplets, unigrams+triplets, bigrams+triplets. For all algorithms the performance measures calculated by Weka are summarized in tables.

D.1 Word unigrams

This section contains the detailed classification results of the unigrams conditions excluding and including smileys.

D.1.1 Unigrams excluding smileys

	SMO	LibSVM	LibSVM Z	LibLIN	LibLIN Z	LibLIN S4	LibLIN S4 Z
Correctly Classified	9.33	11.43	11.63	10.83	14.74	12.44	14.24
Incorrectly Classified	90.67	88.57	88.65	89.17	85.26	87.56	85.76
Kappa statistic	0.07	0.09	0.09	0.09	0.13	0.10	0.12
Mean absolute error	0.05	0.04	0.04	0.045	0.04	0.044	0.04
Root mean squared e	0.16	0.21	0.21	0.21	0.21	0.21	0.21
Relative absolute e	99.38	90.83	90.62	91.44	87.43	89.80	87.95
Root relative squared e	99.82	134.77	134.62	135.23	132.23	134.01	132.62
Coverage of cases	86.76	11.43	11.63	10.83	14.74	12.44	14.24
Mean rel. region size	77.51	2.5	2.5	2.5	2.5	2.5	2.5
TP rate	0.09	0.114	0.12	0.11	0.147	0.12	0.14
FP rate	0.02	0.032	0.02	0.02	0.022	0.02	0.02
Precision	0.16	0.132	0.17	0.10	0.142	0.12	0.14
Recall	0.09	0.114	0.12	0.11	0.147	0.12	0.14
F-measure	0.10	0.106	0.12	0.10	0.141	0.12	0.14
ROC Area	0.64	0.546	0.55	0.54	0.563	0.55	0.56

Table D.1: Results for the unigrams condition excluding smileys.

D.1.2 Unigrams including smileys

	SMO	LibSVM	LibSVM Z	LibLIN	LibLIN Z	LibLIN S4	LibLIN S4 Z
Correctly Classified	13.24	15.55	13.64	14.74	15.85	16.05	15.65
Incorrectly Classified	86.76	84.45	86.36	85.26	84.15	83.95	84.35
Kappa statistic	0.11	0.13	0.11	0.13	0.14	0.14	0.13
Mean absolute error	0.05	0.04	0.04	0.04	0.04	0.04	0.04
Root mean squared e	0.16	0.21	0.21	0.21	0.21	0.20	0.21
Relative absolute e	99.19	86.61	88.56	87.43	86.30	86.10	86.51
Root relative squared e	99.63	131.61	133.08	132.23	131.37	131.21	131.53
Coverage of cases	88.06	15.55	13.64	14.74	15.85	16.05	15.65
Mean rel. region size	77.53	2.5	2.5	2.5	2.5	2.5	2.5
TP rate	0.13	0.16	0.14	0.15	0.16	0.16	0.16
FP rate	0.02	0.02	0.02	0.02	0.02	0.02	0.02
Precision	0.18	0.20	0.18	0.12	0.16	0.14	0.16
Recall	0.13	0.16	0.14	0.15	0.16	0.16	0.16
F-measure	0.14	0.16	0.14	0.13	0.16	0.15	0.15
ROC Area	0.67	0.57	0.56	0.56	0.57	0.57	0.57

Table D.2: Results for the unigrams condition including smileys.

D.2 Word bigrams

This section contains the detailed classification results of the bigrams conditions excluding and including smileys.

D.2.1 Bigrams excluding smileys

	SMO	LibSVM	LibSVM Z	LibLIN	LibLIN Z	LibLIN S4	LibLIN S4 Z
Correctly Classified	3.11	5.82	3.41	8.33	10.43	8.12	9.63
Incorrectly Classified	96.89	94.18	96.59	91.68	89.57	91.88	90.37
Kappa statistic	00.01	0.03	0.01	0.06	0.08	0.06	0.07
Mean absolute error	0.05	0.05	0.05	0.05	0.04	0.05	0.05
Root mean squared e	0.16	0.22	0.22	0.21	0.21	0.21	0.21
Relative absolute e	99.91	96.59	99.06	94.02	91.86	94.22	92.68
Root relative squared e	100.34	138.98	140.74	137.12	135.53	137.27	136.14
Coverage of cases	79.74	5.82	3.41	8.33	10.43	8.12	9.63
Mean rel. region size	77.53	2.5	2.5	2.5	2.5	2.5	2.5
TP rate	0.03	0.06	0.03	0.08	0.10	0.08	0.10
FP rate	0.03	0.02	0.03	0.02	0.02	0.02	0.02
Precision	0.11	0.13	0.14	0.09	0.12	0.08	0.10
Recall	0.03	0.06	0.03	0.08	0.10	0.08	0.10
F-measure	0.01	0.05	0.02	0.08	0.10	0.08	0.09
ROC Area	0.54	0.52	0.51	0.53	0.54	0.53	0.54

Table D.3: Results for the bigrams condition excluding smileys.

D.2.2 Bigrams including smileys

	SMO	LibSVM	LibSVM Z	LibLIN	LibLIN Z	LibLIN S4	LibLIN S4 Z
Correctly Classified	3.11	6.62	3.51	9.63	11.03	9.53	11.13
Incorrectly Classified	96.89	93.38	96.49	90.37	88.97	90.47	88.87
Kappa statistic	0.01	0.04	0.01	0.07	0.09	0.07	0.09
Mean absolute error	0.05	0.05	0.05	0.05	0.04	0.05	0.04
Root mean squared e	0.16	0.22	0.22	0.21	0.21	0.21	0.21
Relative absolute e	99.89	95.76	98.95	92.68	91.24	92.78	91.14
Root relative squared e	100.33	138.39	140.67	136.14	135.08	136.21	135.00
Coverage of cases	79.34	6.62	3.51	9.63	11.03	9.53	11.13
Mean rel. region size	77.55	2.5	2.5	2.5	2.5	2.5	2.5
TP rate	0.03	0.07	0.04	0.10	0.11	0.10	0.11
FP rate	0.03	0.02	0.03	0.02	0.02	0.02	0.02
Precision	0.10	0.18	0.13	0.11	0.13	0.09	0.13
Recall	0.03	0.07	0.04	0.10	0.11	0.10	0.11
F-measure	0.02	0.06	0.02	0.10	0.11	0.09	0.11
ROC Area	0.55	0.52	0.51	0.54	0.54	0.54	0.54

Table D.4: Results for the bigrams condition including smileys.

D.3 Dependency triplets

This section contains the detailed classification results of the conditions including dependency triplets: only triplets, unigrams+triplets and bigrams+triplets.

D.3.1 Only triplets

	SMO	LibSVM	LibSVM Z	LibLIN	LibLIN Z	LibLIN S4	LibLIN S4 Z
Correctly Classified	4.21	4.61	3.71	6.42	7.02	6.12	7.42
Incorrectly Classified	95.79	95.39	96.29	93.58	92.98	93.88	92.58
Kappa statistic	0.02	0.022	0.01	0.04	0.05	0.04	0.05
Mean absolute error	0.05	0.048	0.05	0.05	0.05	0.05	0.05
Root mean squared e	0.16	0.22	0.22	0.22	0.22	0.22	0.22
Relative absolute e	99.91	97.82	98.75	95.97	95.35	96.28	94.94
Root relative squared e	100.34	139.86	140.52	138.53	138.09	138.76	137.79
Coverage of cases	79.64	4.61	3.71	6.42	7.02	6.12	7.42
Mean rel. region size	77.57	2.5	2.5	2.5	2.5	2.5	2.5
TP rate	0.04	0.05	0.04	0.06	0.07	0.06	0.07
FP rate	0.03	0.03	0.03	0.02	0.02	0.02	0.02
Precision	0.16	0.09	0.09	0.10	0.08	0.08	0.10
Recall	0.04	0.05	0.04	0.06	0.07	0.06	0.07
F-measure	0.03	0.04	0.03	0.06	0.07	0.06	0.07
ROC Area	0.52	0.51	0.51	0.52	0.52	0.52	0.53

Table D.5: Results for the dependency triplets condition.

D.3.2 Unigrams and triplets

	SMO	LibSVM	LibSVM Z	LibLIN	LibLIN Z	LibLIN S4	LibLIN S4 Z
Correctly Classified	5.52	10.93	5.72	10.13	13.94	11.84	13.34
Incorrectly Classified	94.48	89.07	94.28	89.87	86.06	88.16	86.66
Kappa statistic	0.03	0.09	0.03	0.08	0.12	0.10	0.11
Mean absolute error	0.05	0.04	0.05	0.04	0.04	0.04	0.04
Root mean squared e	0.16	0.21	0.22	0.21	0.21	0.21	0.21
Relative absolute e	99.66	91.34	96.69	92.16	88.26	90.42	88.87
Root relative squared e	100.10	135.15	139.05	135.76	132.85	134.47	133.31
Coverage of cases	83.65	10.93	5.72	10.13	13.94	11.84	13.34
Mean rel. region size	77.53	2.5	2.5	2.5	2.5	2.5	2.5
TP rate	0.06	0.11	0.06	0.10	0.14	0.12	0.13
FP rate	0.02	0.02	0.02	0.02	0.02	0.02	0.02
Precision	0.13	0.13	0.12	0.09	0.14	0.11	0.13
Recall	0.06	0.11	0.06	0.10	0.14	0.12	0.13
F-measure	0.05	0.09	0.05	0.09	0.13	0.11	0.13
ROC Area	0.61	0.54	0.52	0.54	0.56	0.55	0.56

Table D.6: Results for the unigrams+triplets condition.

D.3.3 Bigrams and triplets

	SMO	LibSVM	LibSVM Z	LibLIN	LibLIN Z	LibLIN S4	LibLIN S4 Z
Correctly Classified	3.21	6.22	3.51	8.73	10.33	9.32	10.13
Incorrectly Classified	96.79	93.78	96.49	91.27	89.67	90.67	89.87
Kappa statistic	0.01	0.04	0.01	0.06	0.08	0.07	0.08
Mean absolute error	0.05	0.05	0.05	0.05	0.04	0.05	0.045
Root mean squared e	0.16	0.22	0.22	0.21	0.21	0.21	0.21
Relative absolute e	99.92	96.18	98.95	93.60	91.96	92.99	92.16
Root relative squared e	100.36	138.68	140.67	136.82	135.61	136.36	135.76
Coverage of cases	79.84	6.22	3.51	8.73	10.33	9.33	10.13
Mean rel. region size	78.02	2.5	2.5	2.5	2.5	2.5	2.5
TP rate	0.03	0.16	0.04	0.09	0.10	0.09	0.10
FP rate	0.03	0.02	0.03	0.02	0.02	0.02	0.02
Precision	0.11	0.20	0.15	0.10	0.12	0.10	0.11
Recall	0.03	0.16	0.04	0.09	0.10	0.09	0.10
F-measure	0.01	0.16	0.02	0.09	0.10	0.09	0.10
ROC Area	0.54	0.57	0.51	0.53	0.54	0.54	0.54

Table D.7: Results for the bigrams+triplets condition.

References

[1] A.T. Arampatzis, Th.P. van der Weide, C.H.A. Koster, and P. van Bommel. An Evaluation of Linguistically–motivated Indexing Schemes. In *Proceedings of the 22nd BCS–IRSG Colloquium on IR Research*, pages 34–45, April 2000.

[2] 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):121–131, 1996.

[3] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[4] C. Chaski. Who wrote it? steps towards a science of authorship identification. *National Institute of Justice Journal*, September 1997.

[5] R. Clement and D. Sharp. Ngram and baysian classification of documents for topic and authorship. *Literary and Linguistic Computing*, 18(4), 2003.

[6] M. Corney, A. Anderson, G. Mohay, and O. de Vel. Identifying the authors of suspect mail. *Communications of the ACM*, 2001.

[7] Malcolm Corney. Analysing e-mail text authorship for forensic purposes. Master's thesis, Queensland University of Technology, Brisbane Australia, 2003.

[8] Koby Crammer, Yoram Singer, Nello Cristianini, John Shawe-Taylor, and Bob Williamson. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

[9] Joachim Diederich, Jörg Kindermann, Edda Leopold, and Gerhard Paass. Authorship attribution with support vector machines. *Applied Intelligence*, 19(1-2):109–123, 2003.

[10] Yasser EL-Manzalawy and Vasant Honavar. *WLSVM: Integrating LibSVM into Weka Environment*, 2005. Software available at http://www.cs.iastate.edu/ yasser/wlsvm.

[11] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[12] Quentin Gouedard. The largest message boards on the web!, http://www.big-boards.com, 2010.

[13] Neil Graham, Graeme Hirst, and Bhaskara Marthi. Segmenting documents by stylistic character. *Nat. Lang. Eng.*, 11(4):397–415, 2005.

[14] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: An update; SIGKDD Explorations, 2009. Software available at http://www.cs.waikato.ac.nz/ ml/weka/.

[15] G. Hirst and Ol'ga Feiguina. Bigrams of syntactic labels for authorship discrimination of short texts. *Literary and Linguistic Computing*, 22(4), 2007.

[16] J.F. Hoorn, S.L. Frank, W. Kowalczyk, and F. van der Ham. Neural network identification of poets using letter sequences. *Lit Linguist Computing*, 14(3):311–338, 1999.

[17] John Houvardas and Efstathios Stamatatos. N-gram feature selection for authorship identification. In *AIMSA*, pages 77–86, 2006.

[18] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning (ECML)*, pages 137–142, Berlin, 1998. Springer.

[19] B. Kjell. Authorship determination using letter pair frequency features with neural network classifiers. In *in Literacy and Linguistic Computing 9*, 1994.

[20] C.H.A. Koster, M. Seutter, and S. Seibert. Parsing the medline corpus. In *Proceedings RANLP 2007*, pages 325–329, 2007.

[21] Cornelis H.A. Koster and Jean G. Beney. Phrase-based document categorization revisited. In *PaIR '09: Proceeding of the 2nd international workshop on Patent information retrieval*, pages 49–56, New York, NY, USA, 2009. ACM.

[22] Niamh McCombe. Methods of author identification. Master's thesis, Trinity College, Dublin Ireland, 2002.

[23] John C. Platt. Fast training of support vector machines using sequential minimal optimization. pages 185–208, Cambridge, MA, USA, 1999. MIT Press.

[24] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[25] E. Stamatatos, N. Fakotakis, and G. Kokkinakis. Computer-based authorship attribution without lexical measures. In *Computers and the Humanities*, pages 193–214, 2001.

[26] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[27] Yuta Tsuboi and Yuji Matsumoto. Authorship identification for heterogeneous documents. Master's thesis, Nara Institute of Science and Technology, Japan, 2002.

[28] L. van der Knaap and F. A. Grootjen. Author identification in chatlogs using formal concept analysis. In M. M. Dastani and E. de Jong, editors, *Proceedings of the 19th Belgian-Dutch Conference on Artificial Intelligence (BNAIC2007)*, pages 181–188, Utrecht, the Netherlands, November 2007.

[29] O. de Vel, A. Anderson, M. Corney, and G. Mohay. Mining e-mail content for author identification forensics. *SIGMOD Rec.*, 30(4):55–64, 2001.

[30] Rong Zheng, Jiexun Li, Hsinchun Chen, and Zan Huang. A framework for authorship identification of online messages: Writing-style features and classification techniques. *J. Am. Soc. Inf. Sci. Technol.*, 57(3):378–393, 2006.

[31] Rong Zheng, Yi Qin, Zan Huang, and Hsinchun Chen. Authorship analysis in cybercrime investigation. In *ISI'03: Proceedings of the 1st NSF/NIJ conference on Intelligence and security informatics*, pages 59–73, Berlin, Heidelberg, 2003. Springer-Verlag.