An Authorship Study on the Letters of Saint Paul

Katarina Laken
s4448774
29 July 2018
Linguistics
Hans van Halteren
Matthijs den Dulk

Preface

I more or less accidentally enrolled in the bachelor's program in Theology at Radboud University. I already was a student in the Linguistics program, and I quite impulsively decided that I wanted to do something extra. The idea of studying old texts and languages appealed to me, so I picked Theology. Most of all I enjoyed biblical exegesis. And although I dropped out of the program after two years, I knew I was not yet done with the field. My bachelor's thesis was a good opportunity to combine linguistics and theology. I had always liked the courses with a focus on computational linguistics, and when I first heard about authorship attribution, I immediately started thinking about applying these methods to the Bible.

The process was not always easy, and many steps took considerably longer than I had anticipated. However, overall, I enjoyed working on this thesis. I greatly enhanced my programming skills and biblical knowledge. After conducting the research, I became aware of several ways the research could be improved, and I would like to see if it would change the results. However, to be honest, I am also relieved to finish my bachelor's and move on to the next step.

I would like to thank everybody who has helped me write this thesis. First of all, my supervisor Hans van Halteren, who conducted the analyses of the features, one of the most crucial parts of the research, and who was so kind to give feedback on the first version of this thesis when he was on a holiday. Then, my proofreader Dennis Joosen, who patiently pointed out all my typos and spelling errors. And of course Martijn Beukenhorst, my beloved boyfriend, for the proofreading, the brainstorming, the biblical knowledge, the moral support, and the food. Thanks to all of you, I am very grateful for everything.

Table of Contents

Abstract

The Letters of Paul are an important part of the New Testament canon, but for several of them, the authorship is disputed. In this study, I applied authorship attribution techniques to the Letters of St. Paul. Samples were taken from the letters commonly accepted as genuinely Pauline, the disputed letters (Colossians, Ephesians, and 2 Thessalonians), and several non-Pauline letters from the New Testament. Features were divided into general text measurements, syntactic features, vocabulary features and character *n*-grams. All types of features did relatively well at distinguishing between Pauline and non-Pauline samples. Although the results are not unambiguous, the careful conclusion is that Ephesians, Colossians and 2 Thessalonians are probably not written by Paul. 2 Thessalonians significantly deviated from the Pauline letters in overall measures, whereas Ephesians and Colossians deviated in syntactic and vocabulary features.

"We ask you not to let your mind be quickly shaken or be troubled, neither in spirit nor by word, nor by letter coming as though from us…" (2 Thess. 2:2, MEV).

Introduction

Many scholars reckon that it was not Jesus Christ who founded Christianity as a religion. Even though he is the central figure in Christian belief, it was Saint Paul who established and shaped the religion we know today. The epistles of Saint Paul are the oldest known Christian writings: the First Letter to the Thessalonians is dated around 50 CE.

The epistles of Saint Paul as we have them today are part of the correspondence between Paul and several Christian communities and churches throughout the Mediterranean. Paul still has great authority within most Christian denominations, and the theological teachings that arise from his letters lie at the foundation of many present-day dogmas and religious practices.

Considering the great influence of these documents, it is remarkable that for several epistles, the authorship is disputed. Certain letters are generally considered to be of the hand of Paul himself; others are widely accepted to be written by someone else. The authorship of some letters is still subject of debate. For an overview of the letters and their status, see table 1 (Den Heyer 1998; Klauck 1998; Ebel 2012; Theissen 2017). In this thesis, I will use computational authorship verification methods to asses the probability of Paul being the author of the disputed letters.

Although it is commonly acknowledged that not all letters are authentic, most modern Christian denominations consider all letters to be inspired by the Holy Spirit and thus valid, regardless of who wrote it in a worldly sense. In this thesis, I will disregard any possible divine interventions and assume that the Holy Spirit, whether it inspired the letters or not, has no distinctive style that could interfere with my results. Moreover, the results presented in this thesis do not hold any dogmatic claim, and I will hold far from any questions regarding the theological relevance of the authorship dispute.

Table 1. The Pauline corpus

Truly Pauline	**Disputed**	**Considered pseudephigraphic**
Romans	2 Thessalonians	1 Timothy
1 Corinthians	Ephesians	2 Timothy
2 Corinthians	Colossians	Titus
Galatians		(Hebrews)
Philippians		
1 Thessalonians		
Philemon		

The structure of this thesis is as follows. First, I will discuss authorship attribution and verification techniques. Next, I will describe the life and person of St. Paul. Subsequently, I will discuss the properties of the documents, considering each letter separately. I will not extensively discuss the theological stances Paul's letters express, since they are not relevant to this thesis. Then, I will discuss the specific challenges one faces when trying to apply the abovementioned techniques to the epistles of St. Paul. The used samples, the features and the processing of those features will be discussed in the section on methods. Finally, I will present the results and discuss what they mean for the authorship dispute of the Pauline letters.

Authorship verification and attribution

Authorship attribution is any attempt to determine characteristics of the author of a text. Historically, this is often done by manually comparing a relatively small set of stylistic features of a text that is known to be written by a certain author to those of a text with an unknown author. Nowadays, computers and large, searchable text corpora have made it possible to use great amounts of features (Juola, 2006). The assumption behind authorship attribution is that every person has a distinct way of writing, a 'human stylome' (Van Halteren, Baayen, Tweede, Haverkort & Neijt, 2005). This stylome is a large set of patterns in language use that can be detected in their texts.

Computational authorship attribution techniques rely mainly on the statistical analysis of textual features. Many kinds of statistics have been suggested, such as average sentence length, the distribution of parts of speech and type/token ratios. Another useful characteristic can be the appearance of words or misspellings that are typical for a certain author. When operating on the lexical level, function words are especially useful. Since they describe relations between words, they tend to reflect syntax rather than semantics. This enables the researcher to compare texts across topics. Other interesting syntactic features include the use of certain constructions, for example the relative amount of relative clauses, the use of participles or the relative amount of auxiliary verbs. The distribution of parts of speech over a text can also be useful. A problem with the use of syntactic features is that they heavily rely on the quality of the parser. Automatised taggers are bound to make mistakes, and in most cases, it is not possible to manually tag the whole corpus. Moreover, even manual tagging is not completely reliable: different people tend to make different tagging choices, and the researcher should be sure that they are measuring features of the actual text, rather than the differences between taggers (Juola, 2006).

Another type of feature that has shown to be very powerful are *n*-grams (Juola, 2006). *N*-grams are combinations of (aspects of) lexical items. What makes them powerful, is that they combine syntactic and lexical information. The *n*-grams can consist of various aspects of lexical items, such as token forms, parts of speech, relations, lemmas and combinations of the above. A variation on the *n*-gram is the skip-gram, in which one or more slots are left open, in order to catch the co-appearance of two words that are not standing next to each other (Juola, 2006). It is also possible to analyse a text in terms of sequences of characters. An advantage of character *n*-grams is that they can catch morphological relationships between words. They are not dramatically effected by spelling noise and appear to be quite succesful (Juola, 2006; Stamatatos, 2009).

It is proven useful to analyse the distribution of words in a text (Juola, 2006). This distribution is not random: frequency and certain properties of words, such as length, morphological decomposability and semantics, are related to each other. The Zipf distribution poses that the frequency of a lexical item is related to its rank in the distribution. This distribution is not a law of nature, but a way to describe a universal tendency. Individual texts differ to a bigger or lesser degree from this ideal distribution. The degree to which a text deviates from the Zipf distribution can be used as a measure for textual richness, which can be a useful feature for authorship attribution. Another measure of textual richness is entropy, which measures how much information a 'bit' (in the case of authorship attribution usually a feature) provides (Juola, 1997; Juola, 2006; Rocha et al., 2017).

Semantic features are potentially powerful (Whissel, 2004), but little research has been done due to parsing difficulties. It is possible to use automatically searchable corpora or dictionaries that contain semantic classes or ratings, but these are not available for all languages (Stamatatos, 2009).

The end goal of feature extraction is a set of features and their (relative) presence for each text. There are many different ways in which the differences between feature sets can be measured. For this thesis, the machine learning part was carried out by my thesis supervisor, Hans van Halteren. Therefore, I will not extensively discuss this subject here.

It is intuitive that the reliability of an authorship attribution study stands or falls by the amount of text available for analysis. However, this is not necessarily the case: a large number of

smaller documents appears to be more important than the availability of one large document (Juola, 2006). Possibly, this has something to do with the representativeness of the sample. Especially important is the amount of features: large feature sets perform much better than small feature sets. However, exactly which features are the best depends on the texts that are being analysed (Juola, 2006; Stamatatos, 2009).

Authorship attribution seems to work in several different cases, but there are some major caveats. One major problem is the question what one is measuring. The human stylome may be detectable within genre, but across genres, several problems arise. When comparing a letter to a novel, it is unclear whether differences in style are due to authorship or due to inherent differences in style between letters and novels. It is virtually impossible to effectively separate between genre-related and author-related differences (Juola, 2006). Another problem is the competence of the analyst. Researchers can (consciously or subconsciously) make choices that bias the result, especially in selecting the features. It is often the minor details that make a difference. Therefore, it is important to carefully document all steps in the process.

Background of the data

St. Paul

There are two sources on the life and person of Paul: the letters that are commonly acknowledged to be written by Paul himself, and the book of Acts of the Apostles. The book of Acts, which is part of the New Testament Canon, is generally considered to be written by the evangelist Luke several decades after Paul's death. Since it is unlikely that he knew Paul personally, we should consider the letters as our primary source. Interestingly, the book of Acts does not mention that Paul wrote any letters. It is important to keep in mind that both the letters and the book of Acts are not neutral sources: both authors use the information they give for their own purposes and argumentations (Ebel, 2012).

The purpose of this thesis is not to give a profound image of Paul or his theology; therefore, some general remarks on his life will suffice. Paul, born Saul, was probably born around 15 CE in Tarsus, a hellenistic city in the east of Asia Minor (now Turkey). He was a diaspora Jew from the tribe of Benjamin. The author of Acts claims Paul was a Roman citizen, but this cannot be established with certainty (Den Heyer, 1998; Ebel, 2012).

Paul was educated in Jerusalem by Gamaliel, a famous scholar and thinker of that time. If this is true, he must have had a profound knowledge of Jewish thinking and the Tenakh. This undertaking would have cost a lot of money, suggesting that Paul came from a well-off family. From his letters, it is clear that Paul had knowledge of the style of Greek orators, which would mean that he had had at least some higher education in Tarsus or elsewhere in the hellenistic world (Johnson, 1987).

Being a man of the world, Paul spoke several languages. His mother tongue was probably Hebrew or Aramaic, but it might have been Greek. Surely, Paul must have had an excellent, native or near-native, command of Greek. In Tarsus, he certainly received his education in Greek; this is also the language in which his letters are written. If Paul truly was a Roman citizen, he also must have had at least some basic knowledge of Latin.

After his education, Paul became a pharisee. He actively persecuted Christians, as is emphasised on several occasions in both Acts and the epistles (e.g. Gal. 1:13; Den Heyer, 1998). At some point in his life, he experienced a revelation from Jesus Christ, after which he radically converted. Although he had never met Jesus when he was still alive, Paul claimed that he had revealed himself to him and that Paul's knowledge and theological ideas came directly from Jesus (Johnson, 1987; Den Heyer, 1998; Ebel, 2012).

After his conversion, Saul changed his name to Paul and started his mission to spread the newly emerging religion. He travelled all across the Mediterranean parts of the Roman empire, visiting Christian communities and preaching in public. His mission was primarily aimed at the

Gentils, non-Jews. His goal was not just to convert people: he also wanted to establish a proper religion. In early Christianity, there was no united church: Christians organised themselves in small sects, with enormous variety in theology, lifestyle and organisation. In Paul's letters, which were written during his mission, he tries to establish unity. He wrote to Christian communities he had visited before, explaining theological, but also organisational and ecclesiastical matters.

Paul died in Rome around 65 CE. Acts does not give any details surrounding his death, but according to tradition, he was beheaded in Rome (Johnson, 1987).

Koine Greek

Like all writings of the New Testament, the Letters of St Paul are written in Koine Greek (*κοινή* meaning 'common'). This variety of Greek was spoken as a lingua franca all across the Near East. It emerged because of the expansion of the hellenistic culture. It can be seen as a somewhat simplified version of Attic Greek, the literary language of Athens that was seen as the 'purest' version of Classical Greek. Koine Greek contains elements of several Greek dialects (Bieringer, 1998). In many aspects, it can be seen as an intermediary stage between Classical and Modern Greek (Kirk, 2012). Joosten (2013) calls New Testament Greek "Hellenistic Greek tainted by Semitic influences" (Joosten, 2013, p. 37).

Until the end of the 19th century, scholars could not establish in which variety of Greek the New Testament was written. Some viewed everything in the New Testament as 'pure' Attic Greek, because of the status connected to that dialect. Others acknowledged the deviations, but attributed them to the influence of Hebrew. It was only at the end of the 19th century that it became commonly accepted that the NT was written in Koine Greek (Kirk, 2012).

The style used in all of the NT, including the letters of Paul, is much poorer than the style attested in other literary works around that time. The main reason for this is probably that the authors had not had the education to produce literary masterpieces. The newtestamentical writings are written after the best of their abilities. Moreover, the Septuagint may have played a role: much of the religious language is directly borrowed from the Septuagint. Some authors even seem to mimick its syntax (Joosten, 2013). The Greek used in the Letters of Paul is not characterised by a high style, but it is fluent, and there is no reason to doubt that Paul knew Greek on a native or near-native level. There are no apparent influences of a Semitic mother tongue interfering with the Greek, but Paul does use certain expressions from the Septuagint (Joosten, 2013).

Greek (including Koine Greek) is an inflectional language with fusional nominal and verbal morphology, meaning that information such as case, gender, and number are fused in one morpheme. Verbs are marked for tense, aspect, voice, and mood, and agree in person and number with the clause subject. The four 'main' cases are nominative, genitive, dative and accusative; some nouns have a distinct form for the vocative. There are three genders, namely masculine, feminine, and neuter, and two numbers, namely singular and plural (Kirk, 2012).

The word order in Koine Greek is quite free: all word orders are allowed, since syntactic relations are expressed by case and verbal agreement. In the NT, the predominant word orders are SVO and VSO. It is not obligatory to express S, V, and O, and often, one or more of these is omitted (Kirk, 2012).

Letters in antiquity

Antique letters usually contain a number of standardised components. First, there was the letter opening, consisting of a prescript and the letter proem. The prescript consisted of three parts: the superscription, stating the sender's name in the nominative case; the adscription, stating the addressee's name in the dative; and the salutation, a greeting in the infinitive (Klauck, 1998). The proem was the transition between the prescript and the letter body, which contained the main message. It could contain highly stereotypical phrases, but also freely formulated health wishes,

thanksgivings and prayers. Because of this free formulation, it is often hard to draw an exact line between the proem and the body opening.

The body starts with a body opening, containing more formulas that usually express joy or gratefulness. The core of the body contains the main message. Sometimes there is a body closing that expresses for example a request. This request (for example the request to send a letter back) can also appear elsewhere in the body.

The last part of the letter is the letter closing, consisting of a highly formulaic greeting. This greeting can be directed towards the reader or a third person (2 Tim. 4:19: "Greet Prisca and Aquila, and the household of Onesiphorus"), and it can come from the sender or from someone else (1 Cor. 16:19: "Aquila and Prisca, together with the churches in their house, send you hearty greetings in the Lord"). Sometimes, the letter ends with the date. It was uncommon to add the name of the sender at the end (Klauck, 1998).

In the times of the Roman Empire, it was extremely common to dictate the letter to a scribe, rather than writing it in the own hand. This is not necessarily because the author was unable to read and/or write: especially the upperclass was very well-educated. The degree of influence the scribe had on the final version of the letter varied greatly. Sometimes, the scribe recorded the letter verbatim. In this case, the input of the scribe was minimal. In other cases, the author dictated the letter, while the scribe took extensive notes. It was also possible that the author wrote an extensive draft, which the scribe had to make into a proper letter. In these cases, the scribe can be considered the editor of the letter. Another possibility was that the scribe played the role of a co-author, for example when the author only provided a very minimal draft. It also happened that the scribe wrote the whole letter more or less independently. This was possible because of the great degree to which ancient letters consisted of stereotypical, fixed formulas (Richards, 1991, p. 49).

The Pauline letters

Paul, who had had a good education in the heavily hellenicised city of Tarsus, was very aware of the antique conventions of letter writing. His letters follow the conventional antique letter structure, but Paul also added some new elements. First, there are some subtle Jewish influences. For example, his prescripts often consist of two parts, as is common in Jewish (but not in Greek) letters. He also made the private letter into a community letter that was meant to serve as a guideline for the whole Church. The letters were not just meant as friendship letters: they were also public texts of worship. This might have been inspired by the Jewish letter culture as well (Theissen, 2007, pp. 61-73). Community letters were not just meant for the addressee: they were supposed to be read out loud to the whole community (cf. Col. 2:1; Stuckenbruck, 2003).

Although all Pauline letters were written in the first century CE, the oldest copies we have today are no younger than the beginning of the third century CE (Hurtado, 2006, p. 38). This means that we cannot be sure that the text we have today is exactly the text Paul wrote. Possibly, copyists have made mistakes while copying the original letters. It is even thinkable that someone has edited the text at a very early stage, and it would be hard or even impossible to determine to what extent the original text has been altered. However, we do know that the Pauline epistles were seen as authoritative already in the second century CE. Therefore, it would be safe to assume that copyists were very careful in copying the text, minimising the risk of mistakes (Hurtado, 2006, p. 39).

Precisely because of this authority, using Paul's name when writing a letter could give the letter a certain status. Pseudepigraphy, writing a text under the name of someone else, was a very common practice throughout antiquity. For us, readers from the 21st century, challenging the authorship of a letter sounds like a charge of fraud. In modern Western society, it is unacceptable to claim to be someone you are not and write to others on their behalf. However, in the Ancient world, pseudepography was completely accepted, as long as the writing was in the spirit of and in line with the views of the claimed author. In a messenger culture such as the one the first Christians lived in, the messenger represents the actual sender of the message. Pseudonimity was the norm, especially in

Jewish literary culture, where almost all religious texts are either anonymous or claimed to be written by figures such as Moses or Ezra (Theissen, 2007, pp. 109-115). These are all figures from the distant (almost legendary) past. At the time when the first pseudepigraphic letters in Paul's name would have been written, Paul's corpse was still warm (Stuckenbruck, 2003). However, quotes like the one in 2 Thess 2:2 show that, at an early stage, fake Pauline letters were in circulation.

Theissen (2007, pp. 105-115) notes that, in early Christianity, there was a whole 'pseudepigraphic phase'. During this phase, the authority of Paul was fully established, and claiming to be him gave weight to any dogmatic claim. Moreover, it is possible that already during Paul's life, he used other people to deliver his messages. Especially when he was in prison, it is possible that Paul's coworkers and disciples preached in his name. It is reasonable that they would continue doing so after he had died. While doing this, they often tried to mimick Paul's style and closely followed the structure of his letters. This makes pseudepigraphy often hard to detect. Uneducated, lower-class Christians thus might not have realised that many Christian writings were pseudepigraphic, but educated people surely knew that this was common (Theissen, 2007, pp. 115).

Even the genuine Pauline letters are not written by Paul himself in the modern sense of the word. It is certain that Paul used scribes, but it is not exactly clear to what extent this interfered with the style or structure of the letters.

Moreover, we cannot be sure that all parts of a letter were originally written for that specific letter. In the antique world, 'recycling' older material in letters was extremely common. Robson (1917) points out that sudden pivots in style, such as the one in Rom. 19, can indicate that Paul inserted part of a speech or letter he had writtten before. It is also very likely that certain letters, such as 1 Corinthians, actually contain portions of several other letters, that have been incorporated by a third party at a later stage (Klauck, 1998, pp. 305-308).

Genuine Pauline letters

According to tradition, all letters in the Pauline corpus (including Hebrews) are written by St. Paul. However, the last two centuries have seen a heated debate about the authorship of the letters. At the moment, there is somewhat of a consensus among scholars which letters are actually written by Paul, and which are not. Arguments pro or contra Pauline authorship are mainly based on theology, style, and archaeology (Dunn, 2003b, p. 11). The theological discussion is beyond the scope of this thesis; for an overview of the debate, I refer to the handbook edited by Dunn (2003a).

The Letter to the Romans

The Letter to the Romans is probably the youngest Pauline letter in the New Testament, as it was written around 56-57 CE, less than a decade before Paul's death. It was dictated to the scribe Tertius, who was a professional secretary (Richards, 1991, p. 172; cf. Rom 16:22). Paul had never visited the house churches in Rome he directed the letter to (Klauck, 1998, p. 301).

Not only is this letter the youngest, it is also the longest (7094 tokens in total) and the one with the largest vocabulary (Wischmeyer, 2012, p. 246). Especially the ending is the subject of a heated debate: it is unclear which parts actually compose the letter closing, and for some parts of the ending, the originality is disputed (Wischmeyer, 2012, p. 250). It has been suggested that Romans is the result of the synthesis of three different Pauline letters, but the consensus is still that the letter opening and body of Romans, at least up till 16:20, should be considered one whole letter from the hand of Paul (Klauck, 1998, pp. 301-303). Since for this research, I only used the proem and body of each letter, the debate on the authorship of the last part of Romans is not relevant here.The epistle has a conversational style: Paul often addresses imaginary interlocutors and answers rhetorical questions in order to make his point clear (Holloway, 2003).

The First and Second Letter to the Corinthians

The letters to the Corinthians we have today constitute part of a larger correspondence. In 1 Cor. 5:9, Paul refers to an earlier letter he had written: "*I wrote unto you in an epistle not to company with fornicators"* (MEV). However, the other letter or letters to the Corinthians have been lost to time (Klauck, 1998, p. 306).

The First Letter to the Corinthians is, after Romans, the longest epistle in the New Testament. It was written around 55 CE from Ephesus (Klauck, 1998, p. 308) with help of the scribe Sosthenes (Richards, 1991, p. 172). Because there were earlier letters from Paul to the Corinthians, it has been suggested that the letter is the result of an editing process in which one or more other Pauline letters have been included.

2 Corinthians was written in 56 CE, but was probably originally at least two different letters. One of them (letter A, chapter 1-9) warm and friendly, the other (letter B, chapter 10-13) with a harsher tone (Klauck, 1998, p. 308-310; Murphy O'Connor, 2003). It is written by Paul, who presents Timothy as a co-author (2 Cor 1:1). Some researchers have suggested that 2 Corinthians might consist of several letters as well, but this is not generally accepted (Klauck, 1998, p. 310).

The Letter to the Galatians

The Letter to the Galatians used to be considered the oldest or second oldest letter in the Pauline corpus, but nowadays, it is more often dated between 2 Corinthians and Romans (Klauck, 1998, p. 313). This letter is remarkably confrontational: Paul is arguing against moral and theological opponents who seem to have gotten some kind of foothold in the Galatian churches (Longenecker, 2003). In doing so, he is not afraid to use severe language (Du Toit, 2014).

The Letter to the Philippians

The dating of the Letter to the Philippians is disputed. It was written from prison, but it is unclear whether Paul wrote it during his imprisonment in Ephesus (around 56-57 CE) or from the prison in Rome (around 60 CE). There is some discussion about the literary integrity of this letter. Some scholars claim that the letter consists of two or three different letters (all originally Pauline), but the majority of present-day scholars consider this letter a self-contained whole (Klauck, 1998, p. 318-319). Even though the authenticity of Philippians is not disputed, relatively much of the vocabulary is unique to this letter (Stuckenbruck, 2003).

The First Letter to the Thessalonians

According to most scholars, the First Letter to the Thessalonians is the oldest letter in the Pauline collection. It was probably written around 50-51 CE, but some researchers argue that the dating might be as early as 40-41 CE (Klauck, 1998, p. 356). In the prescript, the senders identify themselves as Timothy, Silvanus and Paul, but in the letter, it seems to be only Paul who speaks (cf. 2 Thess 2:18). Throughout the letter, the *logos* (the word) is emphasised. The letter has a friendly tone (Mitchell, 2003).

The Letter to Philemon

The letter to Philemon is the shortest Pauline letter (328 words) and is addressed to Philemon and the church in his house. Similarly to the Letter to the Philippians, it was written from prison, but it is not exactly clear from which prison. Because of the length of this letter, I have not included it in my analysis for this research.

Disputed letters

The fact that the disputed letters differ from the authentic Pauline letters was noticed already in the nineteenth century (Polhill, 1973; Klauck, 1998). Many scholars claim that they were written by

followers of Paul, who knew Paul and his teachings well. This has led to the name Deutero-Pauline letters.

The Letter to the Ephesians and the Letter to the Colossians

Obviously, the dating of the letters to the Colossians and the Ephesians depends on whether one believes they are written by Paul or not. If Colossians is of Paul's hand, it was written from prison, so it should be dated around 55 or 60 CE (Stuckenbruck, 2003). Klauck (1998) states that Colossians is the oldest Deutero-Pauline letter; according to this theory, it was written around 70 CE by a student of Paul. The author self-identifies as Paul, together with Timothy (Col 1:1). The letter was dictated to a scribe (Col 4:18).

Ephesians, then, would be written around 80 or 90 CE (Klauck, 1998, p. 316). The letter does not tell us much about how, where, when and even to whom it was written: the claim that it was directed to the Ephesians lacks in the oldest manuscripts (Lincoln, 2003).

Ephesians and Colossians are similar to each other in theology, structure and style, and there is a general consensus that they are dependent on each other. However, there is some disagreement as to which letter came first. Already in 1838, Mayerhoff (according to Polhill, 1973) posed that Colossians was written by someone else than Paul, who drew from the authentic epistle to the Ephesians. Many present-day scholars reckon that it is Ephesians that is dependent on Colossians (Polhill, 1973; Klauck, 1998, p. 322). Arguments for both sides are mostly theological in nature; therefore, I will not discuss them here.

These letters differ from the other letters in the Pauline corpus in theology and style. They display a more elaborate style, with longer sentences, more relative and participial clauses, and more genitival constructions. They are also different in wording: Polhill (1973) noted that there are several words and combinations of words that are not found in any other Pauline letters. These differences have led many modern scholars to believe that Ephesians and Colossians are not written by Paul himself (Anderson, 1996). Some believe that these differences can be attributed either to the use of a scribe or the involvement of a co-author (Richards, 1991) or that the letter was written by someone close to Paul during Paul's lifetime (Stuckenbruck, 2003).

The Second Letter to the Thessalonians

Out of all disputed Pauline epistles, the Second Letter to the Thessalonians is the one of which most scholars still believe it is truly written by Paul himself (Klauck, 1998). In structure, it closely mimics 1 Thessalonians, suggesting that the author must have been familiar with it (Mitchell, 2003).

Presumably non-Pauline letters

The Pastorals

The 'pastoral letters' (1 and 2 Timothy and Titus) are dated even later than the deutero-Pauline letters (given that those are, indeed, deutero-Pauline), which is why they are sometimes referred to as 'trito-Pauline'. The addressees are Paul's closest friends or co-workers, but probably both sender and recipients are faked. The true author presumably wanted to provide an image of the correspondence between 'perfect Christians' (Klauck, 1998).

There are several reasons why the Pastorals are generally not attributed to Paul. First of all, they were not included in the oldest collections of Pauline letters. Another, in this context more relevant reason is the difference in vocabulary between the Pastorals and the Pauline letters. The Pastorals are characterised by the extensive use of proper names (both personal names and proper names). As much as 36% of the words used in the Pastorals do not appear elsewhere in the Pauline corpus. More than a third of those are used in Christian writings from the second century. The opposite is also true: many typically Pauline expressions do not appear in the Pastorals. For example, 'with' can be translated into Greek as either μετά or σύν. In the undisputed letters, Paul uses both

almost equally frequently (28 times σύν, 37 times μετά); in the Pastorals, only μετά is used. Certain key theological concepts are also missing, such as the idea of living 'in Christ'. Finally, it is hard to fit the writing of the Pastorals into Paul's life story as we know it (Hultgren, 2003).

Richards (1991) argues that the Pastorals are written by Paul in the ancient sense of the word. According to this theory, the differences are due to the greater artistic freedom of the scribe. This theory, however, is far from accepted (Hultgren, 2003).

Hebrews

The Letter to the Hebrews differs from the Pauline letters in many aspects, and does not even claim to be written by him. When the New Testament canon was being established, there was a heated debate about whether this (quite unpopular) letter should even be included. A passage mentioning Timothy was the final reason to include this letter; however, this passage might have been included later. Nowadays, virtually all scholars agree that Paul is not the author of the letter to the Hebrews (Den Heyer, 1998). The Greek in this letter is of higher quality than the Greek used in the Pauline letters and the author uses many figures of speech (Klauck, 1998, p. 335).

The Catholic Letters: James and 1 Peter

The Catholic Letters are all letters in the New Testament canon that are not traditionally attributed to St. Paul. Due to the short size of many of these letters and the fact that they are not all available in the data set, I only included the Letter of James and the First Letter of Peter in this research.

The Letter of James claims to be written by James, the brother of Jesus, which would mean that it was written around 60 CE. However, most scholars agree that it is younger (90-100 CE), implying that it cannot have been Jesus' brother who wrote it. In style, it is similar to Hebrews: the author belongs to the class of the teachers (Jas 3:1), and this is reflected in an extensive use of figures of speech and metaphors (Klauck, 1998, p. 339).

The First Letter of Peter is dated around 80-90 CE by followers of Simon Peter. The style reminds those of Hebrews and James, meaning that the author must have a high (probably native) command of Greek (Klauck, 1998, p. 340; Joosten, 2013, p. 43).

Specific problems and challenges for this case

When trying to apply authorship verification techniques to the Pauline letter collection, several problems arise. First and foremost, the reliability of the data. As mentioned above, our oldest sources date from the second century CE, which is more than a century after Paul's death. This makes some features, such as punctuation, useless, since this was a later addition. It also makes it likely that character-grams, which are influenced by spelling, will not be of much use.

Another problem is that the authors who wrote in Paul's name might have actively tried to imitate his style. Therefore, it is important to pick features that are hard to imitate, such as the distribution of frequencies over words.

Moreover, the letters have been edited to some extent over the course of history. When comparing different manuscripts, we can see subtle (or sometimes not so subtle) differences between the versions. By comparing these versions, scholars have tried to establish for each difference which version is most likely to reflect the original. Unfortunately, the 'final version' remains hypothetical. Furthermore, some letters (for example 1 Corinthians) are considered to be the result of a process in which parts of other Pauline letters were incorporated as well. Possibly, the text has been edited in order to make the separate parts fit better into the whole. Some editing might also have been done by the scribes to whom Paul dictated his letters. This is not necessarily fatal (Holmes, 2003), but it can considerably obscure the data.

Stamatatos (2009) emphasises that the choice for certain features should depend on the language the texts are written in. There are several studies on text classification in Modern Greek

(Tambouratzis, Markantonatou, Hairetakis, Vassilou, Carayannis & Tambouratzis, 2004; Mikros & Carayannis, 2000), but they are only of limited use. Most of these studies rely on features related to the diglossia situation in present-day Greece, where two varieties of Greek (*katharevousa* and *dimotiki*) exist alongside each other. The New Testament is written entirely in Koine Greek, which is comparable to the *dimotiki* variety of Modern Greek. However, it is likely that some features of Attic Greek (on which *katharevousa* is based) might be reflected in the Koine of the NT; therefore, it will still be useful to take into consideration features such as morphology of nouns and adverbials (Tambouratzis et al., 2004). On the other hand, it is not unlikely that most or all morphological anomalies have been lost somewhere in the editing process.

Whissel (2004) has attempted to classify the disputed Pauline letters using measures of emotional space ('Pleasantness' and 'Activation') and mental representation. Ratings were taken from a Dictionary of Affect in Language (Whissel, Fournier, Pelland, Weir & Makarec, 1986). In addition, she used statistics such as word length and sentence length and richness features such as repetitiveness and the distribution of parts of speech. The samples were quite small (around 800 words) and sample sizes differed. Her system classified 2 Thessalonians and two samples from Hebrews as undisputedly Pauline. The features based on emotional and mental ratings performed best. This is not surprising, given the fact that Whissel (2004) used English translations, and not the original Greek text, as her input. This renders statistics such as word length and sentence length useless. Vocabulary richness features become unreliable as well: some Greek words have several possible English translations, and vice versa. Therefore, what is measured is often the stylistic choices of the translator, rather than the authentic stylome of Paul. These features are even more unreliable because of the different sizes of the samples. She also included statistics of punctuation, even though punctuation marks are sparse in the oldest Greek manuscripts (Hurtado, 2006, p. 177).

Research question

The topic of authorship in the epistles of St. Paul has thus been the subject of a great deal of research throughout the centuries. Most of the research has focused on differences in manually identified stylistic deviations, or differences in theology. Whissel (2004) has used computational methods to determine the authorship of the disputed letters, but, as mentioned above, her research has some serious flaws. In this thesis, I will try a computational linguistics approach to the problem, trying to answer the following questions:

- Are modern authorship recognition methods able to succesfully classify the epistles in the Pauline corpus by author?
- And if so, which epistles are classified as genuinely Pauline, and which are not?
- Which (kinds of) features are most distinctive?

I expect that the answer to the first question will be yes. Authorship attribution techniques have proven to be useful in similar situations. Elliot and Valenza (1996) used computational methods to classify disputed claimants that were allegedly written by Shakespeare. Their tests quite succesfully distinguished between Shakespeare and several of his contemporaries. Other studies (Kestermont, 2012; Van Halteren & Rem, 2013) have applied authorship attribution techniques to medieval texts, with promising results. However, I do not expect the results of this research to be completely conclusive, since the data are very noisy. Therefore, the authorship question cannot be conclusively solved with computers alone: human mediation will be necessary.

As for the second question, I expect that 1 and 2 Timothy, James, 1 Peter and Hebrews will be classified as definitely not Pauline. 2 Thessalonians is often considered the most Paul-like of all disputed epistles (Mitchell, 2003); therefore, I expect it to be similar to the Pauline epistles. Ephesians and Colossians might not resemble the Pauline epistles, especially in vocabulary and style. Philippians, a genuinely Pauline epistle, might not resemble the other genuinely Pauline letters vocabulary-wise, since it is known to contain a remarkable amount of words and phrases that are not

seen anywhere else in the Pauline corpus (Hultgren, 2003). In syntax, it should not deviate significantly from the other Pauline epistles. The distance between letters with different co-authors or scribes can also be expected to be relatively big. If this hypothesis is borne out, Romans, which was written by Paul through the help of the scribe Tertius, will differ from 1 Corinthians, which was written through the help of the scribe Sosthenes, and 2 Corinthians, that has Timothy as its co-author.

Choosing the right features is extra hard in cases like these, because one needs to find a balance between very subtle features that are hard to mimick and very 'crude' features that are less likely to be affected by the influence of an editor or scribe. I expect that especially richness measures can be useful. Juola (2007) reports that differences in entropy are particularly informative. Moreover, they are also universal. This is relevant to this research: Stamatatos (2009) noted that authorship attribution techniques tend to perform better on English texts. Another reason why I expect richness measures to perform well, is because I suspect these are relatively unaffected by later editing. Especially in a language with a free word order, such as Greek, a scribe or editor is likely to (consciously or not) make small changes to word order. It is also possible for a scribe or editor to change verbal voice, without adjusting the lexical choices of the author. Moreover, statistics such as entropy and deviation from the Zipf-distribution are hard to mimick. This is especially relevant for letters such as 2 Thessalonians, where the author seems to have attempted to make his style similar to Paul's.

I also expect that features based on word order will be more useful than those based on syntactic relations. Word order in Koine Greek is free, meaning that differences in word order reflect the personal choices of the author. On the other hand, as mentioned above, it is also possible that word order is one of the first things to be affected by a scribe or editor.

Vocabulary-based features might also perform well. Many of the authorship claims are (partially) based on the absence or presence of certain central Pauline concepts in the disputed letters. It is likely that this will be reflected in features that reflect the use of certain forms or lemmas. Tambouratzis et al. (2004), who conducted an authorship attribution study in Modern Greek, found lemma frequencies useful when determining authorship. However, it is also possible that these features will be highly skewed because of the relatively small size of the samples.

Stamatatos, Fakotakis, and Kokkinakis (2001) performed an authorship attribution study in Modern Greek that excluded all lexical features and focused purely on stylistic markers, such as punctuation, syntactic relations and parts of speech. They found this method to be quite successful, but the most successful analyses combined lexical with non-lexical information. Therefore, I expect features that combine syntactic and lexical information, such as *n*-grams that contain both a part of speech and a lemma, to be very powerful.

Method

Data

For this research, we used the tagged Greek New Testament from the PROIEL Treebank family (Haug & Jøhndal, 2008). The PROIEL (*Pragmatic Resources in Old Indo-European Languages*) is a project that builds treebanks for early attestations of Indo-European languages. The treebank is freely available on the internet and includes historical texts in languages like Classical Armenian, Gothic and Church Slavonic. Their treebank of the Greek New Testament still lacks some parts of the Catholic Letters, but they had a complete treebank of the Pauline letter corpus. For the Greek text, the version of Tischendorf (1869) was used. For our goals, this is the largest drawback of this data set: on several occasions, present-day scholars make slightly different choices when reconstructing the most authentic version of the Greek text (Black & Davidson, 1981, p. 34). Moreover, this treebank is not complete yet. The Pauline corpus is completely tagged, but the work on the Catholic Letters is still ongoing.

All texts are tagged manually, but with computer support. This method has shown to be the most accurate and efficient (Eckhoff et al., 2018). After being tagged, each sentence was reviewed by a second annotator to maximise accuracy and consistency.

Every token line in the treebank contains the following information:

- a numeric id that identifies the token;
- the form of the token. Forms contain diacritics, but are stripped of punctuation;
- the letter, chapter and verse;
- the lemma. Following common Greek conventions, this is the first person singular present indicative active form of the verb. For nouns, the nominative singular is used. Homonymous lemmas are made distinct by adding #1, #2 etc.;
- the part of speech;
- morphological features, including information on person, number, tense, mood, voice, gender, case and the presence of inflection;
- if present, the id of the head of the token;
- if present, the relation to the head of the token;
- the character following the token. This might be a white space, but also a punctuation mark. Because punctuation was largely absent in the original texts (Hurtado, 2006, p. 177), I have not used this field.

Apart from the overt tokens, the treebank also included empty verb and conjunction nodes. They are used to model empty categories such as ellipsis and gaps. Secondary dependencies were used to reflect structure sharing in control structures with non-finite verbs. In my features, I have only looked at primary dependencies.

In the treebank, the text is divided into sentences and word tokens. Especially the division into sentences is sometimes artificial. However, since the taggers collaborated closely in order to establish a fitting protocol for each language, the divisions are consistent. Therefore, I decided to include sentence length in my features; however, this feature should be approached with some care.

Samples

For each letter, we extracted one or more cases: subsets of text, consisting of a minimum of 700 and a maximum of 1500 tokens. Each case consisted of sentences that were drawn from the proem and body of the text as determined in Klauck (1998); see table 2. The sentences were drawn from the original data set, such that no sentence appeared twice in any case. When the threshold of 1500 tokens was reached, the sentence was cut off. For the longer letters, with more than 1500 tokens in the letter proem and body, sentences were drawn randomly (so not necessarily in consecutive order); for shorter letters, the letter proem+body was taken as a whole, without making any changes to the order of the sentences. Because the sentences were drawn randomly, *n*-grams and character-grams were calculated per sentence, so no *n*-grams cross sentence boundaries.

As mentioned above, some Pauline epistles as we have them today are thought to incorporate one or more other Pauline letters. It is unlikely that one sentence would contain parts of different origins. Therefore, taking random sentences minimises the influence of literary integrity, as long as all incorporated units are originally Pauline.

Table 2. Overview of samples

Letter	Total tokens (in PROIEL)	Proem + body	Tokens body	Interpolations	Samples
Romans	7371	1:8 - 16:20	6887		4 x 1500

1 Corinthians	6828	1:4 - 16:12	6612		4 x 1500
2 Corinthians	4474	1:3 - 13:10	4248	6:14 - 7:1	2 x 1500
Galatians	2227	1:6 - 6:10	2027		1 x 1500
Philippians	1626	1:3 - 4:9	1370		1 x 1370
1 Thessalonians	1476	1:2 - 5:22	1385		1 x 1385
Philemon	330	1:4 - 1:20	228		none
Ephesians	2413	1:3 - 6:20	2306		1 x 1500
Colossians	1577	1:3 - 4:6	1327		1 x 1327
2 Thessalonians	822	1:3 - 3:13	718		1 x 718
Hebrews	4574	1:5 - 13:17	4502		2 x 1500
1 Timothy	1589	1:3 – 6:19	1557		1 x 1500
2 Timothy	1237	1:3 – 4:8	1208		1 x 1208
Titus	658	1:5 – 3:11	523		none
1 Peter	859	1:3 – 5:9	832		1 x 832
James	1742	1:2 – 5:6	1466		1 x 1466

In this thesis, when referring to a sample, I will use the name of the letter if there is only one sample taken from it; if there are multiple samples taken from one letter, I will refer to them with numbers after the name of the letter, eg. Romans 4, 1 Corinthians 2 etc.

Features

For this study, I extracted features from the data using the Perl code in appendix A. In this section, I will give a full list of all kinds of features that I used. The part in italics is variable (i e '*pos*' means that the field is reserved for a certain part of speech). All feature values that were not character-grams and that did not start with *RATIO* were divided by the total number of tokens in the text. Ratios were only included as a feature if they had 5 or more occurrences in the text sample. Whissel (2004) quite succesfully used measures of emotional space to classify the Pauline epistles. However, she analysed the English translation of the texts; therefore, she could draw from an existing databank with ratings for English words. As far as I am aware, no such thing exists for first-century Koine Greek, which is why I was unable to include these statistics.

Hirst and Feiguina (2007) found that bigrams from a stream of syntactic labels were useful to classify very short texts (about 200 words long). In the feature set, I included bi- and trigrams of syntactic categories (part of speech and relation to the head) in the order they appeared in the text. I also added bi- and trigrams of syntactic categories in the order they appear in their syntactic tree structure.

Koppel, Akiva, and Dagan (2006) proposed feature instability as a criterion for feature selection. The stability of a word or phrase can be defined as the availability of synonyms for it. In the feature set used for this thesis, I manually defined a set of unstable words, based on the word list in Bieringer (1998). For each meaning, I calculated the relative use per synonym and included it as a feature.

Vocabulary features, syntactic features and *n*-grams

- RATIO_AFTER_HEAD_TYPE_ *lemma* and

- RATIO_AFTER_HEAD_POS_*pos*: ratio that reflects how often a certain lemma or part of speech comes after its head.
- RATIO_POS_HEADFINAL_*pos*: ratio that reflects how often the head of a certain part of speech is in head-final position.
- RATIO_ADJ_ARTICLE_*adjective*: ratio that reflects how often a certain adjective appears in a determiner phrase with an article.
- RATIO_ADJ_BEFORE_ART_*adjective*: ratio that reflects how often a certain adjective precedes the article of the DP in which it is embedded.
- RATIO_PREP_WITH_ART_*preposition*: ratio that reflects how often the complement of a preposition is a DP with an article.
- RATIO_NOUN_FORM_ART_*form (of a noun)* and RATIO_NOUN_LEMMA_ART_*lemma (of a noun)*: ratio that reflects how often a certain noun (form or lemma) is embedded in a DP with an article
- RATIO_VERB_RELAT_*relation*: relation that reflects how often a certain relation to the head is seen if the head is a verb.
- RATIO_VERB_LEMMA_TRANS_*lemma (of a verb)* ratio that reflects how often a certain verb (lemma) is transitive
- RATIO_VERB_FORM_TRANS_*form (of a verb)*: ratio that reflects how often a certain verb (form) is transitive
- RATIO_VERBFORM_AS_AUX_*form (of a verb)* and RATIO_VERBLEMMA_AS_AUX_*lemma (of a verb)*: ratio that reflects how often a certain verb (form or lemma) is used as an auxiliary verb.
- RATIO_VERBFORM_WITH_AUX_*form (of a verb*) and RATIO_VERBLEMMA_WITH_AUX_*lemma (of a verb*): ratio that reflects how often a certain verb (form or lemma) appears with an auxiliary as its head.
- RATIO_ADJPERNOUN_*lemma (of a noun)*: the average amount of adjectives for a given noun (lemma).
- RATIO_NOUNS_ADJECTIVES: the average of adjectives per noun
- RATIO_CASEPERNOUN_*lemma (of a noun)_case*: ratio that reflects how often a certain noun appears in a certain case.
- RATIO_CASEPERNOUN_EXCLPP_*lemma (of a noun)_case*: ratio that reflects how often a certain noun appears in a certain case, excluding occurrences after a preposition.
- RATIO_NOUNCASE_*case*: ratio that reflects how often a certain case is used on nouns and pronouns
- RATIO_SING_NOUN_*noun*: ratio that reflects how often a certain noun is used in the singular number
- RATIO_UNSTABLE_*meaning_lemma*: feature for the unstable features: how often, when a word has a meaning A, lemma X is used? For a list of all unstable words I included, see appendix B
- RATIO_COORD_POS_POSITION_*pos_pos_*… and RATIO_COORD_LEM_POSITION_*lemma_lemma_*...: ratio that reflects how often certain parts of speech or lemmas are used in a coordination with *καὶ* (‘and’), in the order they appear in the original text.
- RATIO_COORD_POS_ALPHA_*pos_pos_…* and RATIO_COORD_LEM_ALPHA_*lemma_lemma_*...: ratio that reflects how often certain parts of speech or lemmas are used in a coordination with *καὶ* (‘and’), in alphabetical order.
- C*[size]_characters:* these features contain the character-grams. I have made character-grams up to five places long. The “&” sign is a white space. The characters are not stripped of diacritics, so ‘óv’ and ‘ov’ count as two separate character-grams. The total amount of appearances for each character-gram is divided by the total amount of characters in the text.

- WORDLENGTH_*length*_CHAR: every wordlength counts as one feature.
- TREE3_[*kinds of elements*]_*element_element_element_element* and TREE2_*[kinds of elements]_]element_element*: the TREE-features contain chunks of syntax trees. Each element can be either a form (marked by a W in the 'kinds of elements' section of the feature), a lemma (L), a part of speech (P), a relation to the head (G) or a SKIP (S). The trees are built by starting from the last branches and climbing up, adding heads to the structure. From each tree, all bi- and trigrams are taken and stored as features. Obviously, there is a considerable amount of overlap within these features. However, modern machine learning techniques should be able to deal with overlap.
- T*[size]_[kinds of elements]_token_token_…*: the T-features contain tri-, bi- and unigrams of the tokens in the text, in order of appearance. Token *n*-grams are built for each sentence separately, so no token-grams cross sentence boundaries. Similarly to the tree *n*-grams, the token *n*-grams contain all possible combinations of forms (W), lemmas (L), parts of speech (P), relations to their head (G) and SKIPS (S).
- VERB_: the *VERB*-features combine, for each verb, one, two or three elements from the list below. The *lemma* element was never used separately, since the presence of a certain lemma is already incorporated in the token unigrams. Each element was only present as far as it was relevant; for example, the *case* element was only present in participles, and the *person* element only in finite verbs. The elements included in this feature are:
    - L_*lemma*
    - P_*person and number*
    - T_*tense*
    - M_*mood*
    - V_*voice*
    - C_*case*
- VERB_DEP_POS_*lemma (of a verb)_POS*: feature that counts which parts of speech the dependencies of a certain verb are.
- VERBMOOD_DEP_POS_*mood_POSdependency* and VERBTENSE_DEP_POS_*tense_POSdependency*: these features count the parts of speech that appear as the complement of any verb in a certain mood or tense.
- VERBTENSE_AS_AUX_*tense* and VERBMOOD_AS_AUX_*mood* and VERBVOICE_AS_AUX_*voice:* counting for each auxiliary verb the tense, mood, and voice.
- VERBLEMMA_AS_AUX_*lemma* and VERBFORM_AS_AUX_*form*: counting for each auxiliary verb the lemma and form
- VERBTENSE_WITH_AUX_*tense* and VERBMOOD_WITH_AUX_*mood* and VERBVOICE_WITH_AUX_*voice*: these features count the tense, mood, and voice for each verb that has an auxiliary verb as its head
- AUX_OBJ_POS_*pos*: for every main verb that has an auxiliary verb as its head, this feature counts the part of speech of its complement(-s).
- PP_NUMBER_*preposition_number*: this feature counts the number of the DP that is the complement of a certain preposition
- PP_CASE_*preposition_case*: this feature counts the case of the DP that is the complement of a certain preposition. In Greek, the meaning of many prepositions is partially defined by the case of the complement. Therefore, the lemma of one preposition can often be divided into several meanings. This is not reflected elsewhere in the features, since in the token- and tree *n*-grams, case is not counted as a separate element.
- PP_NUMBER_CASE_*preposition_case_number*: feature that for each case and number counts how often a certain preposition is used with it.
- PP_HEADDEP_LEMMA_*preposition_lemma*: counts the lemma that is the head of the DP that complements a given preposition

- ADJ_VERBHEAD_HEAD_*lemma (adjective)_lemma (verb)*: how often is adjective X the head of verb Y?
- ADJ_WITHVERBHEAD_*lemma*: how often does a given adjective appear with a verb as its head?
- VERB_ADJDEP_*lemma (verb)*: counts how often a certain verb is used with an (any) adjective
- PRONOUN_PERSON_*person*: counts how often a pronoun in the first, second or third person is used
- PRONOUN_NUMBER_*number*: counts how often a pronoun in the singular or plural is used
- PRONOUN_CASE_*case*: counts how often a (any) pronoun in a certain case is used
- REWRITE_HEAD_DEPS_LOCATION_*rewrite* and REWRITE_HEAD_DEPS_ALPHA_*rewrite*: these features contain the rewrites. A rewrite consists of the part of speech of a head, followed by the relations of all its dependencies to the head. The LOCATION features contain the rewrites with the dependencies in the order they originally appeared; the ALPHA rewrites contain the rewrites with the dependencies in alphabetical order.

Richness features

Features of vocabulary richness tell us something about the distribution of words and frequencies over a text. I calculated the following ratios:

- TTR (Type/token ratio): one of the most straight-forward measures of vocabulary richness. It is obtained by dividing the number of types by the total number of tokens.
- PHPX (Hapax probability): the probability that a certain lemma, form, rewrite etc. is a hapax. A hapax is an element that occurs only once.
- NONZIPF (deviation from Zipf distribution). Zipf's formula describes the relationship between the rank of a word and its frequency. The NONZIPF features give the average deviation from this distribution. Since all individual deviations are standardised, this value does not give us any information on how exactly the distribution of frequencies over ranks deviates from an ideal Zipf distribution.
- ENTROPY (entropy). In quantitative linguistics, entropy is used to measure randomness of a text. Although its quality as a measure of vocabulary richness has been disputed, it has been proven useful in authorship attribution studies (Juola, 1997; Grabchak, Zhang & Zhang, 2013; Rocha et al., 2017). The entropy of each element was calculated using the following formula:

  (frequency / total number of elements) * log(frequency / total number of elements) / log(2)

These ratios were calculated for the following elements:

- Lemmas;
- Forms;
- Rewrites with the dependencies in order of appearance in the original text;
- Rewrites with the dependencies in alphabetical order;
- Sentence length;
- Word length.

For sentence length and word length, only entropy and nonzipf were included. These features were included in the total feature set, but they were also considered separately, because of the alleged strength of these samples (Juola, 2007; Rocha et al., 2013). The values of the separate features were graphically displayed in scatterplots, enabling us to visually distinguish clusters of samples. Moreover, because these measures are based on relatively small amounts of words, I was able to create a larger set of smaller samples. For the new sample set, I decided to take samples of 650 words from the body of each letter. Each sample consists of random sentences that were drawn in such a way that no sample contained a sentence that was already in another sample. Because of the slightly

smaller size of each sample, I was able to extract 51 unique samples (table 3). For each of these samples, I calculated the same richness features as for the samples in the first analysis.

Table 3. Secondary sample set

Letter	Samples
Romans	10
1 Corinthians	10
2 Corinthians	6
Galatians	3
Philippians	2
1 Thessalonians	2
Ephesians	3
Colossians	2
2 Thessalonians	1
Hebrews	6
1 Timothy	2
2 Timothy	1
1 Peter	1
James	2

Feature processing[1]

"From the 22 letter samples, we extracted a total of 3,203,458 features. We disregarded all features with a coefficient of variation (standard deviation / mean) below 0.05. This left 367,822 features. If a sample did not have a certain feature, because it did not occur or did not reach a threshold (relevant for certain ratios), we set the feature at the lowest value observed in all letter samples.
We separated the features into:

- M: 21 overall measurements
- S: 22,898 measurements on syntax, without any reference to lexical items
- V: 311,928 token *n*-grams and syntactic features with reference to lexical items
- C: 32,975 character *n*-grams

The union of all four was kept as X. Note that X is dominated by V, because that has by far the most features.

We took the six first (sometimes only) samples of each letter generally attributed to Paul. We then built models using five of these samples. A model consists of the mean and standard deviation of the

[1] Since the processing of the features was conducted by my supervisor Hans van Halteren, the whole section on feature processing is written by him.

five feature values. In addition, we built 100 models each using a random choice of five of the 22 samples. For each model, we calculated a score for each sample by adding the penalties for each feature:

- Base value is Abs(FeatVal - ModelMean) ^ DiffExp
- This is only used if (FeatVal - ModelMean) > DiffThresh
- If the result is larger than PenaltyCeiling, it is set to PenaltyCeiling
- If MeanSd for a feature is 0, and FeatVal is not equal to ModelMean, penalty is PenaltyCeiling

Various hyperparameters were used:

- DiffExp 0.0 to 4.0 by +0.5
- DiffThresh 0.0 to 2.0 by +0.5
- PenaltyCeiling 5.0 to 40.0 by *2

After running a hyperparameter setting on all 106 models and all 22 samples, a linear model was trained to predict the sample score on the basis of the mean model score over all samples and the mean sample score over all models. The final sample score was then calculated by dividing the raw score by the linear model prediction. As scores are penalties, low is good.

We then picked the best hyperparameters for each of M, S, V, C, and X. The random models were no longer used, but only the six models based off Pauline letters (i. e. five first samples). We used two criteria:

- If we rank all samples, how many certainly non-Pauline samples have lower penalties than the highest penalty Pauline sample (false accepts)? This should be as low as possible.
- What is the Z-score of the held out Pauline samples with regard to the scores of the non-Pauline samples? This should be as low (high is negative) as possible.

We averaged these values over the six models. The first criterion is the primary one, the second only comes to play with equal values for the first. The best settings with values were:

Table 4. Optimal settings with values

Type	DiffThresh	PenaltyCeiling	DiffExp	FA	Z
M	0.0	40.0	2.0	0.83	-0.67
S	2.0	40.0	2.5	0.67	-0.015
V	2.0	40.0	3.5	1	-0.24
C	2.0	10.0	0.5	1.67	-0.22

Results

Results per feature group

The following values were obtained:

Table 5. Values per sample per feature group

Sample	M	Syntax	Vocabulary	Character *n*-grams	All
1 Cor 1	-1.138095	-1.727107	-0.803788	-0.524077	-0.890378
1 Cor 2	-0.987828	-2.223819	-0.850189	0.377609	-0.853105
1 Cor 3	-1.119360	-1.921689	-0.926717	-0.149115	-1.017185
1 Cor 4	-1.078545	-2.621106	-1.035375	0.165176	-0.391000
2 Cor 1	-0.914258	-1.257746	-0.708530	-1.565795	-0.888516
2 Cor 2	-1.431675	-3.326214	-1.483363	-0.946768	-1.593510

1 Thess	-1.018884	-0.459351	-0.603312	-0.628848	-0.618089
Gal	-1.114659	-1.340088	-1.183259	-0.994891	-1.274485
Philipp	-0.933405	-0.401283	-0.983138	-1.254817	-0.943273
Rom 1	-1.150298	-1.742128	-0.651400	-0.621359	-0.757932
Rom 2	-0.809745	-1.802174	-0.954889	-0.889056	-1.109776
Rom 3	-1.167751	-2.880910	-1.234934	-0.082030	-1.289783
Rom 4	-1.127546	-2.623060	-1.073000	-0.635229	-1.210235
2 Thess	1.325834	-0.908061	-1.265642	-0.853087	-1.324957
Eph	-0.972449	0.911420	1.623533	-0.200124	1.453423
Col	-0.814596	0.047741	1.174052	-0.641928	1.077883
1 Tim	-0.517246	0.313876	0.190895	2.051789	0.443260
2 Tim	1.042791	1.297225	0.656345	0.065078	0.776023
Heb 1	-0.783702	-0.032647	-0.319913	0.227454	-0.339816
Heb 2	-0.357897	-1.084067	-0.660051	0.545323	-0.570577
1 Pet	1.506373	0.738301	-0.482064	0.165176	-0.391000
Jas	-0.429109	-1.283788	-0.917154	-1.359681	-1.124240

For the general features, all samples taken from genuinely Pauline letters are below 0.80, with Romans 2 as the least 'Paul-like' sample (-0.81). Interestingly, both Colossians (-0.81) and Ephesians (-0.97) scored within the Paul range, albeit barely, with only one (Colossians) or three (Ephesians) genuinely Pauline samples scoring higher. 2 Thessalonians (1.33), the other disputed letter, does not fit the model very well: only 1 Peter (1.51) scored higher. There were no false accepts, since no sample that is known to be non-Pauline scored higher than Romans 2.

The syntactic features do not perform equally well. Here, Philippians was the highest scoring Pauline sample with a value of -0.40. Two samples from non-Pauline letters. James (-1.28) and Hebrews 2 (-1.08), scored lower, although both are in the higher regions of the 'Pauline zone', with only two (Hebrews 2) or three (James) Pauline samples scoring higher. Colossians (0.04) and Ephesians (0.91), both of which had Paul-like values in the general features, have relatively high values in the syntactic features. Based on these results, I would not qualify them as genuinely Pauline. 2 Thessalonians (-1,.27) appears to be within the Paul range, with both 1 Thessalonians (-0.46) and Philippians (-0.40) scoring higher.

Vocabulary-wise, the highest scoring Pauline sample was 1 Thessalonians (-0.46). James (-0.92) and Hebrews 2 (-0.66) again scored lower than the highest Pauline samples, with five (James) and two (Hebrews 2) genuinely Pauline samples above them. Colossians (1.17) and Ephesians (1.62) have the highest values of all samples and would thus not be classified as genuinely Pauline. Interestingly, the disputed Second Letter to the Thessalonians (-1.27) had the second lowest score. In fact, it scored lower than all samples on which the model was based.

In the character *n*-grams, the highest scoring genuinely Pauline sample is 1 Corinthians 2 (0.38). This sample scores remarkably high: only two samples (Hebrews 2 and 1 Timothy, both non-Pauline) score higher. All samples from 1 Corinthians have remarkably high values, ranging from -0.52 (1 Cor 1) to 0.38 (1 Cor 2). Interestingly, James (-1.36) has the second lowest value: only 2 Cor 1 (-1.57) better fits the model. All other non-Pauline samples have a value of 0.06 (2 Timothy) or higher, meaning that they score higher than all Pauline samples except the two highest-scoring samples from 1 Corinthians. The Second Letter to the Thessalonians (-0.85) and the Letter to the Colossians (-0.64) score well within the Pauline range, each having eight genuinely Pauline samples scoring higher. Ephesians (-0.20) has a notably higher value, but it still scores lower than four genuinely Pauline samples (including the high scoring samples from 1 Corinthians, which, based on these values, would not be classified as genuinely Pauline).

Since the vocabulary features constitute by far the largest group in the feature set, the values obtained over all features resemble the values of the vocabulary features. The highest-scoring Pauline

sample is 1 Cor 4 (-0.39). The non-Pauline samples taken from James (-1.12), and Hebrews 2 (-0.57) scored lower than 1 Cor 4. Just like in the vocabulary features, 2 Thessalonians (-1.32) has the second lowest value. Colossians (1.08) and Ephesians (1.45), the other disputed letters, have the highest values of all samples.

Figure 1 shows the two feature groups with the least false accepts (general and syntactic) graphically displayed in a scatterplot. The plot clearly shows that none of the disputed letters deviates obviously from the genuinely Pauline letter samples in both features. 2 Thessalonians appears far from the clustered Pauline samples because of its high value for the richness measures, but in the syntactic features, its value is not significantly higher than the Pauline samples. For Ephesians and Colossians, the opposite holds: their values for the general features are comparable to those of the Pauline samples, but in the syntactic features, they deviate. Of the non-Pauline samples, only 2 Timothy and 1 Peter are clearly distinct in both overall measures and syntax.

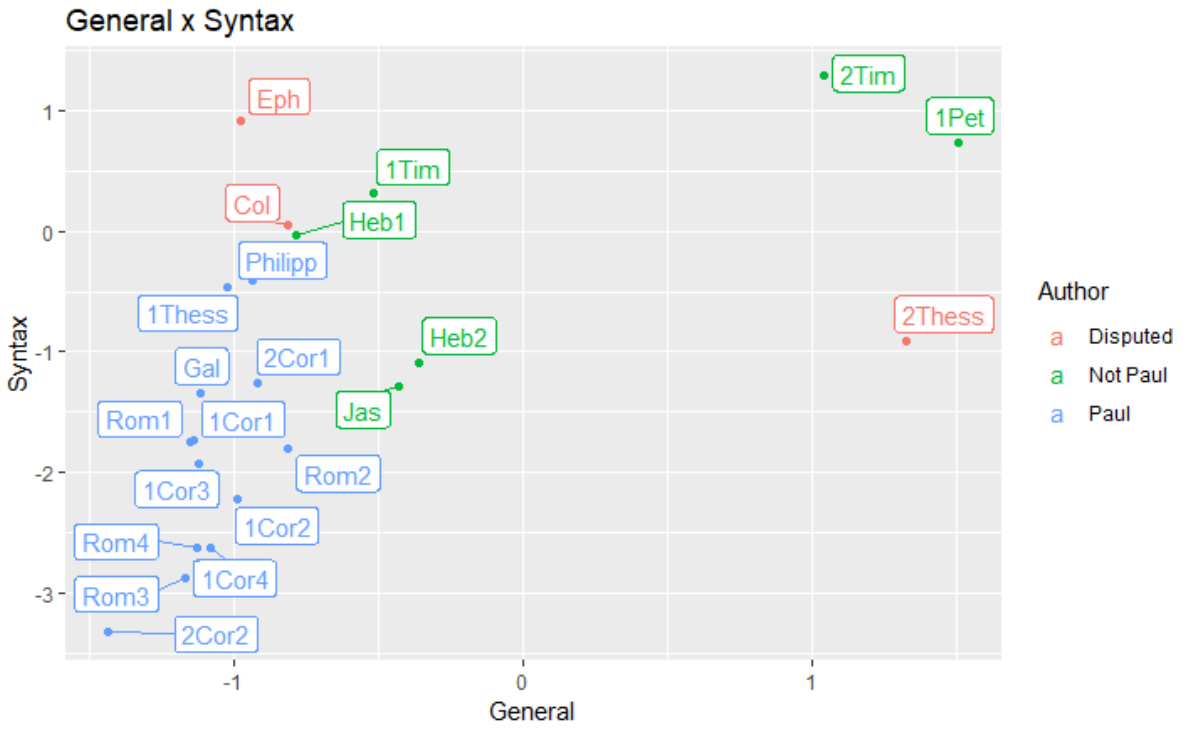

Figure 1. General measurements x Syntactic features

Richness features

The figures 2 and 3 visualises the values of the features ‘form nonzipf’ (deviation of the zipf distribution for forms) and form entropy for both sample sets. Unsurprisingly, both plots show a similar tendency. The non-Pauline letters tend to have higher values for form entropy and relatively low values for the nonzipf measure; the disputed letters have lower values for form entropy and higher values for the nonzipf measure; the Pauline letter samples are somewhat in between. However, the three groups of samples show a considerable amount of overlap. The disputed letters are most notably close to 1 Thessalonians and Philippians and (in the large sample set) to 2 Corinthians. Based on these plots, the disputed letters should not be qualified as non-Pauline. On the other hand, these plots do not provide evidence that they are genuinely Pauline either.

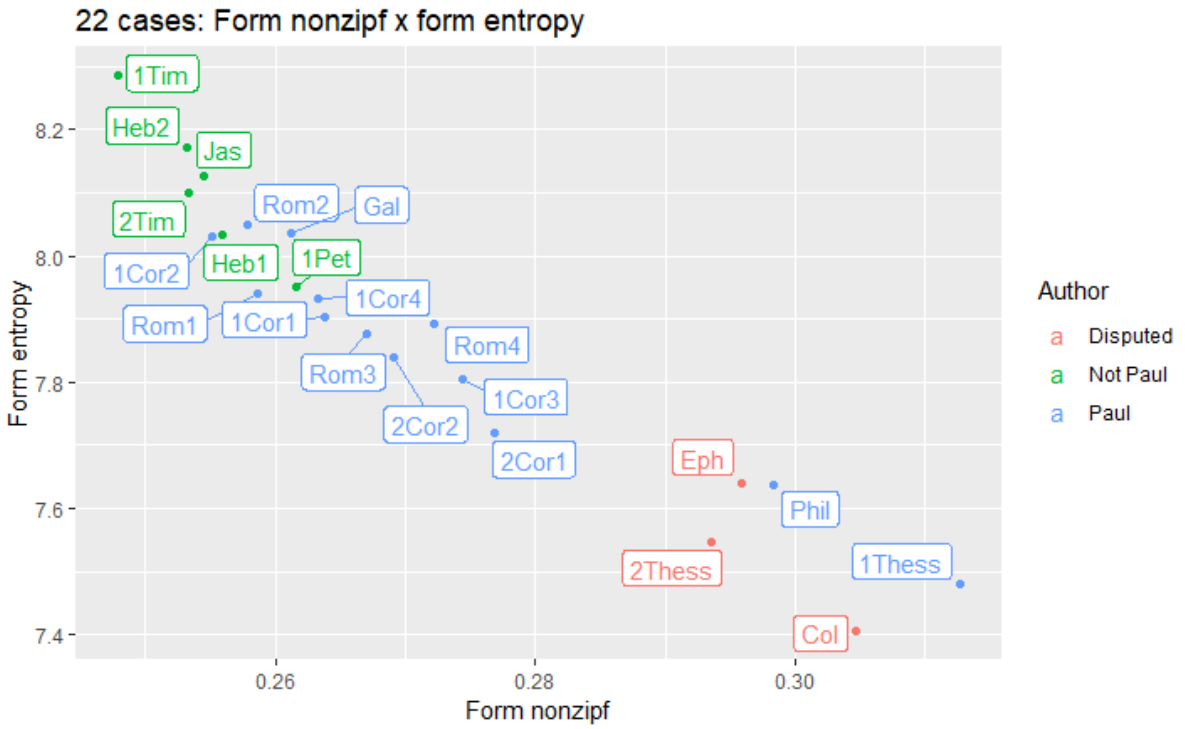

Figure 2. 22 samples: form nonzipf x form entropy

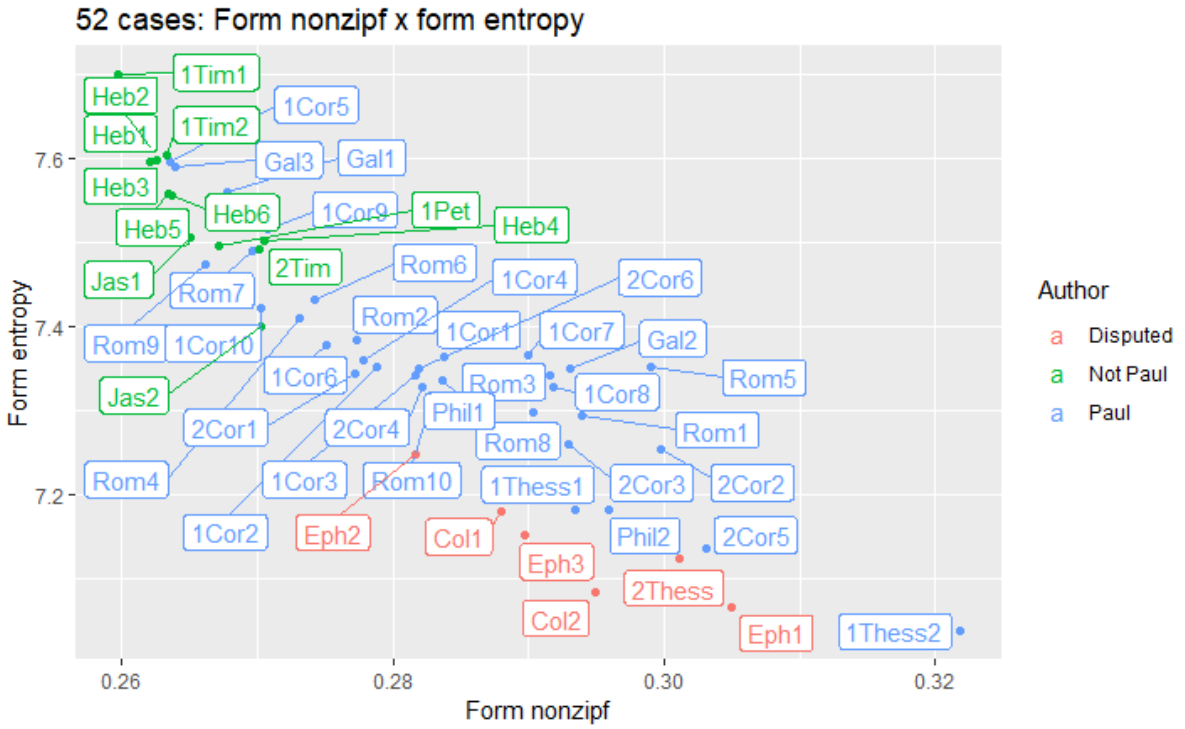

Figure 3. 52 samples: form nonzipf x form entropy

The same can be said for the scatterplots in the figures 4 and 5, showing for each sample the type/token ratio of the form and probability of a hapax legomenon. The plot based on 22 samples seems to show a cluster consisting of the disputed letters, Philippians and 1 Thessalonians, this cluster does not appear in the plot based on 52 samples.

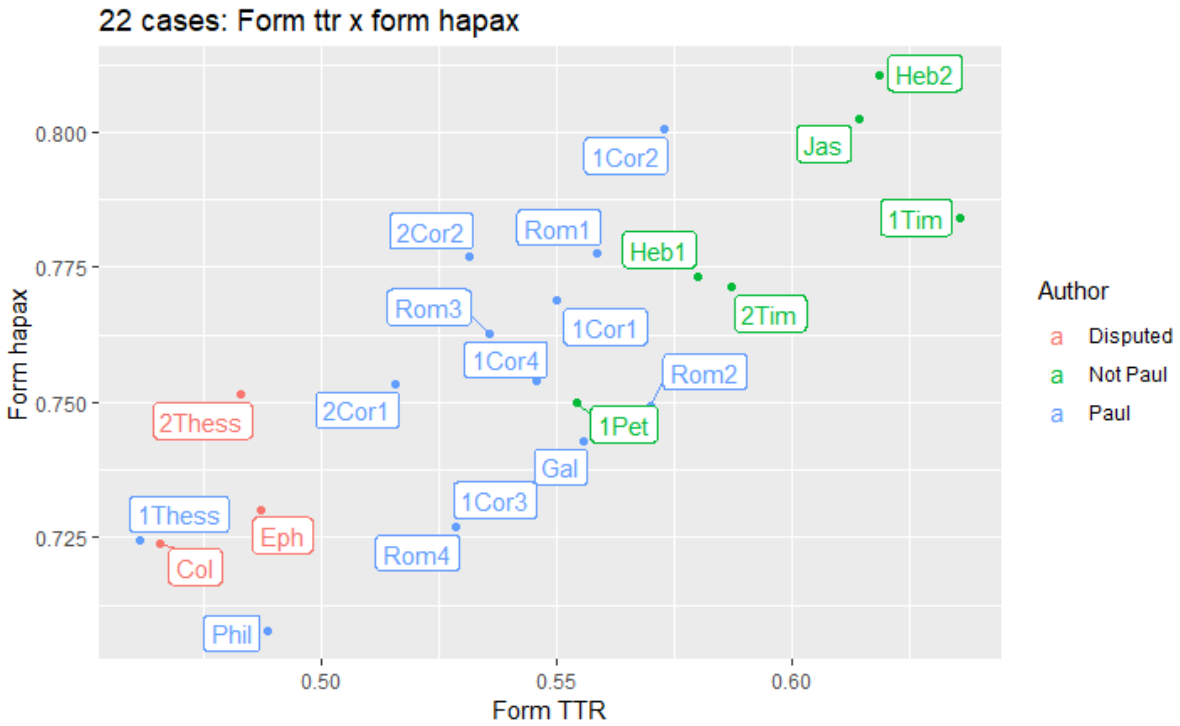

Figure 4. 22 samples: form TTR x form hapax legomenon probability

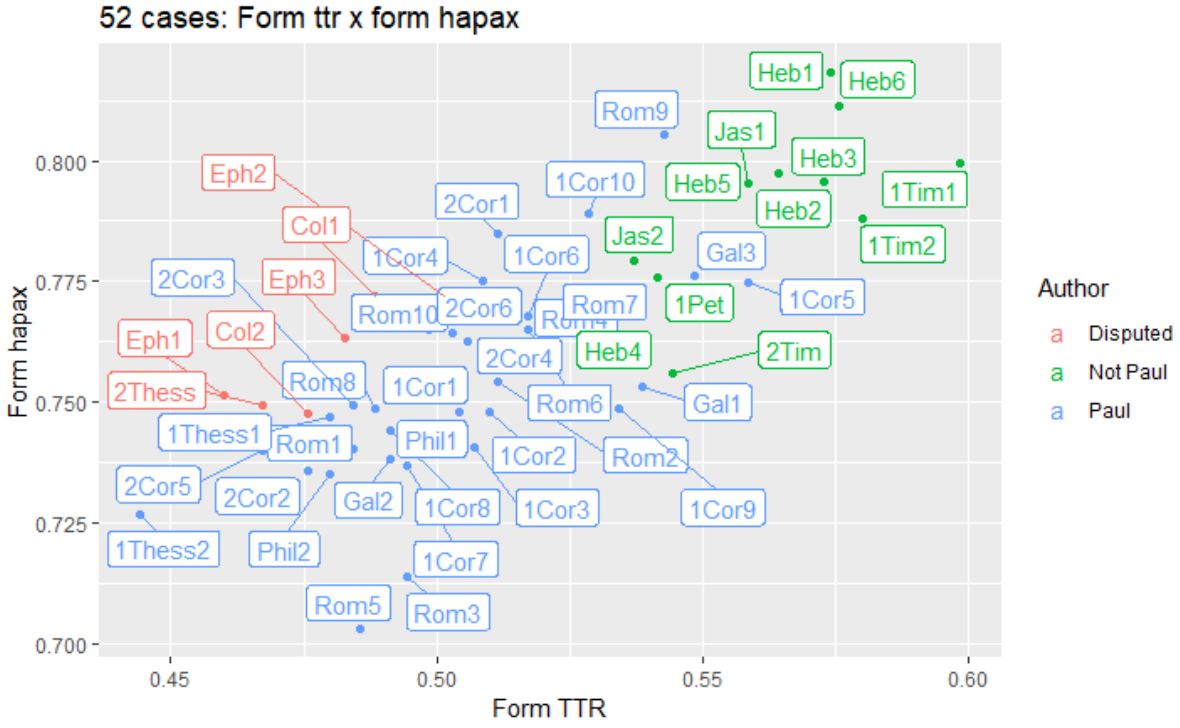

Figure 5. 52 samples: form TTR x form hapax legomenon probability

Figures 6-9 show the same measurements plotted for the alphabetical rewrites. These plots are even more obscure: no clusters are visible whatsoever, and no inferences can be made about the authorship of any letter based on these plots.

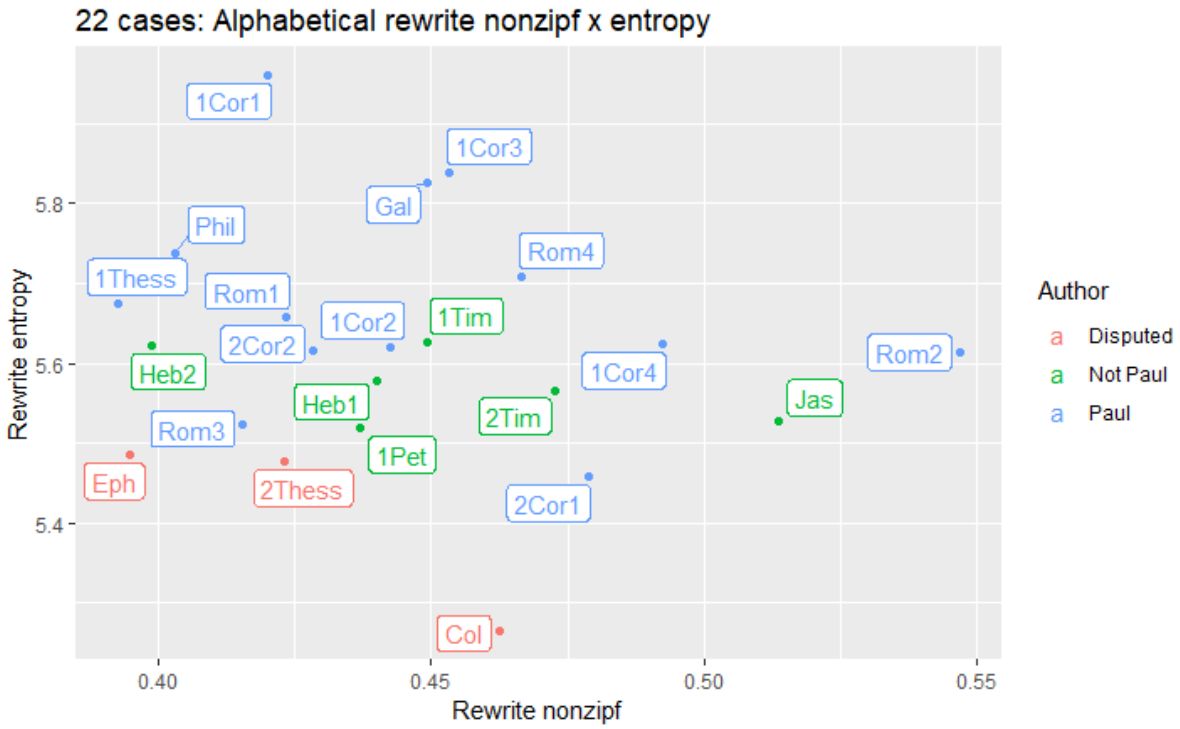

Figure 6. 22 samples: alphabetical rewrites nonzipf x entropy

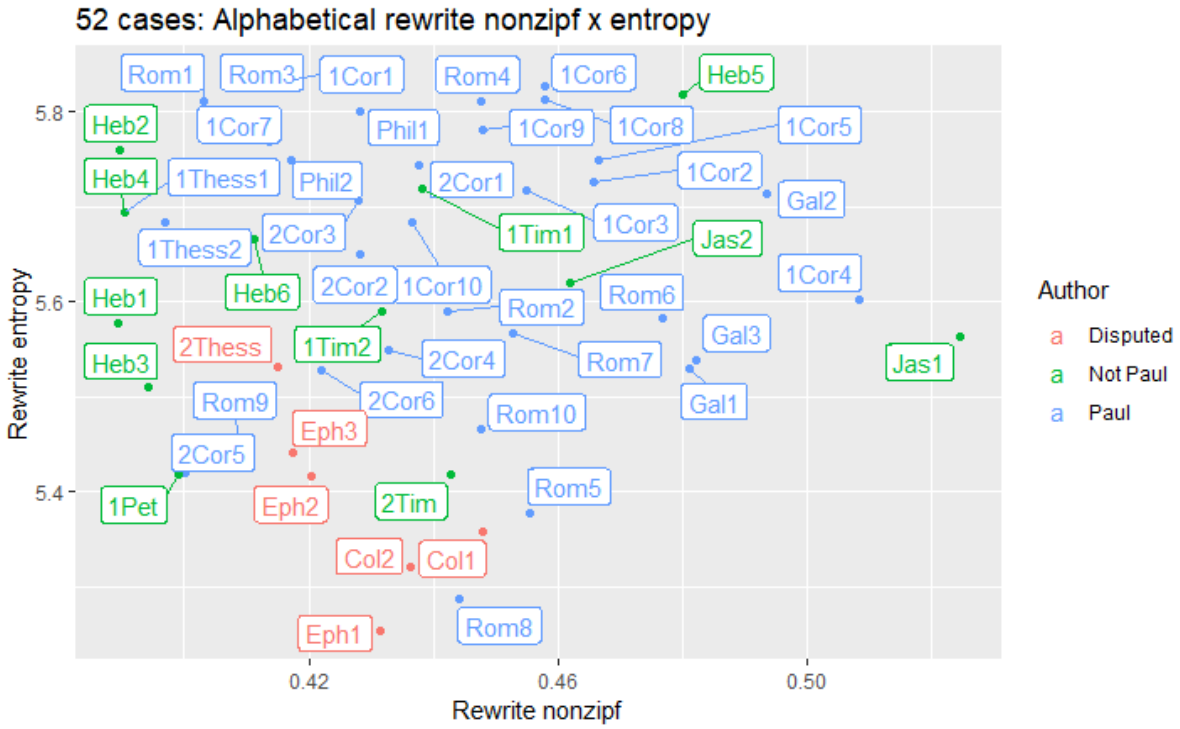

Figure 7. 52 samples: alphabetical rewrites nonzipf x entropy

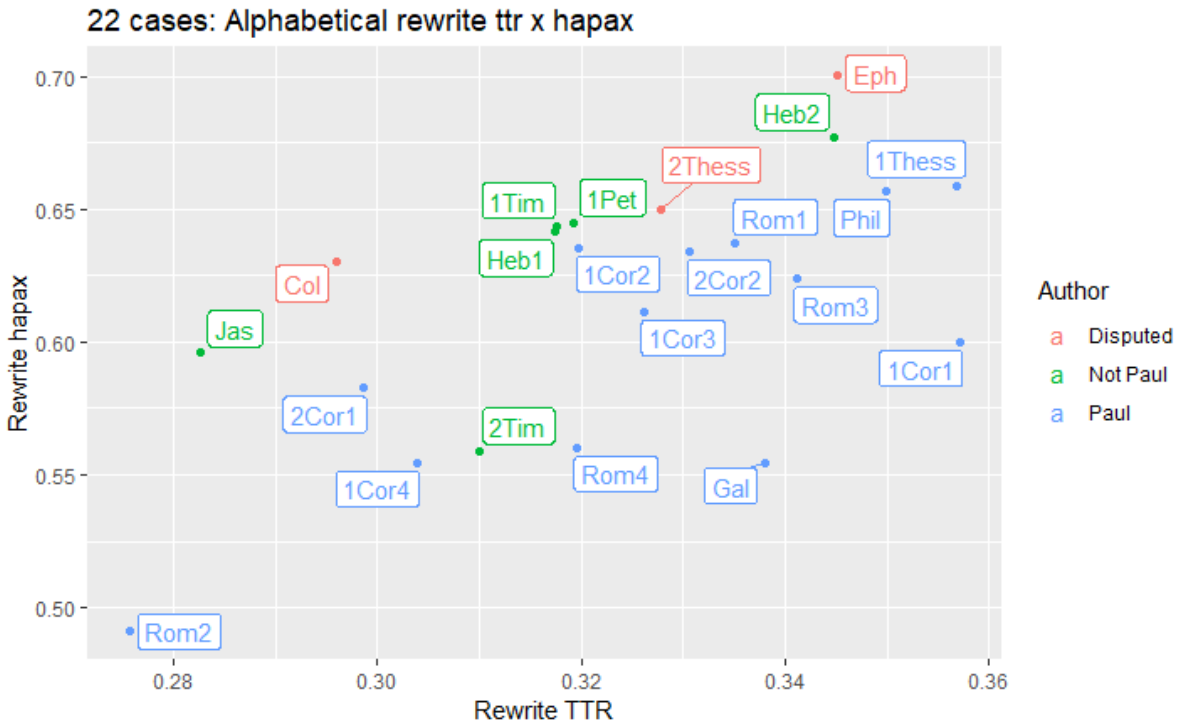

Figure 8. 22 samples: alphabetical rewrites TTR x probability hapax legomenon

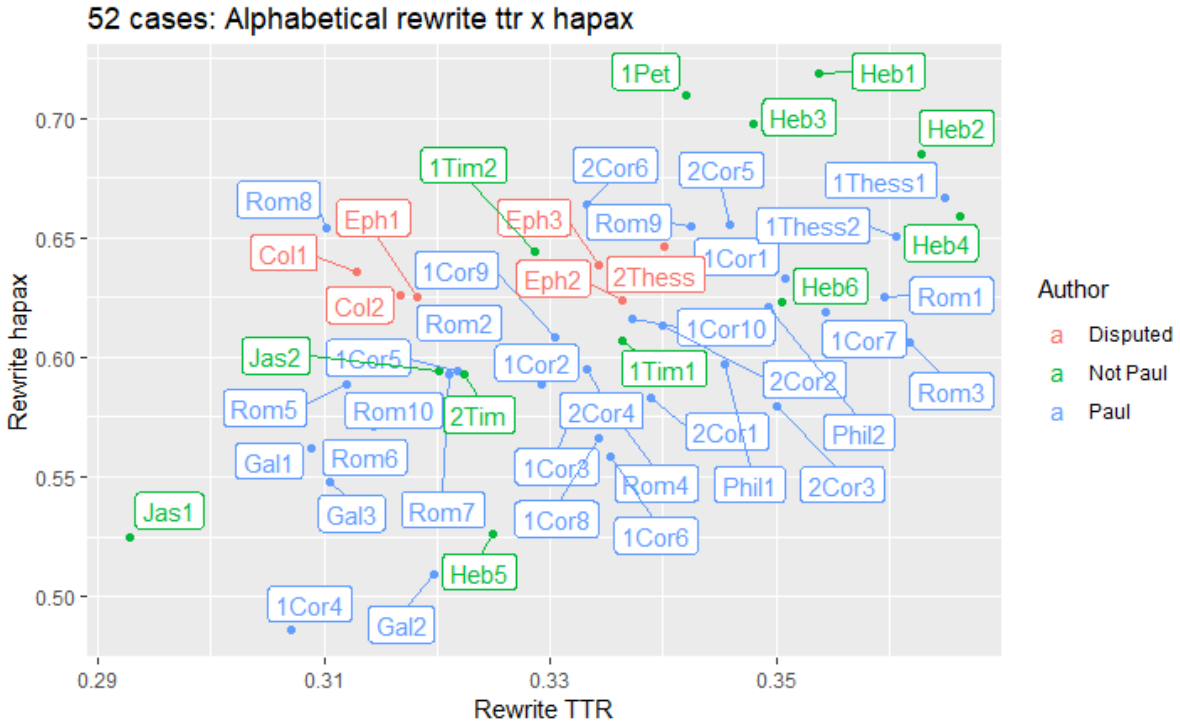

Figure 9. 52 samples: alphabetical rewrites TTR x probability hapax legomenon

Discussion and conclusions

Feature groups

When taking a closer look at the features, it becomes clear which features are the most distinctive for each sample. In this section, I will discuss each broad category (vocabulary, syntax, character *n*-

grams and general features) separately. For each category, I will try to explain why certain (kinds of) features perform well, and how this might have possibly affected the outcomes. I will focus on the most distinctive features for each letter, meaning the features with the highest Z-scores.

Vocabulary features

In most publications concerning the authorship question of the disputed letters, claims of authorship are often based on deviations in the vocabulary. Especially the vocabulary of the Pastoral Epistles deviates so strongly from the other letters, that many scholars have used it as an argument against their authenticity. This is indeed reflected in the outcome of our model: the vocabulary features of 1 and 2 Timothy are rated as very un-Pauline.

The vocabulary features can be expected to reflect the content of the letters. This appears to be the case. One of the concepts that sets the pastorals apart from the genuinely Pauline letters is the concept of 'godliness' (εὐσέβεια). The token-1-gram of the lemma εὐσέβεια is indeed the 7th strongest feature of all T1-grams of 1 Timothy. Another notable difference between the Pastorals and the other letters is the high prevalence of proper names (both personal names and toponyms) in the former. This is reflected in the data as well: the 50 best T1-grams in 2 Timothy contain 6 proper names, compared to only one in the 50 best T1-grams in Galatians.

The Letter to the Romans is often described as an ambassadorial letter (Klauck, 1998, p. 303). This might explain the prevalence of the word σύ ('you') in the dative case: the reader is being addressed. The personal contact between author and addressee might also be reflected in the remarkable use of the word ἀσπάζομαι ('greet'): all samples from Romans are characterised by this lemma and combinations with it, such as καί_ἀσπάζομαι ('and_greet'). Holloway (2013) notes that the Letter to the Romans has a conversational rhetorical style, with a lot of rhetorical questions. This might also be an explanation why the reader is adressed so often. It might also explain the prevalence of the word μή, the subjective negation. It is used with imperatives, certain kinds of conjunctives and infinitives. Especially imperatives can be expected to be common when the author is actively engaging in a debate on moral and theological questions. Also notable is the use of the preposition εἰς in Romans. Possibly, this has something to do with the fact that the direct object of the verb πιστεύω ('believe') takes a prepositional phrase with εἰς as its direct object.

The letter to the Galatians is infamous for its harsh language (Longenecker, 2003). This is mostly reflected by the use of the lemma μετατίθημι ('remove, desert'). This was already noted by Du Toit (2014) and is reflected in our feature set. According to Du Toit (2014), this lemma is especially useful to convey a severe message, since the combination of tau and theta makes it sound harsh and hard to pronounce. Moreover, the lemma ἀνάθεμα ('accursed') is characteristic for the Letter to the Galatians. It is not hard to imagine how this would fit into such a severe letter!

General observations

For all letters, many of the most distinctive T1-grams are lemmas that are somehow related to God or theology. This is somewhat surprising, since many of those words can be expected to be relevant to all letters. These include words like λατρεύω ('serve', typical for the third sample from Romans), ἀγάπη ('love' (noun), typical for 1 Thessalonians) and καθαρός ('pure', typical for 1 Timothy).

Especially powerful throughout many samples are tree *n*-grams containing the lemma ἀλλά ('but'). Apparently, not the frequency of this word, but rather the contexts in which it is used, have great distinctive power. The complexity of how this word can be used to establish authorship potentially makes it a powerful feature: it is not very likely that someone trying to mimick Paul's style would successfully forge the subtle differences.

The relation between content and most distinctive T1-grams is not always clear. For example, 1 Corinthians is generally a friendly letter that puts great emphasis on unity. However, the 15 best T1-grams include μωρία ('foolishness') and ἄπιστος ('infidel'). Possibly, Paul uses other words to express these concepts in his other letters. It is also likely that these concepts are used to establish a

strong black-and-white distinction between the 'good guys' (namely all those who believe in Christ) and the 'bad guys' (those who don't), as a way to establish unity through a common enemy. All four samples from 1 Corinthians are distinguished by lemmas that are somehow related to language, such as γλῶσσα ('tongue, language'), λαλέω ('talk'), αἰτέω ('ask, request'), and προφητεύω ('preach'). Especially the third sample of 1 Corinthians is characterised by the overuse of the word ἐάν ('if'). This does not necessarily imply a difference in topic or content: it is possible that in the other letters (or samples!), a synonym (such as εἰ) prevails.

According to Lincoln (2003), the Letter to the Ephesians is characterised by a focus on the ἐκκλησία ('church') and the σῶμα ('body (of Christ)') as a metaphor for the church. However, this is not apparent from our features: ἐκκλησία nor σῶμα appear in the 300 most distinctive vocabulary features. Possibly, this is because the author of Ephesians uses those words in a different meaning or context, but not necessarily more frequently than the other authors.

Generally, the tree *n*-grams and certain ratios are most distinctive, followed by T1-grams of lemmas, followed by T1-grams of forms, which are followed by T2-grams, usually of parts of speech or relations in combination with lemmas. I had expected that the token *n*-grams would outperform the tree *n*-grams, since they are more likely to reflect the author's stylistic choices in a language with a free word order, such as Greek. A possible explanation is that in Greek, dependencies can stand relatively far away from their heads compared to English dependencies, because the inflection often clarifies the relation between the two. Therefore, units that belong together will often not be captured in the token *n*-grams.

It is worth noticing that the tree *n*-grams mostly contain function words, parts of speech and relations, whereas the best performing T1-grams are almost all content words. These are often words that only appear once or twice in the letter of interest and almost nowhere else. Due to the small sample size, this might be a matter of chance and might thus negatively influence the analysis. In an inflectional language like Greek, this is especially the case for forms. For example, the most distinctive form 1-gram in Philippians is αἰσχυνθήσομαι ('I will be put to shame'), which appears only twice in the whole NT: once in Philippians and once in 1 Corinthians. The corresponding lemma, αἰσχύνομαι, appears five times in the whole NT (www.biblehub.com), making it not incredibly rare, but not very common either. It is therefore very doubtful if this feature can be considered typical for the Letter to the Philippians. The lemmas and forms included in the tree *n*-grams, however, are most often common function words. These are probably more reliable, both because of their frequency and because of the fact that they are harder to mimick, making them less susceptible to forgery.

As expected, lemmas seem to perform better than forms, especially in the token *n*-grams. This is in line with findings of Tambouratzis et al. (2004). I suspect that this is connected to the fact that in an inflectional language like Greek, certain forms can be extremely rare, especially in such small sample sizes.

Another difference between the tree *n*-grams and the token *n*-grams is that, for the token *n*-grams, the shorter *n*-grams outperform the longer *n*-grams for all samples. For tree *n*-grams, bi- and trigrams seem to perform equally well.

There are several elements in the features that might obscure the outcome of the analyses. First, for all samples, the feature types RATIO_ADJPERNOUN, RATIO_CASEPERNOUN and RATIO_AFTER_HEAD_TYPE were present in the best 200 features. For example, in the sample taken from the Letter to the Galatians, almost one fifth of the 100 best features consisted of features of the RATIO_CASEPERNOUN type. In the third sample taken from 1 Corinthians, 12 out of 100 most distinctive features are of the type RATIO_ADJPERNOUN, 11 are of the type RATIO_AFTER_HEAD, and 37 are of the RATIO_CASEPERNOUN type. This is problematic, since the feature values indicate that many of the nouns that appeared in the features were not very frequent at all. Due to the small sample size, it is very possible that these ratios are for a great deal based on chance, rather than reflecting the choices of the author. On the other hand: if you do not

include a feature in the feature set because it is not common enough, the importance of the feature in samples where the threshold is only barely met can be overdone, skewing the results of the analysis.

There is also a great deal of overlap in the features, most notably for lemmas that only have one form, as is the case for most conjunctions and adverbs. Although modern authorship attribution algorithms should be able to deal with overlap in features, it might be useful to create one category for words where the form equals the lemma in all cases. A related problem is the problem of empty categories. All empty categories have the same lemma, form and part of speech, namely 'EMPTY'. It might be useful to exclude the form and lemma from empty categories from the tree *n*-grams.

There also appear to be some problems with the code that was used to extract the features. 1 Corinthians contains the feature V_T2_WL_κερδήσω_. Here, the lemma seems to have disappeared from the token *n*-gram. There also seems to be a problem with the VERB_DEP_POS types of features: these include not only the parts of speech of the dependency (usually the object) of the verb, but also the parts of speech of constituents that share a head with the verb. Since the features were extracted in exactly the same way for all cases, this does not necessarily make the results less reliable, but it is very unclear what these features actually mean.

Syntactic features

One of the reasons the authorship of Colossians and Ephesians is disputed, is the elaborate style the author of these letters employs. The letters are characterised by large amounts of conjunctions and subjunctions and elaborate prepositional phrases (Lincoln, 2003). This is indeed reflected in our features. Both Ephesians and Colossians have a high ratio of coordinations where a prepositional phrase and two verbal phrases are coordinated, which indicate long sentences. They also contain notably many rewrites that start with a conjunction and have many dependencies, including many auxiliaries and adverbials. Ephesians also has a high ratio of coordinated conjunctions.

In the vocabulary features, the tree *n*-grams outperformed the token *n*-grams, but in the syntactic features, this is no longer the case. Possibly, it is the combination of syntax and lexical items that makes the trees with lexical features so powerful. For many samples, among the most distinctive features were rewrites and coordination ratios. The rewrites are only taken from the first 700 words of each sample; a follow-up study could extract rewrites over the whole sample.

Unfortunately, due to a typo in the program used to extract all features, the tree bigrams that were to have a relation as their first element, all have the lemma (so a Tree2_RR type of feature can include "ὑπακούω_obl" instead of "comp_obl"). Therefore, some lexical information did end up in the syntactic features, and there are no tree bigrams that start with a relation.

Character *n*-grams

The character *n*-grams performed less than the other feature groups, but were still able to distinguish moderately succesfully between Paul and not-Paul, with two false accepts and two false rejects.

It may be expected that the most distinctive character *n*-grams are also present in the most distinctive T1-grams. This appears to be the case. For example, in the first sample of 1 Corinthians, the character *n*-gram C4_ἀνακ corresponds to the high-scoring feature ἀνακρίνω. However, this is not the case for all high-scoring character *n*-grams. For example, the highest scoring character *n*-gram for Colossians is C5_χόμεν, but this is not reflected in the most distinctive 2000 features.

As mentioned above, the samples from 1 Corinthians all have a high prevalence of words that are related to language. In the character *n*-grams from the second sample from 1 Corinthians, the features C3_γλώ, C3_ώσσ, C4_γλώσ, C4_λώσσ, and C5_γλώσσ are all very distinctive. In the T1-grams from that same sample, there are no extremely high-scoring T1-grams that correspond to any of these character *n*-grams; however, a bit lower are T1_W_γλώσσαις and T2_GL_pred_γλῶσσα. This is one of the strengths of character *n*-grams: they incorporate words that share a stem, but not a lemma.

Another advantage of character *n*-grams is the fact that they can contain affixes. Greek is characterised by a large amount of prefixes, such as ana-, kata-, meta-, and idio-. They can be attached to different parts of speech. In the PROIEL treebank, there is no morphological analysis of the lemmas, and neither did I extract them manually, so these are not counted in the vocabulary features.

The character *n*-grams also contain syntactic information. Because Greek is an inflectional language, character *n*-grams of inflectional suffixes can reflect case, number, and gender. Interestingly, one of the most distinctive character *n*-grams of the first sample from 1 Corinthians is C5_oς&δὲ (the & reflecting a white space). Δὲ is a particle that can indicate the beginning of a new sentence, 'and', 'but' or even 'because'. How this particle is used largely depends on the personal style of the author, which makes it very useful in authorship attribution. The 5-gram oς&δὲ indicates that the author of 1 Corinthians often used this particle after a word in the nominative singular (oς being the suffix indicating nominative singular for many male nouns and adjectives).

Du Toit (2014) states that, in terms of style, Paul seems to be abherent to Hermogenes**,** who recommended that the sound of the language used should mimick the intended tone of the letter. As I mentioned above, an example of this would be the use of the verb μετατίθεσθε in the letter to the Galatians. At first glance, the character *n*-grams from Galatians do not seem to contain more 'harsh' sounds than the other sets of character *n*-grams, but it might be interesting to look into this.

General discussion

As expected, the results of the analysis do not provide an ultimate answer to the question who wrote the disputed epistles. However, the results at hand are interesting and this study can be a beginning for further investigation.

The main question I tried to answer in this thesis was whether it is possible to use modern authorship attribution techniques in order to classify the letters in the Pauline corpus by author. Although the feature set used in this study had some significant flaws, especially measures of vocabulary richness quite succesfully distinguished the samples from non-Pauline letters (Hebrews, 1 Peter, 1 and 2 Timothy, and James) from the genuinely Pauline letter samples. Syntax, vocabulary and character *n*-gram features also pointed in the right direction. With some adjustments to the feature extraction algorithm, these types of features are potentially useful to classify letters as Pauline or non-Pauline.

The next question was which of the disputed epistles can be classified as genuinely Pauline, and which cannot. As expected, the answer to this question is not somewhat ambiguous. Based on syntax and vocabulary, Colossians and Ephesians are very probably not Pauline. The character *n*-grams and richness measures provide a more subtle image. Here, the epistles do not tend to cluster with the genuinely Pauline epistles, but they are not as 'far away' as the undisputedly non-Pauline epistles. I lean towards the conclusion that they were not written by Paul. As I expected, Colossians and Ephesians tended to cluster together: if one was classified as Pauline or non-Pauline for a certain feature group, so was the other. This is in line with the general consensus that the letters depend on each other (Klauck, 1998, p. 322).

The Second Letter to the Thessalonians is considered a good piece of forgery by many scholars (Klauck, 1998, p. 395). This is reflected in the data. In vocabulary, character *n*-grams and syntax, 2 Thessalonians does not deviate from the genuinely Pauline epistles. However, the richness features show us another image. Here, 2 Thessalonians is very different from most genuinely Pauline epistles. Vocabulary richness is hard to imitate. I therefore think that this letter is indeed written by someone else, who tried to imitate Paul in vocabulary and syntax. Possibly, part of the resemblance in syntax and vocabulary is caused by the author using genuine Pauline material in his letter.

However, these results should be treated with caution. Although there are few cases where the plot seems to indicate a non-Pauline letter as genuinely Pauline, there are plenty of cases where a Pauline letter deviates from the other genuinely Pauline letters in one or more aspects. If one would

purely base themself on the results of these analyses, Philippians and 1 Thessalonians would not without a doubt be accepted as Pauline. Moreover, based on the character *n*-grams, most samples from the Second Letter to the Corinthians would not or hardly be classified as Pauline. Possibly, this has to do with the influence of Timothy, who is presented as a co-author. It is important to realise that the question of authorship cannot be realised without human interference.

The last question was which (kinds of) features were most distinctive. In line with what I expected, measures of vocabulary richness were useful to distinguish between Pauline and non-Pauline epistles. The reason might be that these measures are relatively unaffected by editing (either by a scribe or by a later editor or copyist). Moreover, they are hard to imitate. However, they need to be handled with care. We know that Paul had an excellent command of Greek, but this was certainly not the case for all early Christians. If we only compare Paul to authors with a limited command of Greek, the text of another native speaker of Greek can be relatively similar to Paul's texts. Although the combination of all general text measures was powerful, no combination of two measures allowed to distinguish between the Pauline and the non-Pauline samples, even when applied to a larger sample set.

Furthermore, I expected that features based on word order (such as the token *n*-grams) would perform better than those based on syntactic structure (such as the tree *n*-grams). This was not the case: for all samples, tree *n*-grams containing lexical items were more powerful than token *n*-grams containing lexical features. The purely syntactic tree *n*-grams did not perform better than purely syntactic token *n*-grams. Possibly, this is because of the distance between head and dependency. Because Greek is an inflectional language, the head and the dependency can be too far away from each other to end up in one token *n*-gram. The token *n*-grams, then, are either based on syntactic relations that are already present in the tree *n*-grams, or they reflect more or less random combinations of words that have no direct syntactic relation to each other. The fact that the tree *n*-grams only perform better than token *n*-grams if they contain lexical items might indicate that especially the combination of vocabulary and syntax is powerful.

I did not have clear expectations of the character *n*-grams, but those performed relatively well. I assume that this is mostly due to their ability to capture frequency of morphemes. Moreover, they combine lexical and syntactic information.

A subtype of the vocabulary features that showed to be very distinctive for all samples were the ratios, especially the RATIO_ADJPERNOUN, RATIO_CASEPERNOUN, and RATIO_AFTER_HEAD_TYPE types of features prevailed. Many of these features captured elements or relations that were quite rare; not necessarily because they were very special, but rather due to the specific nature of the feature combined with the short size of each sample. Therefore, I doubt if they are very reliable. In the analysis, the influence of these rare features is limited by the penalty ceiling, but it would be worth testing if the model improves if these features are removed.

Further research

The findings of this research are interesting, but they are first and foremost an invitation for further research. In this section, I will discuss how the current research could be repeated with possibly better or more reliable results.

For this research, I took random samples from the body and proem of each letter. The only interpolation I removed was the one in 2 Corinthians 6:14 - 7:1. A follow-up study could be more selective in making the samples, for example by identifying and removing more possible interpolations. Other parts that could be omitted are the hymns that Paul quotes in for example Col 1:15-20 and Philippians 2:6-11. Not only are these parts based on non-Pauline material, they are also a different genre and can therefore be expected to deviate (Stuckenbruck, 2003). A drawback of deleting passages is that some samples would become even shorter.

It would also be useful to repeat the analysis with an adjusted feature set. First of all, some typos and coding errors would need to be fixed. The RATIO_ features also need to be revised. Now,

the threshold for a RATIO feature to be included in the final set is that it should appear at least five times. This threshold might be too low. On the other hand: increasing the threshold does not solve the problem that the importance of features that only barely reach the threshold is overdone. Therefore, one might decide to remove (some of) these features alltogether. It might also be advantagious to extend the feature set with more character *n*-grams. Stamatatos (2009) has noted that the optimal length of character *n*-grams depends on the language. Because Greek tends to have longer words than English, it might be useful to incorporate even larger character *n*-grams in a follow-up study. Moreover, a follow-up study could include *n*-grams that explicitly incorporate sentence boundaries, thus showing which lexical items appear at the beginning or the end of a sentence.

The features in each set overlapped for a great deal. This was partially due to the fact that, for some words, the form is always the same as the lemma. Although modern machine learning methods should be able to deal with overlap, it might be more elegant to collapse the form in cases where the form is always equal to the lemma.

Since authorship attribution methods tend to perform better with larger datasets (Juola, 2006), it would do no harm to extract some more features. One obvious feature that is not included in the current feature set would be a feature that relates words (forms and lemmas) to the relation to their heads.

Finally, in this thesis I made a (rather crude) distinction between Pauline, non-Pauline and disputed letters. However, it is known that the Pauline letters came into being through the help of scribes and even co-authors. This might, for example, explain why 2 Corinthians could hardly be classified as Pauline if one were to base the decision purely on the character *n*-grams. Further research could dive deeper into this question.

Bibliography

Anderson, C. P. (1996). Who wrote "the epistle from Laodicea"? *Journal of Biblical Literature, 85*(4), 436-440.

Barr, G. K. (2003). Two Styles in the New Testament Epistles. *Literary and Linguistic Computing, 18*(3), 235-248.

Bee, R. E. (1973). The Use of Statistical Methods in Old Testament Studies. *Vetus Testamentum, 23*(3), 257-272.

Bieringer, R. (1998). *Inleiding tot het Grieks van het Nieuwe Testament*. Leuven, Belgium: Peeters.

Black, M., & Davidson, R. (1981). *Constantin von Tischendorf and the Greek New Testament*. Glasgow, UK: University of Glasgow Press.

Böttrich, C. (2017). Codex Sinaiticus and the use of manuscripts in the Early Church. *The Expository Times, 128*(10), 469-478.

Crossway. (2012). *The Greek-English New Testament*. Wheaton, IL: Crossway.

Den Heyer, C. J. (1998). *Paulus, Man van Twee Werelden*. Utrecht, The Netherlands: Meinema.

Du Toit, A. (2014). Galatians and the περί ίδεων λόγον of Hermogenes : a rhetoric of severity in Galatians 1-4. *HTS: Theological Studies, 70*(1), 1-10.

Dunn, J. D. G. (Red.). (2003a). *The Cambridge Companion to St. Paul*. Cambridge, UK: Cambridge University Press.

Dunn, J. D. G. (2003b). Introduction. In J. D. G. Dunn (Red.), *The Cambridge Companion to St. Paul* (pp. 1-18). Cambridge, UK: Cambridge University Press.

Ebel, E. (2012). The Life of Paul. In O. Wischmeyer (Ed.), *Paul: Life, Setting, Work, Letters* (pp. 97-110). London, UK: T&T Clark International.

Eckhoff, H., Bech, K., Bouma, G., Eide, K., Haug, D., Haugen, O. E., & Jøhndal, M. (2018). The PROIEL treebank family: a standard for early attestations of Indo-European languages. *Language Resources & Evaluation*, *52*(1), 29-65.

Elliot, W., & Valenza, R. J. (1996). And then there were none: Winnowing the Shakespeare claimants. *Computers and the Humanities*, *30*, 191-245.

Grabchak, M., Zhang, Z., & Zhang, D. T. (2013). Authorship Attribution Using Entropy. *Journal of Quantitative Linguistics*, *20*(4), 301-313.

Hatzigeorgiu, N., Mikros, G., & Carayannis, G. (2001). Word Length, Word Frequencies and Zipf's Law in the Greek Language. *Journal of Quantitative Linguistics*, *8*(3), 175-185.

Haug, D. T. T., & Jøhndal, M. L. (2008). Creating a Parallel Treebank of the Old Indo-European Bible Translations. In C. Sporleder, & K. Ribarov (Eds.), *Proceedings of the Second Workshop on Language Technology for Cultural Heritage Data* (pp. 27-34). Retrieved from https://github.com/proiel/proiel-treebank

Hirst, G., & Feiguina, O. (2007). *Authorship Attribution for Small Texts: Literary and Forensic Experiments.*. Paper gepresenteerd op de Proceedings of the SIGIR 2007 International Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection, PAN 2007, Amsterdam, The Netherlands. Geraadpleegd van https://www.researchgate.net/publication/221299184_Authorship_Attribution_for_Small_Texts

Holloway, P. A. (2003). The Rhetoric of Romans. *Review and Exposito*, *100*, 112-127.

Holmes, D. I. (2003). Stylometry and the Civil War: The Case of the Pickett Letters. *Chance*, *16*(2), 18-25.

Hubner, H. (2003). Forschungsberichte - Die Diskussion um die deuteropaulinischen Briefe seit 1970. Der Kolosserbrief (I). *Theologische Rundschau*, *68*(3), 264-285.

Hultgren, A. J. (2003). The Pastoral Epistles. In J. D. G. Dunn (Red.), *The Cambridge Companion to St. Paul* (pp. 141-158). Cambridge, UK: Cambridge University Press.

Hupperts, C. A. M. (2008). *Woordenboek Grieks-Nederlands*. Leeuwarden, The Netherlands: Eisma Edumedia bv.

Hurtado, L. W. (2006). *The earliest Christian artifacts: manuscripts and christian origins*. Cambridge, UK: William B. Eerdmans Publishing Company.

Jewett, R. (2003). Romans. In J. D. G. Dunn (Red.), *The Cambridge Companion to St. Paul* (pp. 91-104). Cambridge, UK: Cambridge University Press.

Johnson, S. E. (1987). *Paul the Apostle and His Cities*. Wilmington, DE: Michael Glazier, Inc..

Joosten, J. (2013). Varieties of Greek in the Septuagint and the New Testament. In J. Carleton Paget, & J. Schaper (Reds.), *Languages, writing systems and book production* (pp. 22-45). Cambridge, UK: Cambridge University Press.

Juola, P. (1997). What can we do with small corpora? Document categorization via cross-entropy. In M. Ramscar, & ORG. University of Edinburgh, Department of Artificial Intelligence (Eds.), *Proceedings of an Interdisciplinary Workshop on Similarity and Categorization* (pp. 31-36). Edinburgh, UK: University of Edinburgh.

Juola, P. (2006). Authorship Attribution. *Foundations and Trends in Information Retrieval*, *1*(3), 233-334.

Juola, P. (2007). Becoming Jack London. *Journal of Quantitative Linguistics*, *14*(2-3), 145-147.

Kalimeri, M., Constantoudis, V., Papadimitriou, C., Karamanos, K., Diakonos, F. K., & Papageorgiu, H. (2015). Word-length Entropies and Correlations of Natural Language Written Texts. *Journal of Quantitative Linguistics*, *22*(2), 101-118.

Kestemont, M. (2012). Stylometry for medieval authorship studies. An application to rhyme words. *Digital philology. A journal of medieval cultures*, *1*(1), 42-72.

Kirk, A. (2012). *Word order and information structure in New Testament Greek* (Doctoral dissertation). Utrecht, The Netherlands: LOT.

Klauck, H.-J. (1998). *Ancient Letters and the New Testament* (D. P. Bailey, trans.). Waco, TX: Baylor University Press.

Koppel, M., Akiva, N., & Dagan, I. (2006). Feature instability as a criterion for selecting potential style markers. *Journal of the American Society for Information Science and Technology*, *57*(11), 1519-1525.

Lincoln, A. T. (2003). Ephesians. In J. D. G. Dunn (Red.), *The Cambridge Companion to St. Paul* (pp. 133-140). Cambridge, UK: Cambridge University Press.

Longenecker, B. (2003). Galatians. In J. D. G. Dunn (Red.), *The Cambridge Companion to St. Paul* (pp. 64-73). Cambridge, UK: Cambridge University Press.

MGNT Search Results for "φανερωθη". Retrieved from https://www.blueletterbible.org//search/search.cfm?Criteria=%CF%86%CE%B1%CE%BD%CE%B5%CF%81%CF%89%CE%B8%E1%BF%87&t=mGNT#s=s_primary_0_

Mikros, G., & Carayannis, G. (2000). Modern Greek corpus taxonomy. *Proceedings of the 2nd International Conference on Language Resources and Evaluations*, *3*, Athens, Greece, 31 May - 2 June, pp. 129-134.

Military Bible Association. (2014). *The Holy Bible, Modern English Version (MEV)*. Retrieved from https://www.bible.com/en-GB/versions/1171-mev-modern-english-version

Mitchell, M. M. (2003). 1 and 2 Thessalonians. In J. D. G. Dunn (Red.), *The Cambridge Companion to St. Paul* (pp. 51-63). Cambridge, UK: Cambridge University Press.

Murphy O'Connor, J. (2003). 1 and 2 Corinthians. In J. D. G. Dunn (Red.), *The Cambridge Companion to St. Paul* (pp. 74-90). Cambridge, UK: Cambridge University Press.

Nederlands Bijbelgenootschap. (2008). NBV Studiebijbel: De Nieuwe Bijbelvertaling met uitleg, achtergronden en illustraties. Heerenveen, The Netherlands: Royal Jongbloed.

Polhill, J. B. (1973). The Relationship between Ephesians and Colossians. *Review & Expositor*, *70*(4), 439-450.

Richards, R. E. (1991). *The Secretary in the Letters of Paul*. Tübingen, Germany: J. C. B. Mohr.

Robson, E. I. (1917). Composition and Dictation in New Testament Books. *Journal of Theological Studies*, *18*, 288-301.

Rocha, A., Scheirer, W. J., Forstall, C., Cavalcante, T., Theophilo, A., Shen, B., . . . Stamatatos, E. (2017). Authorship Attribution for Social Media Forensics. IEEE Transactions on Information Forensics and Security, 12(1), 5-31.

Stamatatos, E. (2009). A Survey of Modern Authorship Attribution Methods. *Journal of the American Society for Information Science and Technology*, *60*(3), 538-556.

Stamatatos, E., Fakotakis, N., & Kokkinakis, G. (2001). Computer-Based Authorship Attribution Without Lexical Measures. *Computers and the Humanities*, *35*, 193-214.

Stuckenbruck, L. T. (2003). Colossians and Philemon. In J. D. G. Dunn (Red.), *The Cambridge Companion to St. Paul* (pp. 116-132). Cambridge, UK: Cambridge University Press.

Tambouratzis, G., Markantonatou, S., Hairetakis, N., Vassiliou, M., Carayannis, G., & Tambouratzis, D. (2004). Discriminating the registers and styles in the Modern Greek language – Part 2: Extending the feature vector to optimize author discrimination. *Literary and Linguistic Computing*, *19*(2), 221-242.

Tauber, J. K., ed. (2017) *MorphGNT: SBLGNT Edition*. Version 6.12 [Data set]. https://github.com/morphgnt/sblgnt DOI: 10.5281/zenodo.376200

Theissen, G. (2007). *The New Testament: A Literary History* (L. M. Maloney, trans.). Minneapolis, MN: Fortress Press.

Van Halteren, H., Baayen, R. H., Tweedie, F., Haverkort, M., & Neijt, A. (2005). New machine learning methods demonstrate the existence of a human stylome. *Journal of Quantitative Linguistics*, *12*(1), 65-77.

Van Halteren, H., & Rem, M. (2013). Thumbnail Dealing with orthographic variation in a tagger-lemmatizer for fourteenth century Dutch charters. *Language Resources and Evaluation*, *47*(4), 1233-1259.

Whissel, C., Fournier, M., Pelland, R., Weir, D., & Makarec, K. (1986). A dictionary of affect in language: IV. Reliability, validity, and applications. *Perceptual and Motor Skills*, *62*(3), 875-888.

Whissel, C. (2004). Using Computer-Scored Measures of Emotion and Style to Discriminate among Disputed and Undisputed Pauline and Non-Pauline Epistles. *Perceptual and Motor Skills*, *98*(6), 1117-1125.

Wischmeyer, O. (2012). The Letter to the Romans. In O. Wischmeyer (Ed.), *Paul: Life, Setting, Work, Letters* (pp. 245-276). London, UK: T&T Clark International.

Appendix A: code

```
#GET_FEATURES #tweede versie
use utf8;

#algemene features
my($inputfile, $outputfile, $zipffile);
my(%feats);
my(%monsterfeats);

#steeds gereset
my($tokenposition);
my(%sentenceids, %idform, %idlemma, %idpartofspeech, %idhead, %idmorphology, %idlineno,
%sentenceposition);
my(@currentsentence);

#nodig voor ngrams
my(@tokwor, @toklem, @tokpos, @tokrel);
my(@postree, @formtree, @lemtree, @reltree);
my($ntokens, $rawtext, $nchars);

#globale tellingen
my($nrewrites, $nsentences);
my(@this_richness);
my(%lemmatype, %formtype, %formsperlemma);
my(%lemmatype700, %formtype700);
my(%rewrites_alph, %rewrites_position);
my(%wordlengths, %sentencelengths);

#specifiekere tellingen
my(%totalpos);
my(%posheadfinal, %formheadfinal, %lemmaheadfinal);
my(%lemmaverbperson, %lemmaverbtense, %lemmaverbmood, %lemmaverbvoice,
%lemmaverbrelation, %verbrelations);
my(%verbformasaux, %verblemmaasaux, %verbformwithaux, %verblemmawithaux);
my(%verbwitharticle, %transauxiliaries, %verblemmawithobject, %verbformwithobject);
```

```
my(%nounformarticle, %nounlemmaarticle, %pronouns, %nouns, %nounsadj, %nounscase,
%nounscasenopp, %totalnounscase, %nounssing);
my($totalverbs, $totalprepositions, $totalnouns, $nounsadjectives, $totalpronouns, $totalheadfinal,
$totaltrans, $totalauxverb, $totalsingular);
my(@nounscase);
my($ncoord);
my(%coordposposition, %coordposalpha, %coordlemposition, %coordlemalpha);

chdir("C:/Users/Katarina/Documents/scriptie/Cases/randomcases/featcases4");
open(TOUT, ">:utf8", "outfile.txt");
open(MASTEROUT, ">:utf8", "smallfeatures.txt");
open(PLOT, ">:utf8", "plots.txt");
open(POSPLOT, ">:utf8", "posplots.txt");

my($caseposition, $end);

%monsterfeats = ();

for($caseposition = 1; $caseposition < 23; $caseposition++)
{
   $inputfile = "randomcase${caseposition}.txt";
   $outputfile = "Featcase${caseposition}.txt";
   $zipffile = "Zipfcase${caseposition}.txt";

   open(IN, "<:utf8", $inputfile) or die "no input file";
   open(OUT, ">:utf8", $outputfile) or die "no output file";
 #  open(ZIPF, ">utf8", $zipffile) or die "no zipf file";
   #print PLOT "$inputfile\t";
   make_variables();
   process_parse();
   finish_features();
   get_richness();
   report_features();
}

close IN;
close OUT;
#close ZIPF;
close PLOT;
close MASTEROUT;
close TOUT;
close POSPLOT;

sub make_variables
{
   %transauxiliaries = ();
   %verblemmawithobject = ();
   %verbformwithobject = ();
   %nounformarticle = ();
   %nounlemmaarticle = ();
```

```
%pronouns = ();
%nouns = ();
%nounsadj = ();
%nounscase = ();
%nounscasenopp = ();
%totalnounscase = ();
%nounssing = ();
$totalpronouns = 0;
$totalheadfinal = 0;
$totaltrans = 0;
$totalauxverb = 0;
$totalsingular = 0;

@nounscase = ();

$totalauxverb = 0;
%feats = ();

@tokwor = ();
@toklem = ();
@tokpos = ();
@tokrel = ();

%idpartofspeech = ();
%idform = ();
%idmorphology = ();
%idlemma = ();
%idhead =
%formsperlemma = ();

%totalpos = ();

%lemmatype = ();
%formtype = ();
%lemmatype700 = ();
%formtype700 = ();

$nrewrites = 0;
%rewrites_position = ();
%rewrites_alph = ();

$nsentences = 0;
%sentencelengths = ();
%wordlengths = ();

%lemmaverbperson = ();
%lemmaverbtense = ();
%lemmaverbmood = ();
%lemmaverbvoice = ();
%lemmaverbrelation = ();
```

```
    %verbrelations = ();

    %adjarticle = ();
    %adjbeforearticle = ();

    %nouns = ();
    %nounsadj = ();
    %nounscase = ();
    %pronouns = ();

    %verblemmawithobject = ();
    %verbformwithobject = ();

    %prepositionwitharticle = ();
    %verbwitharticle = ();
    %afterhead = ();

    %posheadfinal = ();
    %formheadfinal = ();
    %lemmaheadfinal = ();

    %coordposposition = ();
    %coordposalpha = ();
    %coordlemposition = ();
    %coordlemalpha = ();

    $nchars = 0;
    $ntokens = 0;
    $rawtext = "";
    $totalsingular = 0;
    $totalnouns = 0;
    $totalpronouns = 0;
    $totalheadfinal = 0;
    $totalverbs = 0;
    $totalprepositions = 0;
    $nounsadjectives = 0;


    %unstable = ();
}

sub reset_variables
{
    %sentenceids = ();
    %idform = ();
    %idlemma = ();
    %idpartofspeech = ();
    %idmorphology = ();
    %idhead = ();
    %idrelation = ();
```

```
    %sentenceposition = ();
    @currentsentence = ();
    $sentenceposition = 0;
    $tokenposition = 0;

    @toklem = ();
    @tokwor = ();
    @tokpos = ();
    @tokrel = ();

    $rawtext = "";
}

sub process_parse
{
    my($line, $lineno);

    $lineno = 0;

    while($line = <IN>)
    {
        chomp($line);
        $lineno++;

        if($line =~ m/<sentence/)
        {
            reset_variables();
        }
        elsif($line =~ m/<token/ or $line =~ m/<slash/)
        {
            push @currentsentence, $line;

            if($line =~ m/ id="([0-9]*)"(?:.*)form="([\p{Greek} ’]*)"(?:.*)lemma="(.+)" part-of-
speech="(..)"(?:.*)morphology="(.{10})"/i)
            {
                my($id, $form, $lemma, $partofspeech, $morphology);

                $ntokens++;
                $sentenceposition++;
                $tokenposition++;

                $id = $1;
                $form = $2;
                $lemma = $3;
                $partofspeech = $4;
                $morphology = $5;

                $sentenceids{$id} = 1;
                $idform{$id} = $form;
                $idlemma{$id} = $lemma;
```

```
    $idpartofspeech{$id} = $partofspeech;
    $idmorphology{$id} = $morphology;
    $idlineno{$id} = $lineno;
    $sentenceposition{$id} = $sentenceposition;
    $totalpos{$partofspeech}++;

    $form =~ s/$form/\L$form/;

    $lemmatype{$lemma}++;
    $formtype{$form}++;

    $formsperlemma{$lemma}{$form}++;
    if($ntokens <= 700)
    {
       $lemmatype700{$lemma}++;
       $formtype700{$form}++;
    }

    $tokwor[$tokenposition] = $form;
    $toklem[$tokenposition] = $lemma;
    $tokpos[$tokenposition] = $partofspeech;

    my$length = get_wordlength($form);

    $rawtext .= $form; $rawtext .= "&";

    if($line =~ m/relation="([a-z]*)"/)
    {
       my($relation);

       $relation = $1;
       $idrelation{$id} = $relation;
       $tokrel[$tokenposition] = $relation;
    }
    else
    {
       $idrelation{$id} = "none";
       $tokrel[$tokenposition] = "none";
    }

    if($ntokens == 700) #is dit slim?
    {
       get_rewrites();
    }
}
elsif($line =~ m/ id="([0-9]*)"(?:.*)empty-token-sort="(.)"/i)
{
    my($id, $form, $lemma, $partofspeech, $morphology);

    $sentenceposition++;
```

```
        $id = $1;
        $form = "EMPTY";
        $lemma = "EMPTY";
        $partofspeech = "$2";
        $morphology = "EMPTY";
        $partofspeech .= "g";

        $sentenceids{$id} = 1;
        $idform{$id} = $form;
        $idlemma{$id} = $lemma;
        $idpartofspeech{$id} = $partofspeech;
        $idmorphology{$id} = $morphology;
        $idlineno{$id} = $lineno;
        $sentenceposition{$id} = $sentenceposition;

        $totalpos{$partofspeech}++;

        if($line =~ m/relation="([a-z]*)"/)
        {
           my($relation);

           $relation = $1;
           $idrelation{$id} = $relation;
        }
        else
        {
           $idrelation{$id} = "none";
        }
     }

     if($line =~ m/id="([0-9]*)"(?:.*)head-id="([0-9]*)"/)
     {
        my($id, $head);

        $id = $1;
        $head = $2;

        $idhead{$id} = $head;
     }

  }
  elsif($line =~ m/<\/sentence/)
  {
     build_trees();
     if($ntokens <= 700)
     {
        get_rewrites();
        $nsentences++;
        $sentencelengths{$tokenposition}++;
```

```
            }
            get_more_features();
            get_tokengrams();
            get_chargrams();
        }
    }
}

sub get_wordlength
{
    my($word) = @_;
    my($length, $feature);
    my(@chars);
    @chars = split(//, $word);
    $length = scalar(@chars);
    $feature = "WORDLENGTH_CHAR_" . $length;
    count_feature($feature);
    $wordlengths{$length}++;
    return($length);
}

sub build_trees
{
    my($wordamount_sentence, $position, $lastleaf, $trees_amount);
    my(%lastnode);

    $wordamount_sentence = scalar (@currentsentence);
    %lastnode = ();

    for($position = 0; $position < $wordamount_sentence; $position++)
    {
        if($currentsentence[$position] =~ m/id="([0-9]*)"/)
        {
            my($id, $key, $last);

            $id = $1;
            $last = 1;

            foreach $key (keys(%idhead))
            {
                if($idhead{$key} == $id)
                {
                    $last = 0
                }
            }

            if($last)
            {
                $lastnode{$id} = 1;
            }
```

```
    }
}

$trees_amount = scalar(keys %lastnode);
foreach $lastleaf (keys %lastnode)
{
    my($postree, $formtree, $lemmatree, $nexthead, $relationtree, $finishedtree);
    my($combitreeposition);

    @postree = (); @formtree = (); @lemtree = (); @reltree = ();
    $postree = $idpartofspeech{$lastleaf};
    $formtree = $idform{$lastleaf};
    $lemmatree = $idlemma{$lastleaf};
    $relationtree = $idrelation{$lastleaf};
    $postree[0] = $idpartofspeech{$lastleaf};
    $formtree[0] = $idform{$lastleaf};
    $lemtree[0] = $idlemma{$lastleaf};
    $reltree[0] = $idrelation{$lastleaf};

    $combitreeposition = 0;
    $finishedtree = 0;

    if(exists $idhead{$lastleaf})
    {
        $nexthead = $idhead{$lastleaf};

        for($position = 0; $position < $wordamount_sentence; $position++)
        {
            if(exists $idhead{$nexthead})
            {
                $combitreeposition++;

                $postree = $idpartofspeech{$nexthead} . "_" . $postree;
                $formtree = $idform{$nexthead} . "_" . $formtree;
                $lemmatree = $idlemma{$nexthead} . "_" . $lemmatree;
                $relationtree = $idrelation{$nexthead} . "_" . $relationtree;

                $postree[$combitreeposition] = $idpartofspeech{$nexthead};
                $formtree[$combitreeposition] = $idform{$nexthead};
                $lemtree[$combitreeposition] = $idlemma{$nexthead};
                $reltree[$combitreeposition] = $idrelation{$nexthead};

                $nexthead = $idhead{$nexthead};
            }
            else
            {
                $combitreeposition++;

                $postree = $idpartofspeech{$nexthead} . "_" . $postree;
                $formtree = $idform{$nexthead} . "_" . $formtree;
```

```
                $lemmatree = $idlemma{$nexthead} . "_" . $lemmatree;
                $relationtree = $idrelation{$nexthead} . "_" . $relationtree;

                $postree[$combitreeposition] = $idpartofspeech{$nexthead};
                $formtree[$combitreeposition] = $idform{$nexthead};
                $lemtree[$combitreeposition] = $idlemma{$nexthead};
                $reltree[$combitreeposition] = $idrelation{$nexthead};

                $finishedtree = 1;
                get_combitreegrams();
                last;
            }
        }
    }
}

sub get_combitreegrams
{
   my($start,$pos,$t1,$t2,$t3, $tg, $skip);

   $skip = "SKIP";

   if(scalar(@postree) >= 3)
   {
      for($start=-1; $start < scalar(@postree); $start++)
      {
         $t1=$start;
         $t2=$start+1;
         $t3=$start+2;
         if(exists $reltree[$t3])
         {
            $tg=sprintf("Tree3_WWW_%s_%s_%s",$formtree[$t1],$formtree[$t2],$formtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WWP_%s_%s_%s",$formtree[$t1],$formtree[$t2],$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WWR_%s_%s_%s",$formtree[$t1],$formtree[$t2],$reltree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WWL_%s_%s_%s",$formtree[$t1],$formtree[$t2],$lemtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WSW_%s_%s_%s",$formtree[$t1],$skip,$formtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WSP_%s_%s_%s",$formtree[$t1],$skip,$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WSR_%s_%s_%s",$formtree[$t1],$skip,$reltree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WSL_%s_%s_%s",$formtree[$t1],$skip,$lemtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WPW_%s_%s_%s",$formtree[$t1],$postree[$t2],$formtree[$t3]);
count_feature($tg);
```

```
            $tg=sprintf("Tree3_WPP_%s_%s_%s",$formtree[$t1],$postree[$t2],$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WPR_%s_%s_%s",$formtree[$t1],$postree[$t2],$reltree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WPL_%s_%s_%s",$formtree[$t1],$postree[$t2],$lemtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WRW_%s_%s_%s",$formtree[$t1],$reltree[$t2],$formtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WRP_%s_%s_%s",$formtree[$t1],$reltree[$t2],$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WRR_%s_%s_%s",$formtree[$t1],$reltree[$t2],$reltree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WRL_%s_%s_%s",$formtree[$t1],$reltree[$t2],$lemtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WLW_%s_%s_%s",$formtree[$t1],$lemtree[$t2],$formtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WLP_%s_%s_%s",$formtree[$t1],$lemtree[$t2],$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WLR_%s_%s_%s",$formtree[$t1],$lemtree[$t2],$reltree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WLL_%s_%s_%s",$formtree[$t1],$lemtree[$t2],$lemtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_PWW_%s_%s_%s",$postree[$t1],$formtree[$t2],$formtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_PWP_%s_%s_%s",$postree[$t1],$formtree[$t2],$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_PWR_%s_%s_%s",$postree[$t1],$formtree[$t2],$reltree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_PWL_%s_%s_%s",$postree[$t1],$formtree[$t2],$lemtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_PPW_%s_%s_%s",$postree[$t1],$postree[$t2],$formtree[$t3]);
count_feature($tg);
          $tg=sprintf("Tree3_PPP_%s_%s_%s",$postree[$t1],$postree[$t2],$postree[$t3]);
count_feature($tg);
          $tg=sprintf("Tree3_PPR_%s_%s_%s",$postree[$t1],$postree[$t2],$reltree[$t3]);
count_feature($tg);
          $tg=sprintf("Tree3_PPL_%s_%s_%s",$postree[$t1],$postree[$t2],$lemtree[$t3]);
count_feature($tg);
          $tg=sprintf("Tree3_PRW_%s_%s_%s",$postree[$t1],$reltree[$t2],$formtree[$t3]);
count_feature($tg);
          $tg=sprintf("Tree3_PRP_%s_%s_%s",$postree[$t1],$reltree[$t2],$postree[$t3]);
count_feature($tg);
          $tg=sprintf("Tree3_PRR_%s_%s_%s",$postree[$t1],$reltree[$t2],$reltree[$t3]);
count_feature($tg);
          $tg=sprintf("Tree3_PRL_%s_%s_%s",$postree[$t1],$reltree[$t2],$lemtree[$t3]);
count_feature($tg);
          $tg=sprintf("Tree3_PLW_%s_%s_%s",$postree[$t1],$lemtree[$t2],$formtree[$t3]);
count_feature($tg);
          $tg=sprintf("Tree3_PLP_%s_%s_%s",$postree[$t1],$lemtree[$t2],$postree[$t3]);
count_feature($tg);
```

```
        $tg=sprintf("Tree3_PLR_%s_%s_%s",$postree[$t1],$lemtree[$t2],$reltree[$t3]);
count_feature($tg);
        $tg=sprintf("Tree3_PLL_%s_%s_%s",$postree[$t1],$lemtree[$t2],$lemtree[$t3]);
count_feature($tg);
        $tg=sprintf("Tree3_PSW_%s_%s_%s",$postree[$t1],$skip,$formtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_PSP_%s_%s_%s",$postree[$t1],$skip,$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_PSR_%s_%s_%s",$postree[$t1],$skip,$reltree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_PSL_%s_%s_%s",$postree[$t1],$skip,$lemtree[$t3]);
count_feature($tg);
        $tg=sprintf("Tree3_RWW_%s_%s_%s",$reltree[$t1],$formtree[$t2],$formtree[$t3]);
count_feature($tg);
        $tg=sprintf("Tree3_RWP_%s_%s_%s",$reltree[$t1],$formtree[$t2],$postree[$t3]);
count_feature($tg);
        $tg=sprintf("Tree3_RWR_%s_%s_%s",$reltree[$t1],$formtree[$t2],$reltree[$t3]);
count_feature($tg);
        $tg=sprintf("Tree3_RWL_%s_%s_%s",$reltree[$t1],$formtree[$t2],$lemtree[$t3]);
count_feature($tg);
        $tg=sprintf("Tree3_RPW_%s_%s_%s",$reltree[$t1],$postree[$t2],$formtree[$t3]);
count_feature($tg);
        $tg=sprintf("Tree3_RPP_%s_%s_%s",$reltree[$t1],$postree[$t2],$postree[$t3]);
count_feature($tg);
        $tg=sprintf("Tree3_RPR_%s_%s_%s",$reltree[$t1],$postree[$t2],$reltree[$t3]);
count_feature($tg);
        $tg=sprintf("Tree3_RPL_%s_%s_%s",$reltree[$t1],$postree[$t2],$lemtree[$t3]);
count_feature($tg);
        $tg=sprintf("Tree3_RRW_%s_%s_%s",$reltree[$t1],$reltree[$t2],$formtree[$t3]);
count_feature($tg);
        $tg=sprintf("Tree3_RRP_%s_%s_%s",$reltree[$t1],$reltree[$t2],$postree[$t3]);
count_feature($tg);
        $tg=sprintf("Tree3_RRR_%s_%s_%s",$reltree[$t1],$reltree[$t2],$reltree[$t3]);
count_feature($tg);
        $tg=sprintf("Tree3_RRL_%s_%s_%s",$reltree[$t1],$reltree[$t2],$lemtree[$t3]);
count_feature($tg);
        $tg=sprintf("Tree3_RLW_%s_%s_%s",$reltree[$t1],$lemtree[$t2],$formtree[$t3]);
count_feature($tg);
        $tg=sprintf("Tree3_RLP_%s_%s_%s",$reltree[$t1],$lemtree[$t2],$postree[$t3]);
count_feature($tg);
        $tg=sprintf("Tree3_RLR_%s_%s_%s",$reltree[$t1],$lemtree[$t2],$reltree[$t3]);
count_feature($tg);
        $tg=sprintf("Tree3_RLL_%s_%s_%s",$reltree[$t1],$lemtree[$t2],$lemtree[$t3]);
count_feature($tg);
        $tg=sprintf("Tree3_RSW_%s_%s_%s",$reltree[$t1],$skip,$formtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_RSP_%s_%s_%s",$reltree[$t1],$skip,$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_RSR_%s_%s_%s",$reltree[$t1],$skip,$reltree[$t3]);
count_feature($tg);
```

```
            $tg=sprintf("Tree3_RSL_%s_%s_%s",$reltree[$t1],$skip,$lemtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LWW_%s_%s_%s",$lemtree[$t1],$formtree[$t2],$formtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LWP_%s_%s_%s",$lemtree[$t1],$formtree[$t2],$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LWR_%s_%s_%s",$lemtree[$t1],$formtree[$t2],$reltree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LWL_%s_%s_%s",$lemtree[$t1],$formtree[$t2],$lemtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LPW_%s_%s_%s",$lemtree[$t1],$postree[$t2],$formtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LPP_%s_%s_%s",$lemtree[$t1],$postree[$t2],$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LPR_%s_%s_%s",$lemtree[$t1],$postree[$t2],$reltree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LPL_%s_%s_%s",$lemtree[$t1],$postree[$t2],$lemtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LRW_%s_%s_%s",$lemtree[$t1],$reltree[$t2],$formtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LRP_%s_%s_%s",$lemtree[$t1],$reltree[$t2],$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LRR_%s_%s_%s",$lemtree[$t1],$reltree[$t2],$reltree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LRL_%s_%s_%s",$lemtree[$t1],$reltree[$t2],$lemtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LLW_%s_%s_%s",$lemtree[$t1],$lemtree[$t2],$formtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LLP_%s_%s_%s",$lemtree[$t1],$lemtree[$t2],$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LLR_%s_%s_%s",$lemtree[$t1],$lemtree[$t2],$reltree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LLL_%s_%s_%s",$lemtree[$t1],$lemtree[$t2],$lemtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LSW_%s_%s_%s",$lemtree[$t1],$skip,$formtree[$t3]);
count_feature($tg);
               $tg=sprintf("Tree3_LSP_%s_%s_%s",$lemtree[$t1],$skip,$postree[$t3]);
count_feature($tg);
               $tg=sprintf("Tree3_LSR_%s_%s_%s",$lemtree[$t1],$skip,$reltree[$t3]);
count_feature($tg);
               $tg=sprintf("Tree3_LSL_%s_%s_%s",$lemtree[$t1],$skip,$lemtree[$t3]);
count_feature($tg);
            }
            elsif(exists $formtree[$t3])
            {
               $tg=sprintf("Tree3_WWW_%s_%s_%s",$formtree[$t1],$formtree[$t2],$formtree[$t3]);
count_feature($tg);
               $tg=sprintf("Tree3_WWP_%s_%s_%s",$formtree[$t1],$formtree[$t2],$postree[$t3]);
count_feature($tg);
```

```
            $tg=sprintf("Tree3_WWL_%s_%s_%s",$formtree[$t1],$formtree[$t2],$lemtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WPW_%s_%s_%s",$formtree[$t1],$postree[$t2],$formtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WPP_%s_%s_%s",$formtree[$t1],$postree[$t2],$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WPL_%s_%s_%s",$formtree[$t1],$postree[$t2],$lemtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WLW_%s_%s_%s",$formtree[$t1],$lemtree[$t2],$formtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WLP_%s_%s_%s",$formtree[$t1],$lemtree[$t2],$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WLL_%s_%s_%s",$formtree[$t1],$lemtree[$t2],$lemtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WSW_%s_%s_%s",$formtree[$t1],$skip,$formtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WSP_%s_%s_%s",$formtree[$t1],$skip,$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_WSL_%s_%s_%s",$formtree[$t1],$skip,$lemtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_PWW_%s_%s_%s",$postree[$t1],$formtree[$t2],$formtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_PWP_%s_%s_%s",$postree[$t1],$formtree[$t2],$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_PWL_%s_%s_%s",$postree[$t1],$formtree[$t2],$lemtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_PPW_%s_%s_%s",$postree[$t1],$postree[$t2],$formtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_PPP_%s_%s_%s",$postree[$t1],$postree[$t2],$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_PPL_%s_%s_%s",$postree[$t1],$postree[$t2],$lemtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_PLW_%s_%s_%s",$postree[$t1],$lemtree[$t2],$formtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_PLP_%s_%s_%s",$postree[$t1],$lemtree[$t2],$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_PLL_%s_%s_%s",$postree[$t1],$lemtree[$t2],$lemtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_PSW_%s_%s_%s",$postree[$t1],$skip,$formtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_PSP_%s_%s_%s",$postree[$t1],$skip,$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_PSL_%s_%s_%s",$postree[$t1],$skip,$lemtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LWW_%s_%s_%s",$lemtree[$t1],$formtree[$t2],$formtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LWP_%s_%s_%s",$lemtree[$t1],$formtree[$t2],$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LWL_%s_%s_%s",$lemtree[$t1],$formtree[$t2],$lemtree[$t3]);
count_feature($tg);
```

```
            $tg=sprintf("Tree3_LPW_%s_%s_%s",$lemtree[$t1],$postree[$t2],$formtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LPP_%s_%s_%s",$lemtree[$t1],$postree[$t2],$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LPL_%s_%s_%s",$lemtree[$t1],$postree[$t2],$lemtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LLW_%s_%s_%s",$lemtree[$t1],$lemtree[$t2],$formtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LLP_%s_%s_%s",$lemtree[$t1],$lemtree[$t2],$postree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LLL_%s_%s_%s",$lemtree[$t1],$lemtree[$t2],$lemtree[$t3]);
count_feature($tg);
            $tg=sprintf("Tree3_LSW_%s_%s_%s",$lemtree[$t1],$skip,$formtree[$t3]);
count_feature($tg);
               $tg=sprintf("Tree3_LSP_%s_%s_%s",$lemtree[$t1],$skip,$postree[$t3]);
count_feature($tg);
               $tg=sprintf("Tree3_LSL_%s_%s_%s",$lemtree[$t1],$skip,$lemtree[$t3]);
count_feature($tg);
            }
         }
      }
      if(scalar(@postree) >= 1)
      {
         for($start = 0; $start < scalar(@postree); $start++)
         {
            $t1 = $start;
            $t2 = $start+1;

            if(exists $reltree[$t2])
            {
               $tg=sprintf("Tree2_WW_%s_%s",$formtree[$t1],$formtree[$t2]); count_feature($tg);
               $tg=sprintf("Tree2_WP_%s_%s",$formtree[$t1],$postree[$t2]); count_feature($tg);
               $tg=sprintf("Tree2_WR_%s_%s",$formtree[$t1],$reltree[$t2]); count_feature($tg);
               $tg=sprintf("Tree2_WL_%s_%s",$formtree[$t1],$lemtree[$t2]); count_feature($tg);
               $tg=sprintf("Tree2_PW_%s_%s",$postree[$t1],$formtree[$t2]); count_feature($tg);
               $tg=sprintf("Tree2_PP_%s_%s",$postree[$t1],$postree[$t2]); count_feature($tg);
               $tg=sprintf("Tree2_PR_%s_%s",$postree[$t1],$reltree[$t2]); count_feature($tg);
               $tg=sprintf("Tree2_PL_%s_%s",$postree[$t1],$lemtree[$t2]); count_feature($tg);
               $tg=sprintf("Tree2_RW_%s_%s",$reltree[$t1],$formtree[$t2]); count_feature($tg);
               $tg=sprintf("Tree2_RP_%s_%s",$reltree[$t1],$postree[$t2]); count_feature($tg);
               $tg=sprintf("Tree2_RR_%s_%s",$reltree[$t1],$reltree[$t2]); count_feature($tg);
               $tg=sprintf("Tree2_RL_%s_%s",$reltree[$t1],$lemtree[$t2]); count_feature($tg);
               $tg=sprintf("Tree2_RW_%s_%s",$lemtree[$t1],$formtree[$t2]); count_feature($tg);
               $tg=sprintf("Tree2_RP_%s_%s",$lemtree[$t1],$postree[$t2]); count_feature($tg);
               $tg=sprintf("Tree2_RR_%s_%s",$lemtree[$t1],$reltree[$t2]); count_feature($tg);
               $tg=sprintf("Tree2_RL_%s_%s",$lemtree[$t1],$lemtree[$t2]); count_feature($tg);
            }
            elsif(exists $formtree[$t2])
            {
               $tg=sprintf("Tree2_WW_%s_%s",$formtree[$t1],$formtree[$t2]); count_feature($tg);
```

```
            $tg=sprintf("Tree2_WP_%s_%s",$formtree[$t1],$postree[$t2]); count_feature($tg);
            $tg=sprintf("Tree2_WL_%s_%s",$formtree[$t1],$lemtree[$t2]); count_feature($tg);
            $tg=sprintf("Tree2_PW_%s_%s",$postree[$t1],$formtree[$t2]); count_feature($tg);
            $tg=sprintf("Tree2_PP_%s_%s",$postree[$t1],$postree[$t2]); count_feature($tg);
            $tg=sprintf("Tree2_PL_%s_%s",$postree[$t1],$lemtree[$t2]); count_feature($tg);
            $tg=sprintf("Tree2_RW_%s_%s",$lemtree[$t1],$formtree[$t2]); count_feature($tg);
            $tg=sprintf("Tree2_RP_%s_%s",$lemtree[$t1],$postree[$t2]); count_feature($tg);
            $tg=sprintf("Tree2_RL_%s_%s",$lemtree[$t1],$lemtree[$t2]); count_feature($tg);
         }
      }
   }
}

sub get_rewrites #alleen eerste 700 woorden, maar wordt gedeeld door ntokens, oplossen
{
   my($id, $head, $rewriteposition, $rewritealph);
   my(%heads, %headdeps);

   foreach $id (keys (%idhead))
   {
      $heads{$idhead{$id}}++;
      $headdeps{$idhead{$id}}{$id}++;
   }

   foreach $head (keys (%headdeps))
   {
      my($dep); my(%deps);
      %deps = ();
      $nrewrites++;
      $rewriteposition = "REWRITE_HEAD_DEPS_LOCATION_" . $idpartofspeech{$head};
      $rewritealph = "REWRITE_HEAD_DEPS_ALPH_" . $idpartofspeech{$head};
      foreach $id (sort (keys (%{$headdeps{$head}}))) #hij sorteert op ids, dus position
      {
         $rewriteposition .= "_" . $idrelation{$id};
         $deps{$idrelation{$id}}++;
      }
      foreach $dep (sort (keys (%deps)))
      {
         my($amount, $position);
         $amount = $deps{$dep};
         for($position = 1; $position <= $amount; $position++)
         {
            $rewritealph .= "_" . $dep;
         }
      }
    #  print TOUT "$rewriteposition\n";
      count_feature($rewriteposition); count_feature($rewritealph);
      $rewrites_position{$rewriteposition}++;
      $rewrites_alph{$rewritealph}++;
   }
```

```
}

sub get_more_features
{
   my($position, $wordamount_sentence);

   $wordamount_sentence = scalar(@currentsentence);

   for($position = 0; $position < $wordamount_sentence; $position++)
   {
      my($id);

      if($currentsentence[$position] =~ m/ id="([0-9]*)"/)
      {
         $id = $1;
      }

      if(exists $idhead{$id} and $sentenceposition{$id} > $sentenceposition{$idhead{$id}})
      {
         $afterhead{$idlemma{$id}}++;
         $afterhead{$idpartofspeech{$id}}++;
      }

      get_unstable_features($id);

      if($idpartofspeech{$id} eq "V-"){ get_verb_features($id); $totalverbs++; }
      elsif($idpartofspeech{$id} eq "R-") { get_preposition_features($id); }
      elsif($idpartofspeech{$id} eq "A-"){ get_adjective_features($id); }
      elsif($idpartofspeech{$id} eq "Nb" or $idpartofspeech{$id} eq "Pp"){ get_noun_features($id);
$totalnouns++; }

      if($idlemma{$id} =~ m/καί$/) { get_coord_features($id); $ncoord++; }

      if((exists $idhead{$id}) and ($sentenceposition{$id} < $sentenceposition{$idhead{$id}}))
      {
         $posheadfinal{$idpartofspeech{$id}}++;
         $formheadfinal{$idform{$id}}++;
         $lemmaheadfinal{$idlemma{$id}}++;
      }
   }
}

sub get_unstable_features
{
   my($id) = @_;

   if($idlemma{$id} eq "ἅπας" or $idlemma{$id} eq "πᾶς" or $idlemma{$id} eq "ἕκαστος")
   {
      if(not exists($unstable{"all"}))
```

```
    {
        $unstable{"all"} = "ἅπας_0_πᾶς_0_ἕκαστος_0_total_0";
    }

    if($unstable{"all"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
    {
        my($amount, $total);

        $amount = $1;
        $amount++;
        $total = $2;
        $total++;
        $unstable{"all"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
        $unstable{"all"} =~ s/total_([0-9]*)/total_$total/;
    }
}
elsif($idlemma{$id} eq "ἐάν" or $idlemma{$id} eq "εἰ")
{
    if(not exists($unstable{"if"}))
    {
        $unstable{"if"} = "ἐάν_0_εἰ_0_total_0";
    }

    if($unstable{"if"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
    {
        my($amount, $total);

        $amount = $1;
        $amount++;
        $total = $2;
        $total++;
        $unstable{"if"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
        $unstable{"if"} =~ s/total_([0-9]*)/total_$total/;
    }
}
elsif($idlemma{$id} eq "ποιέω" or $idlemma{$id} eq "πράσσω")
{
    if(not exists($unstable{"do"}))
    {
        $unstable{"do"} = "ποιέω_0_πράσσω_0_total_0";
    }

    if($unstable{"do"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
    {
        my($amount, $total);

        $amount = $1;
        $amount++;
        $total = $2;
        $total++;
```

```
        $unstable{"do"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
        $unstable{"do"} =~ s/total_([0-9]*)/total_$total/;
    }
}
elsif($idlemma{$id} eq "τρεῖς" or $idlemma{$id} eq "τρία")
{
    if(not exists($unstable{"do"}))
    {
        $unstable{"do"} = "ποιέω_0_πράσσω_0_total_0";
    }

    if($unstable{"do"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
    {
        my($amount, $total);

        $amount = $1;
        $amount++;
        $total = $2;
        $total++;
        $unstable{"do"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
        $unstable{"do"} =~ s/total_([0-9]*)/total_$total/;
    }
}
elsif($idlemma{$id} eq "διό" or $idlemma{$id} eq "οὖν" or $idlemma{$id} eq "οὕτως")
{
    if(not exists($unstable{"do"}))
    {
        $unstable{"do"} = "ποιέω_0_πράσσω_0_total_0";
    }

    if($unstable{"do"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
    {
        my($amount, $total);

        $amount = $1;
        $amount++;
        $total = $2;
        $total++;
        $unstable{"do"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
        $unstable{"do"} =~ s/total_([0-9]*)/total_$total/;
    }
}
elsif($idlemma{$id} eq "δέ" or $idlemma{$id} eq "καί" or $idlemma{$id} eq "ἀλλά")
{
    if(not exists($unstable{"andbutconj"}))
    {
        $unstable{"andbutconj"} = "δέ_0_καί_0_ἀλλά_0_total_0";
    }

    if($unstable{"andbutconj"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
```

```
    {
      my($amount, $total);

      $amount = $1;
      $amount++;
      $total = $2;
      $total++;
      $unstable{"andbutconj"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
      $unstable{"andbutconj"} =~ s/total_([0-9]*)/total_$total/;
    }
  }
  elsif($idlemma{$id} eq "μηδέ" or $idlemma{$id} eq "μήτε")
  {
    if(not exists($unstable{"andnot1"}))
    {
      $unstable{"andnot1"} = "μηδέ_0_μήτε_0_total_0";
    }

    if($unstable{"andnot1"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
    {
      my($amount, $total);

      $amount = $1;
      $amount++;
      $total = $2;
      $total++;
      $unstable{"andnot1"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
      $unstable{"andnot1"} =~ s/total_([0-9]*)/total_$total/;
    }
  }
  elsif($idlemma{$id} =~ m/οὐδέ/ or $idlemma{$id} eq "οὔτε")
  {
    if(not exists($unstable{"andnot2"}))
    {
      $unstable{"andnot2"} = "οὐδέ_0_οὔτε_0_total_0";
    }

    if($unstable{"andnot2"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
    {
      my($amount, $total);

      $amount = $1;
      $amount++;
      $total = $2;
      $total++;
      $unstable{"andnot2"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
      $unstable{"andnot2"} =~ s/total_([0-9]*)/total_$total/;
    }
  }
  elsif($idlemma{$id} eq "οἰκία" or $idlemma{$id} eq "οἶκος")
```

```
{
   if(not exists($unstable{"house"}))
   {
      $unstable{"house"} = "οἰκία_0_οἶκος_0_total_0";
   }

   if($unstable{"house"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
   {
      my($amount, $total);

      $amount = $1;
      $amount++;
      $total = $2;
      $total++;
      $unstable{"house"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
      $unstable{"house"} =~ s/total_([0-9]*)/total_$total/;
   }
}
elsif($idlemma{$id} eq "οἶδα" or $idlemma{$id} eq "γιγνώσκω")
{
   if(not exists($unstable{"know"}))
   {
      $unstable{"know"} = "οἶδα_0_γιγνώσκω_0_total_0";
   }

   if($unstable{"know"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
   {
      my($amount, $total);

      $amount = $1;
      $amount++;
      $total = $2;
      $total++;
      $unstable{"know"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
      $unstable{"know"} =~ s/total_([0-9]*)/total_$total/;
   }
}
elsif($idlemma{$id} eq "διδάσκω" or $idlemma{$id} eq "μανθάνω")
{
   if(not exists($unstable{"learn"}))
   {
      $unstable{"learn"} = "διδάσκω_0_μανθάνω_0_total_0";
   }

   if($unstable{"learn"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
   {
      my($amount, $total);

      $amount = $1;
      $amount++;
```

```
            $total = $2;
            $total++;
            $unstable{"learn"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
            $unstable{"learn"} =~ s/total_([0-9]*)/total_$total/;
        }
    }
    elsif($idlemma{$id} eq "ζωή" or $idlemma{$id} eq "ψυχή")
    {
        if(not exists($unstable{"life"}))
        {
            $unstable{"life"} = "ζωή_0_ψυχή_0_total_0";
        }

        if($unstable{"life"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
        {
            my($amount, $total);

            $amount = $1;
            $amount++;
            $total = $2;
            $total++;
            $unstable{"life"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
            $unstable{"life"} =~ s/total_([0-9]*)/total_$total/;
        }
    }
    elsif($idlemma{$id} eq "ἀγαπάω" or $idlemma{$id} eq "φιλέω")
    {
        if(not exists($unstable{"love"}))
        {
            $unstable{"love"} = "ἀγαπάω_0_φιλέω_0_total_0";
        }

        if($unstable{"love"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
        {
            my($amount, $total);

            $amount = $1;
            $amount++;
            $total = $2;
            $total++;
            $unstable{"love"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
            $unstable{"love"} =~ s/total_([0-9]*)/total_$total/;
        }
    }
    elsif($idlemma{$id} eq "εἰς" or $idlemma{$id} eq "πρός")
    {
        if(not exists($unstable{"into"}))
        {
            $unstable{"into"} = "εἰς_0_πρός_0_total_0";
        }
```

```
    if($unstable{"into"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
    {
        my($amount, $total);

        $amount = $1;
        $amount++;
        $total = $2;
        $total++;
        $unstable{"into"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
        $unstable{"into"} =~ s/total_([0-9]*)/total_$total/;
    }
}
elsif($idlemma{$id} eq "ἄρτι" or $idlemma{$id} eq "νῦν" or $idlemma{$id} eq "νυνί")
{
    if(not exists($unstable{"now"}))
    {
        $unstable{"now"} = "ἄρτι_0_νῦν_0_νυνί_0_total_0";
    }

    if($unstable{"now"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
    {
        my($amount, $total);

        $amount = $1;
        $amount++;
        $total = $2;
        $total++;
        $unstable{"now"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
        $unstable{"now"} =~ s/total_([0-9]*)/total_$total/;
    }
}
elsif($idlemma{$id} eq "δέχομαι" or $idlemma{$id} eq "παραλαμβάνω")
{
    if(not exists($unstable{"receive"}))
    {
        $unstable{"receive"} = "δέχομαι_0_παραλαμβάνω_0_total_0";
    }

    if($unstable{"receive"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
    {
        my($amount, $total);

        $amount = $1;
        $amount++;
        $total = $2;
        $total++;
        $unstable{"receive"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
        $unstable{"receive"} =~ s/total_([0-9]*)/total_$total/;
    }
```

```
}
elsif($idlemma{$id} eq "ἵνα" or $idlemma{$id} eq "ὅπως")
{
   if(not exists($unstable{"sothat"}))
   {
      $unstable{"sothat"} = "ἵνα_0_ὅπως_0_total_0";
   }

   if($unstable{"sothat"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
   {
      my($amount, $total);

      $amount = $1;
      $amount++;
      $total = $2;
      $total++;
      $unstable{"sothat"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
      $unstable{"sothat"} =~ s/total_([0-9]*)/total_$total/;
   }
}
elsif($idlemma{$id} eq "ἀγγέλλω" or $idlemma{$id} eq "κηρύσσώ")
{
   if(not exists($unstable{"proclaim"}))
   {
      $unstable{"proclaim"} = "ἀγγέλλω_0_κηρύσσώ_0_total_0";
   }

   if($unstable{"proclaim"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
   {
      my($amount, $total);

      $amount = $1;
      $amount++;
      $total = $2;
      $total++;
      $unstable{"proclaim"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
      $unstable{"proclaim"} =~ s/total_([0-9]*)/total_$total/;
   }
}
elsif($idlemma{$id} eq "βούλομαι" or $idlemma{$id} eq "θέλω")
{
   if(not exists($unstable{"wish"}))
   {
      $unstable{"wish"} = "βούλομαι_0_θέλω_0_total_0";
   }

   if($unstable{"wish"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
   {
      my($amount, $total);
```

```
            $amount = $1;
            $amount++;
            $total = $2;
            $total++;
            $unstable{"wish"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
            $unstable{"wish"} =~ s/total_([0-9]*)/total_$total/;
        }
    }
    elsif($idlemma{$id} eq "ἀποστέλλω" or $idlemma{$id} eq "πέμπω")
    {
        if(not exists($unstable{"send"}))
        {
            $unstable{"send"} = "ἀποστέλλω_0_πέμπω_0_total_0";
        }

        if($unstable{"send"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
        {
            my($amount, $total);

            $amount = $1;
            $amount++;
            $total = $2;
            $total++;
            $unstable{"send"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
            $unstable{"send"} =~ s/total_([0-9]*)/total_$total/;
        }
    }
    elsif($idlemma{$id} eq "ἴδε" or $idlemma{$id} eq "ἰδού")
    {
        if(not exists($unstable{"behold"}))
        {
            $unstable{"behold"} = "ἴδε_0_ἰδού_0_total_0";
        }

        if($unstable{"behold"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
        {
            my($amount, $total);

            $amount = $1;
            $amount++;
            $total = $2;
            $total++;
            $unstable{"behold"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
            $unstable{"behold"} =~ s/total_([0-9]*)/total_$total/;
        }
    }
    elsif($idlemma{$id} eq "βλέπω" or $idlemma{$id} eq "θεωρέω" or $idlemma{$id} eq "ὁράω")
    {
        if(not exists($unstable{"see"}))
        {
```

```
            $unstable{"see"} = "βλέπω_0_θεωρέω_0_ὁράω_0_total_0";
        }

        if($unstable{"see"} =~ m/$idlemma{$id}_([0-9]*)(?:.*)total_([0-9]*)/)
        {
            my($amount, $total);

            $amount = $1;
            $amount++;
            $total = $2;
            $total++;
            $unstable{"see"} =~ s/$idlemma{$id}_([0-9]*)/$idlemma{$id}_$amount/;
            $unstable{"see"} =~ s/total_([0-9]*)/total_$total/;
        }
    }
}

sub get_verb_features
{
    my($verbid) = @_;
    my($person, $tense, $mood, $voice, $feature, $case);
    my($position, $position2, $position3, $flemma, $fperson, $ftense, $fmood, $fvoice, $fcase);
    my($dep, $article);
    my(@things);

    if($idmorphology{$verbid} =~ m/(..)(.)(.)(.).(.).../)
    {
        $person = $1;
        $tense = $2;
        $mood = $3;
        $voice = $4;
        $case = $5;
    }

    $lemmaverbperson{$idlemma{$verbid}}{$person}++;
    $lemmaverbtense{$idlemma{$verbid}}{$tense}++;
    $lemmaverbmood{$idlemma{$verbid}}{$mood}++;
    $lemmaverbvoice{$idlemma{$verbid}}{$voice}++;
    $lemmaverbrelation{$idlemma{$verbid}}{$idrelation{$verbid}}++;

    $flemma = "L_$idlemma{$verbid}";
    $fperson = "P_$person";
    $ftense = "T_$tense";
    $fmood = "M_$mood";
    $fvoice = "V_$voice";
    $fcase = "C_$case";

    @things = ($fperson, $flemma, $fmood, $ftense, $fvoice, $fcase);

    if(not $mood eq "n" and not $mood eq "p")
```

```
   {
      for($position = 0; $position < 5; $position++)
      {
         $feature = "VERB_" . $things[$position]; count_feature($feature);
         for($position2 = 0; $position2 < 4; $position2++)
         {
            if($position > $position2)
            {
               $feature = "VERB_" . $things[$position] . "_" . $things[$position2];
count_feature($feature);
               for($position3 = 3; $position3 > -1; $position3--)
               {
                  if($position3 < $position2)
                  {
                     $feature = "VERB_" . $things[$position] . "_" . $things[$position2] . "_" .
$things[$position3];
                     count_feature($feature);
                  }
               }
            }
         }
      }
   }
   if($mood eq "n")
   {
      $feature = "VERB_";
      for($position = 1; $position < 4; $position++)
      {
         $feature = "VERB_" . $things[$position]; count_feature($feature);
         for($position2 = 1; $position2 < 3; $position2++)
         {
            if($position > $position2)
            {
               $feature = "VERB_" . $things[$position] . "_" . $things[$position2];
count_feature($feature);
            }
         }
      }
   }
   if($mood eq "p")
   {
      $feature = "VERB_";
      for($position = 1; $position < 5; $position++)
      {
         $feature = "VERB_" . $things[$position]; count_feature($feature);
         for($position2 = 1; $position2 < 4; $position2++)
         {
            if($position < $position2)
            {
```

```
                $feature = "VERB_" . $things[$position] . "_" . $things[$position2];
count_feature($feature);
            }
        }
    }
  }

  if(exists $idrelation{$verbid})
  {
    $verbrelations{$idrelation{$verbid}}++;
  }

  foreach $dep (keys (%idhead))
  {
    foreach $article (keys %idhead)
    {
      if($idhead{$article} == $dep)
      {
        if($idpartofspeech{$article} eq "S-")
        {
          $verbwitharticle{$idlemma{$verbid}}++;
        }
      }
    }

    $feature = "VERB_DEP_POS_" . $idlemma{$verbid} . "_" . $idpartofspeech{$dep};
count_feature($feature);
    $feature = "VERBMOOD_DEP_POS" . $mood . "_" . $idpartofspeech{$dep};
count_feature($feature);
    $feature = "VERBTENSE_DEP_POS" . $tense . "_" . $idpartofspeech{$dep};
count_feature($feature);
  }

  #transitieve werkwoorden
  foreach my$object (keys(%idhead))
  {
    if($idhead{$object} == $verbid and exists $idrelation{$object} and $idrelation{$object} eq
"obj")
    {
      $verblemmawithobject{$idlemma{$verbid}}++;
      $verbformwithobject{$idform{$verbid}}++;
    }
  }

  #hulpwerkwoorden: aux is het hoofd, dus iets is een aux als het een ww als dep heeft
  foreach my$headverb (keys (%idhead))
  {
    my($headperson, $headtense, $headmood, $headvoice, $headcase, $transitive);
    my(%objects);
    if($idhead{$headverb} == $verbid and $idpartofspeech{$headverb} eq "V-")
```

```
      {
         my($feature);
         $totalauxverb++;
        # my(%verbformasaux, %verblemmaasaux, %verbformwithaux, %verblemmawithaux);
         $verbformasaux{$idform{$verbid}}++; $verblemmaasaux{$idlemma{$verbid}}++;
         $verbformwithaux{$idform{$headverb}}++;
$verblemmawithaux{$idlemma{$headverb}}++;
        # $feature = "VERBFORM_AS_AUX_" . $idform{$verbid}; count_feature($feature);
        # $feature = "VERBLEMMA_AS_AUX_" . $idlemma{$verbid}; count_feature($feature);
        # $feature = "VERBFORM_WITH_AUX_" . $idform{$headverb}; count_feature($feature);
        # $feature = "VERBLEMMA_WITH_AUX_" . $idlemma{$headverb};
count_feature($feature);
        # $feature = "AUXRELATION_" . $idrelation{$headverb}; count_feature($feature);
        # if($idmorphology{$headverb} =~ m/(..)(.)(.)(.).(.).../)
         {
            $headperson = $1;
            $headtense = $2;
            $headmood = $3;
            $headvoice = $4;
            $headcase = $5;
         }
         $feature = "VERBTENSE_AS_AUX_" . $tense; count_feature($feature);
         $feature = "VERBTENSE_WITH_AUX_" . $headtense; count_feature($feature);
         $feature = "VERBMOOD_AS_AUX_" . $mood; count_feature($feature);
         $feature = "VERBMOOD_WITH_AUX_" . $headmood; count_feature($feature);
         $feature = "VERBVOICE_AS_AUX_" . $voice; count_feature($feature);
         $feature = "VERBVOICE_WITH_AUX_" . $headvoice; count_feature($feature);
         $transitive = 0;
         foreach my$object (keys (%idhead))
         {
            if($idhead{$object} == $headverb and exists $idrelation{$object} and $idrelation{$object}
eq "obj")
            {
               $transitive = 1;
               $objects{$object} = 1;
               $transauxiliaries{$idlemma{$verbid}}++;
            }
         }
         foreach my$object (keys (%objects))
         {
            $feature = "AUX_OBJ_POS_" . $idpartofspeech{$object}; count_feature($feature);
         }
      }
   }
}

sub get_preposition_features
{
   my($prepid) = @_;
   my($dep, $depid, $depnumber, $depcase, $feature, $article);
```

```
   foreach $dep(keys(%idhead))
   {
      if($idhead{$dep} == $prepid)
      {

         if($idmorphology{$dep} =~ m/.(.)....(.).../)
         {
            $depnumber = $1;
            $depcase = $2;
         }
         else
         {
            $depnumber = "EMPTY";
            $depcase = "NONE";
         }

         $feature = "PP_NUMB_";
         $feature .= $idlemma{$prepid} . "_" . $depnumber; count_feature($feature);
         $feature = "PP_CASE_";
         $feature .= $idlemma{$prepid} . "_" . $depcase; count_feature($feature);
         $feature = "PP_NUMBER_CASE_";
         $feature .= $idlemma{$prepid} . "_" . $depnumber . "_" . $depcase; count_feature($feature);
         $feature = "PP_HEADDEP_LEMMAS_";
         $feature .= $idlemma{$prepid} . "_" . $idlemma{$dep}; count_feature($feature);
         foreach $article (keys %idhead)
         {
            if($idhead{$article} == $dep)
            {
               if($idpartofspeech{$article} eq "S-")
               {
                  $prepositionwitharticle{$idlemma{$prepid}}++;
               }
            }
         }
      }
   }
}

sub get_adjective_features
{
   my($adjid) = @_;
   my($word);

   #samen met article
   foreach $word (keys (%idhead))
   {
      if(exists $idhead{$adjid} and $idhead{$word} == $idhead{$adjid} and $word != $adjid)
      {
         if($idpartofspeech{$word} eq "S-") #is een article
```

```
        {
          $adjarticle{$idlemma{$adjid}}++;
          if($sentenceposition{$adjid} < $sentenceposition{$word})
          {
            $adjbeforearticle{$idlemma{$adjid}}++;
          }
        }
      }
   }

   if(exists $idhead{$adjid} and $idpartofspeech{$idhead{$adjid}} eq "V-")
   {
      my($feature);
      $feature = "ADJ_VERBHEAD_VERB_" . $idlemma{$adjid} . "_" .
$idlemma{$idhead{$adjid}};
      count_feature($feature);
      $feature = "ADJ_WITHVERBHEAD_" . $idlemma{$adjid};
      count_feature($feature);
      $feature = "VERB_ADJDEP_" . $idlemma{$idhead{$adjid}};
      count_feature($feature);
   }
}

sub get_noun_features
{
   my($nounid) = @_;
   my($adjectiveno, $adjective, $article);

  # $totalnouns++;

   #aantal adjectives per noun
   $adjectiveno = 0;
   foreach $adjective(keys(%idhead))
   {
      if($idhead{$adjective} == $nounid and $idpartofspeech{$adjective} eq "A-")
      {
         $adjectiveno++;
      }
      elsif($idhead{$adjective} == $nounid and $idpartofspeech{$adjective} eq "S-")
      {
         $nounformarticle{$idform{$nounid}}++;
         $nounlemmaarticle{$idlemma{$nounid}}++;
      }
   }

   if(not exists $nounsadj{$idlemma{$nounid}})
   {
      $nounsadj{$idlemma{$nounid}} = "nountotal_1_adj_" . $adjectiveno . "_reladj_" .
$adjectiveno;
   }
```

```
   else
   {
      if($nounsadj{$idlemma{$nounid}} =~ m/nountotal_([0-9]*)_adj_([0-9]*)/)
      {
         my($nountotal, $adjectives);
         $nountotal = $1;
         $adjectives = $2;
         $nountotal++;
         $adjectives = $adjectives + $adjectiveno;
         $nounsadj{$idlemma{$nounid}} =~ s/nountotal_[0-9]*_adj_[0-
9]*/nountotal_${nountotal}_adj_${adjectives}/;
      }
   }

   #distributie van getal en naamval over nouns
   if($idmorphology{$nounid} =~ m/.(.)....(.).../)
   {
      my($case, $number);
      $case = $2;
      $number = $1;
      $totalnounscase{$case}++;
      if(not exists $nounscase{$idlemma{$nounid}})
      {
         $nounscase{$idlemma{$nounid}} = "TOTAL_1";
      }
      if($nounscase{$idlemma{$nounid}} =~ m/TOTAL_([0-9]*)(?:.*)${case}_([0-9]*)/)
      {
         my($total, $amount);
         $total = $1;
         $amount = $2;
         $total++;
         $amount++;
         $nounscase{$idlemma{$nounid}} =~ s/${case}_[0-9]*/${case}_$amount/;
         $nounscase{$idlemma{$nounid}} =~ s/TOTAL_[0-9]*/TOTAL_$total/
      }
      elsif($nounscase{$idlemma{$nounid}} =~ m/TOTAL_([0-9]*)/)
      {
         my($total);
         $total = $1;
         $total++;
         $nounscase{$idlemma{$nounid}} =~ s/TOTAL_[0-9]*/TOTAL_${total}_${case}_1/;
      }

      if(exists $idhead{$nounid} and $idhead{$nounid} ne "R-")
      {
         if(not exists $nounscasenopp{$idlemma{$nounid}})
         {
            $nounscasenopp{$idlemma{$nounid}} = "TOTAL_1";
         }
         if($nounscasenopp{$idlemma{$nounid}} =~ m/TOTAL_([0-9]*)(?:.*)${case}_([0-9]*)/)
```

```
        {
           my($total, $amount);
           $total = $1;
           $amount = $2;
           $total++;
           $amount++;
           $nounscasenopp{$idlemma{$nounid}} =~ s/${case}_[0-9]*/${case}_$amount/;
           $nounscasenopp{$idlemma{$nounid}} =~ s/TOTAL_[0-9]*/TOTAL_$total/
        }
        elsif($nounscasenopp{$idlemma{$nounid}} =~ m/TOTAL_([0-9]*)/)
        {
           my($total);
           $total = $1;
           $total++;
           $nounscasenopp{$idlemma{$nounid}} =~ s/TOTAL_[0-
9]*/TOTAL_${total}_${case}_1/;
        }
     }

     if($number eq "s")
     {
        $totalsingular++;
        if(not exists $nounssing{$idlemma{$nounid}})
        {
           $nounssing{$idlemma{$nounid}}++;
        }
     }
  }

  if($idpartofspeech{$nounid} eq "Pp")
  {
     get_pronoun_features($nounid);
  }
}

sub get_pronoun_features
{
  my($pronounid) = @_;
  my($person, $number, $personnumber, $case, $feature);

  $totalpronouns++;

  if($idmorphology{$pronounid} =~ m/(.)(.)....(.).../)
  {
     $person = $1;
     $number = $2;
     $case = $3;
     $personnumber = $person . $number;
  }
  $feature = "PRONOUN_PERSON_" . $person; count_feature($feature);
```

```
   $feature = "PRONOUN_NUMBER_" . $number; count_feature($feature);
   $feature = "PRONOUN_CASE_" . $case; count_feature($feature);
   $pronouns{$personnumber}++;
}

sub get_coord_features
{
   my($kaiid) = @_;
   my($coordinatedelement, $featureposposition, $featureposalpha, $featurelemposition,
$featurelemalpha);
   my($elements);
   my(%pos, %lem);
   %pos = (); %lem = (); $elements = 0;
   $featureposposition = "RATIO_COORD_POS_POSITION"; $featureposalpha =
"RATIO_COORD_POS_ALPH";
   $featurelemposition = "RATIO_COORD_LEM_POSITION"; $featurelemalpha =
"RATIO_COORD_POS_ALPH";
   foreach $coordinatedelement (sort (keys (%idhead)))
   {
      if($idhead{$coordinatedelement} == $kaiid)
      {
         $elements++;
         $featureposposition .= "_" . $idpartofspeech{$coordinatedelement};
         $featurelemposition .= "_" . $idlemma{$coordinatedelement};
         $pos{$idpartofspeech{$coordinatedelement}}++;
         $lem{$idlemma{$coordinatedelement}}++;
      }
   }
   #print TOUT "$elements $featureposposition\n";
   #if($elements == 0){ print TOUT "lineno: $idlineno{$kaiid}\n"; }
   foreach $coordinatedelement (sort (keys (%pos)))
   {
      my($position);
      for($position = 0; $position < $pos{$coordinatedelement}; $position++) { $featureposalpha .=
"_" . $coordinatedelement; }
   }
   foreach $coordinatedelement (sort (keys (%lem)))
   {
      my($position);
      for($position = 0; $position < $lem{$coordinatedelement}; $position++) { $featurelemalpha .=
"_" . $coordinatedelement; }
   }

   if($elements > 0)
   {
      $coordposposition{$featureposposition}++; $coordposalpha{$featureposalpha}++;
      $coordlemposition{$featurelemposition}++; $coordlemalpha{$featurelemalpha}++;
   }
}
```

```
sub get_chargrams #deel nchartotal
{
   my($cgsize,$start,$pos, $cg, $sentencechars); my(@chars);

   #$rawtext=~s/^/ /;
   #$rawtext=~s/ /&/g; #$rawtext=~s/$/ /;
   $rawtext =~ s/ /&/;
   @chars=split(//,$rawtext);
   $nchars += scalar(@chars);
   $sentencechars = scalar(@chars);
   for($cgsize=1;$cgsize<=5;$cgsize++)
   {
      for($start=0;$start<=$sentencechars-$cgsize;$start++)
      {
         $cg="C${cgsize}_";
         for($pos=0;$pos<$cgsize;$pos++)
         {
            $cg.=$chars[$start+$pos];
         }
      count_feature($cg);
   }
  }
}

sub get_tokengrams
{
   my($start, $pos, $t1, $t2, $t3, $tg, $skip);

  $tokwor[0]="#";
  $toklem[0]="#";
  $tokpos[0]="#";
  $tokrel[0]="#";
  $skip = "SKIP";

  for($start=-1;$start<scalar(@tokwor);$start++)
  {
   $t1=$start;
   $t2=$start+1;
   $t3=$start+2;

   if($t1<1 or $t1>scalar(@tokwor)){ $t1=0; }
   if($t2<1 or $t2>scalar(@tokwor)){ $t2=0; }
   if($t3<1 or $t3>scalar(@tokwor)){ $t3=0; }

   $tg=sprintf("T3_WWW_%s_%s_%s",$tokwor[$t1],$tokwor[$t2],$tokwor[$t3]);
count_feature($tg);
   $tg=sprintf("T3_WWP_%s_%s_%s",$tokwor[$t1],$tokwor[$t2],$tokpos[$t3]);
count_feature($tg);
   $tg=sprintf("T3_WWG_%s_%s_%s",$tokwor[$t1],$tokwor[$t2],$tokrel[$t3]); count_feature($tg);
```

```
   $tg=sprintf("T3_WWL_%s_%s_%s",$tokwor[$t1],$tokwor[$t2],$toklem[$t3]);
count_feature($tg);
   $tg=sprintf("T3_WPW_%s_%s_%s",$tokwor[$t1],$tokpos[$t2],$tokwor[$t3]);
count_feature($tg);
   $tg=sprintf("T3_WPP_%s_%s_%s",$tokwor[$t1],$tokpos[$t2],$tokpos[$t3]); count_feature($tg);
   $tg=sprintf("T3_WPG_%s_%s_%s",$tokwor[$t1],$tokpos[$t2],$tokrel[$t3]); count_feature($tg);
   $tg=sprintf("T3_WPL_%s_%s_%s",$tokwor[$t1],$tokpos[$t2],$toklem[$t3]); count_feature($tg);
   $tg=sprintf("T3_WGW_%s_%s_%s",$tokwor[$t1],$tokrel[$t2],$tokwor[$t3]); count_feature($tg);
   $tg=sprintf("T3_WGP_%s_%s_%s",$tokwor[$t1],$tokrel[$t2],$tokpos[$t3]); count_feature($tg);
   $tg=sprintf("T3_WGG_%s_%s_%s",$tokwor[$t1],$tokrel[$t2],$tokrel[$t3]); count_feature($tg);
   $tg=sprintf("T3_WGL_%s_%s_%s",$tokwor[$t1],$tokrel[$t2],$toklem[$t3]); count_feature($tg);
   $tg=sprintf("T3_WLW_%s_%s_%s",$tokwor[$t1],$toklem[$t2],$tokwor[$t3]);
count_feature($tg);
   $tg=sprintf("T3_WLP_%s_%s_%s",$tokwor[$t1],$toklem[$t2],$tokpos[$t3]); count_feature($tg);
   $tg=sprintf("T3_WLG_%s_%s_%s",$tokwor[$t1],$toklem[$t2],$tokrel[$t3]); count_feature($tg);
   $tg=sprintf("T3_WLL_%s_%s_%s",$tokwor[$t1],$toklem[$t2],$toklem[$t3]); count_feature($tg);
   $tg=sprintf("T3_WSW_%s_%s_%s",$tokpos[$t1],$skip,$tokwor[$t3]); count_feature($tg);
   $tg=sprintf("T3_WSP_%s_%s_%s",$tokpos[$t1],$skip,$tokpos[$t3]); count_feature($tg);
   $tg=sprintf("T3_WSG_%s_%s_%s",$tokpos[$t1],$skip,$tokrel[$t3]); count_feature($tg);
   $tg=sprintf("T3_WSL_%s_%s_%s",$tokpos[$t1],$skip,$toklem[$t3]); count_feature($tg);
   $tg=sprintf("T3_WS2W_%s_%s_%s",$tokpos[$t1],$skip,$tokwor[$t3+1]); count_feature($tg);
   $tg=sprintf("T3_WS2P_%s_%s_%s",$tokpos[$t1],$skip,$tokpos[$t3+1]); count_feature($tg);
   $tg=sprintf("T3_WS2G_%s_%s_%s",$tokpos[$t1],$skip,$tokrel[$t3+1]); count_feature($tg);
   $tg=sprintf("T3_WS2L_%s_%s_%s",$tokpos[$t1],$skip,$toklem[$t3+1]); count_feature($tg);
   $tg=sprintf("T3_WS3W_%s_%s_%s",$tokpos[$t1],$skip,$tokwor[$t3+2]); count_feature($tg);
   $tg=sprintf("T3_WS3P_%s_%s_%s",$tokpos[$t1],$skip,$tokpos[$t3+2]); count_feature($tg);
   $tg=sprintf("T3_WS3G_%s_%s_%s",$tokpos[$t1],$skip,$tokrel[$t3+2]); count_feature($tg);
   $tg=sprintf("T3_WS3L_%s_%s_%s",$tokpos[$t1],$skip,$toklem[$t3+2]); count_feature($tg);
   $tg=sprintf("T3_PWW_%s_%s_%s",$tokpos[$t1],$tokwor[$t2],$tokwor[$t3]);
count_feature($tg);
   $tg=sprintf("T3_PWP_%s_%s_%s",$tokpos[$t1],$tokwor[$t2],$tokpos[$t3]); count_feature($tg);
   $tg=sprintf("T3_PWG_%s_%s_%s",$tokpos[$t1],$tokwor[$t2],$tokrel[$t3]); count_feature($tg);
   $tg=sprintf("T3_PWL_%s_%s_%s",$tokpos[$t1],$tokwor[$t2],$toklem[$t3]); count_feature($tg);
   $tg=sprintf("T3_PPW_%s_%s_%s",$tokpos[$t1],$tokpos[$t2],$tokwor[$t3]); count_feature($tg);
   $tg=sprintf("T3_PPP_%s_%s_%s",$tokpos[$t1],$tokpos[$t2],$tokpos[$t3]); count_feature($tg);
   $tg=sprintf("T3_PPG_%s_%s_%s",$tokpos[$t1],$tokpos[$t2],$tokrel[$t3]); count_feature($tg);
   $tg=sprintf("T3_PPL_%s_%s_%s",$tokpos[$t1],$tokpos[$t2],$toklem[$t3]); count_feature($tg);
   $tg=sprintf("T3_PGW_%s_%s_%s",$tokpos[$t1],$tokrel[$t2],$tokwor[$t3]); count_feature($tg);
   $tg=sprintf("T3_PGP_%s_%s_%s",$tokpos[$t1],$tokrel[$t2],$tokpos[$t3]); count_feature($tg);
   $tg=sprintf("T3_PGG_%s_%s_%s",$tokpos[$t1],$tokrel[$t2],$tokrel[$t3]); count_feature($tg);
   $tg=sprintf("T3_PGL_%s_%s_%s",$tokpos[$t1],$tokrel[$t2],$toklem[$t3]); count_feature($tg);
   $tg=sprintf("T3_PLW_%s_%s_%s",$tokpos[$t1],$toklem[$t2],$tokwor[$t3]); count_feature($tg);
   $tg=sprintf("T3_PLP_%s_%s_%s",$tokpos[$t1],$toklem[$t2],$tokpos[$t3]); count_feature($tg);
   $tg=sprintf("T3_PLG_%s_%s_%s",$tokpos[$t1],$toklem[$t2],$tokrel[$t3]); count_feature($tg);
   $tg=sprintf("T3_PLL_%s_%s_%s",$tokpos[$t1],$toklem[$t2],$toklem[$t3]); count_feature($tg);
   $tg=sprintf("T3_PSW_%s_%s_%s",$tokpos[$t1],$skip,$tokwor[$t3]); count_feature($tg);
   $tg=sprintf("T3_PSP_%s_%s_%s",$tokpos[$t1],$skip,$tokpos[$t3]); count_feature($tg);
   $tg=sprintf("T3_PSG_%s_%s_%s",$tokpos[$t1],$skip,$tokrel[$t3]); count_feature($tg);
   $tg=sprintf("T3_PSL_%s_%s_%s",$tokpos[$t1],$skip,$toklem[$t3]); count_feature($tg);
   $tg=sprintf("T3_PS2W_%s_%s_%s",$tokpos[$t1],$skip,$tokwor[$t3+1]); count_feature($tg);
```

```
    $tg=sprintf("T3_PS2P_%s_%s_%s",$tokpos[$t1],$skip,$tokpos[$t3+1]); count_feature($tg);
    $tg=sprintf("T3_PS2G_%s_%s_%s",$tokpos[$t1],$skip,$tokrel[$t3+1]); count_feature($tg);
    $tg=sprintf("T3_PS2L_%s_%s_%s",$tokpos[$t1],$skip,$toklem[$t3+1]); count_feature($tg);
    $tg=sprintf("T3_PS3W_%s_%s_%s",$tokpos[$t1],$skip,$tokwor[$t3+2]); count_feature($tg);
    $tg=sprintf("T3_PS3P_%s_%s_%s",$tokpos[$t1],$skip,$tokpos[$t3+2]); count_feature($tg);
    $tg=sprintf("T3_PS3G_%s_%s_%s",$tokpos[$t1],$skip,$tokrel[$t3+2]); count_feature($tg);
    $tg=sprintf("T3_PS3L_%s_%s_%s",$tokpos[$t1],$skip,$toklem[$t3+2]); count_feature($tg);
    $tg=sprintf("T3_GWW_%s_%s_%s",$tokrel[$t1],$tokwor[$t2],$tokwor[$t3]); count_feature($tg);
    $tg=sprintf("T3_GWP_%s_%s_%s",$tokrel[$t1],$tokwor[$t2],$tokpos[$t3]); count_feature($tg);
    $tg=sprintf("T3_GWG_%s_%s_%s",$tokrel[$t1],$tokwor[$t2],$tokrel[$t3]); count_feature($tg);
    $tg=sprintf("T3_GWL_%s_%s_%s",$tokrel[$t1],$tokwor[$t2],$toklem[$t3]); count_feature($tg);
    $tg=sprintf("T3_GPW_%s_%s_%s",$tokrel[$t1],$tokpos[$t2],$tokwor[$t3]); count_feature($tg);
    $tg=sprintf("T3_GPP_%s_%s_%s",$tokrel[$t1],$tokpos[$t2],$tokpos[$t3]); count_feature($tg);
    $tg=sprintf("T3_GPG_%s_%s_%s",$tokrel[$t1],$tokpos[$t2],$tokrel[$t3]); count_feature($tg);
    $tg=sprintf("T3_GPL_%s_%s_%s",$tokrel[$t1],$tokpos[$t2],$toklem[$t3]); count_feature($tg);
    $tg=sprintf("T3_GGW_%s_%s_%s",$tokrel[$t1],$tokrel[$t2],$tokwor[$t3]); count_feature($tg);
    $tg=sprintf("T3_GGP_%s_%s_%s",$tokrel[$t1],$tokrel[$t2],$tokpos[$t3]); count_feature($tg);
    $tg=sprintf("T3_GGG_%s_%s_%s",$tokrel[$t1],$tokrel[$t2],$tokrel[$t3]); count_feature($tg);
    $tg=sprintf("T3_GGL_%s_%s_%s",$tokrel[$t1],$tokrel[$t2],$toklem[$t3]); count_feature($tg);
    $tg=sprintf("T3_GLW_%s_%s_%s",$tokrel[$t1],$toklem[$t2],$tokwor[$t3]); count_feature($tg);
    $tg=sprintf("T3_GLP_%s_%s_%s",$tokrel[$t1],$toklem[$t2],$tokpos[$t3]); count_feature($tg);
    $tg=sprintf("T3_GLG_%s_%s_%s",$tokrel[$t1],$toklem[$t2],$tokrel[$t3]); count_feature($tg);
    $tg=sprintf("T3_GLL_%s_%s_%s",$tokrel[$t1],$toklem[$t2],$toklem[$t3]); count_feature($tg);
    $tg=sprintf("T3_GSW_%s_%s_%s",$tokpos[$t1],$skip,$tokwor[$t3]); count_feature($tg);
    $tg=sprintf("T3_GSP_%s_%s_%s",$tokpos[$t1],$skip,$tokpos[$t3]); count_feature($tg);
    $tg=sprintf("T3_GSG_%s_%s_%s",$tokpos[$t1],$skip,$tokrel[$t3]); count_feature($tg);
    $tg=sprintf("T3_GSL_%s_%s_%s",$tokpos[$t1],$skip,$toklem[$t3]); count_feature($tg);
    $tg=sprintf("T3_GS2W_%s_%s_%s",$tokpos[$t1],$skip,$tokwor[$t3+1]); count_feature($tg);
    $tg=sprintf("T3_GS2P_%s_%s_%s",$tokpos[$t1],$skip,$tokpos[$t3+1]); count_feature($tg);
    $tg=sprintf("T3_GS2G_%s_%s_%s",$tokpos[$t1],$skip,$tokrel[$t3+1]); count_feature($tg);
    $tg=sprintf("T3_GS2L_%s_%s_%s",$tokpos[$t1],$skip,$toklem[$t3+1]); count_feature($tg);
    $tg=sprintf("T3_GS3W_%s_%s_%s",$tokpos[$t1],$skip,$tokwor[$t3+2]); count_feature($tg);
    $tg=sprintf("T3_GS3P_%s_%s_%s",$tokpos[$t1],$skip,$tokpos[$t3+2]); count_feature($tg);
    $tg=sprintf("T3_GS3G_%s_%s_%s",$tokpos[$t1],$skip,$tokrel[$t3+2]); count_feature($tg);
    $tg=sprintf("T3_GS3L_%s_%s_%s",$tokpos[$t1],$skip,$toklem[$t3+2]); count_feature($tg);
    $tg=sprintf("T3_LWW_%s_%s_%s",$toklem[$t1],$tokwor[$t2],$tokwor[$t3]);
count_feature($tg);
    $tg=sprintf("T3_LWP_%s_%s_%s",$toklem[$t1],$tokwor[$t2],$tokpos[$t3]); count_feature($tg);
    $tg=sprintf("T3_LWG_%s_%s_%s",$toklem[$t1],$tokwor[$t2],$tokrel[$t3]); count_feature($tg);
    $tg=sprintf("T3_LWL_%s_%s_%s",$toklem[$t1],$tokwor[$t2],$toklem[$t3]); count_feature($tg);
    $tg=sprintf("T3_LPW_%s_%s_%s",$toklem[$t1],$tokpos[$t2],$tokwor[$t3]); count_feature($tg);
    $tg=sprintf("T3_LPP_%s_%s_%s",$toklem[$t1],$tokpos[$t2],$tokpos[$t3]); count_feature($tg);
    $tg=sprintf("T3_LPG_%s_%s_%s",$toklem[$t1],$tokpos[$t2],$tokrel[$t3]); count_feature($tg);
    $tg=sprintf("T3_LPL_%s_%s_%s",$toklem[$t1],$tokpos[$t2],$toklem[$t3]); count_feature($tg);
    $tg=sprintf("T3_LGW_%s_%s_%s",$toklem[$t1],$tokrel[$t2],$tokwor[$t3]); count_feature($tg);
    $tg=sprintf("T3_LGP_%s_%s_%s",$toklem[$t1],$tokrel[$t2],$tokpos[$t3]); count_feature($tg);
    $tg=sprintf("T3_LGG_%s_%s_%s",$toklem[$t1],$tokrel[$t2],$tokrel[$t3]); count_feature($tg);
    $tg=sprintf("T3_LGL_%s_%s_%s",$toklem[$t1],$tokrel[$t2],$toklem[$t3]); count_feature($tg);
    $tg=sprintf("T3_LLW_%s_%s_%s",$toklem[$t1],$toklem[$t2],$tokwor[$t3]); count_feature($tg);
    $tg=sprintf("T3_LLP_%s_%s_%s",$toklem[$t1],$toklem[$t2],$tokpos[$t3]); count_feature($tg);
```

```
  $tg=sprintf("T3_LLG_%s_%s_%s",$toklem[$t1],$toklem[$t2],$tokrel[$t3]); count_feature($tg);
  $tg=sprintf("T3_LLL_%s_%s_%s",$toklem[$t1],$toklem[$t2],$toklem[$t3]); count_feature($tg);
  $tg=sprintf("T3_LSW_%s_%s_%s",$tokpos[$t1],$skip,$tokwor[$t3]); count_feature($tg);
  $tg=sprintf("T3_LSP_%s_%s_%s",$tokpos[$t1],$skip,$tokpos[$t3]); count_feature($tg);
  $tg=sprintf("T3_LSG_%s_%s_%s",$tokpos[$t1],$skip,$tokrel[$t3]); count_feature($tg);
  $tg=sprintf("T3_LSL_%s_%s_%s",$tokpos[$t1],$skip,$toklem[$t3]); count_feature($tg);
  $tg=sprintf("T3_LS2W_%s_%s_%s",$tokpos[$t1],$skip,$tokwor[$t3+1]); count_feature($tg);
  $tg=sprintf("T3_LS2P_%s_%s_%s",$tokpos[$t1],$skip,$tokpos[$t3+1]); count_feature($tg);
  $tg=sprintf("T3_LS2G_%s_%s_%s",$tokpos[$t1],$skip,$tokrel[$t3+1]); count_feature($tg);
  $tg=sprintf("T3_LS2L_%s_%s_%s",$tokpos[$t1],$skip,$toklem[$t3+1]); count_feature($tg);
  $tg=sprintf("T3_LS3W_%s_%s_%s",$tokpos[$t1],$skip,$tokwor[$t3+2]); count_feature($tg);
  $tg=sprintf("T3_LS3P_%s_%s_%s",$tokpos[$t1],$skip,$tokpos[$t3+2]); count_feature($tg);
  $tg=sprintf("T3_LS3G_%s_%s_%s",$tokpos[$t1],$skip,$tokrel[$t3+2]); count_feature($tg);
  $tg=sprintf("T3_LS3L_%s_%s_%s",$tokpos[$t1],$skip,$toklem[$t3+2]); count_feature($tg);
}

for($start=0;$start<scalar(@tokwor);$start++)
{
  $t1=$start;
  $t2=$start+1;

  if($t1<1 or $t1>scalar(@tokwor)){ $t1=0; }
  if($t2<1 or $t2>scalar(@tokwor)){ $t2=0; }

  $tg=sprintf("T2_WW_%s_%s",$tokwor[$t1],$tokwor[$t2]); count_feature($tg);
  $tg=sprintf("T2_WP_%s_%s",$tokwor[$t1],$tokpos[$t2]); count_feature($tg);
  $tg=sprintf("T2_WG_%s_%s",$tokwor[$t1],$tokrel[$t2]); count_feature($tg);
  $tg=sprintf("T2_WL_%s_%s",$tokwor[$t1],$toklem[$t2]); count_feature($tg);
  $tg=sprintf("T2_PW_%s_%s",$tokpos[$t1],$tokwor[$t2]); count_feature($tg);
  $tg=sprintf("T2_PP_%s_%s",$tokpos[$t1],$tokpos[$t2]); count_feature($tg);
  $tg=sprintf("T2_PG_%s_%s",$tokpos[$t1],$tokrel[$t2]); count_feature($tg);
  $tg=sprintf("T2_PL_%s_%s",$tokpos[$t1],$toklem[$t2]); count_feature($tg);
  $tg=sprintf("T2_GW_%s_%s",$tokrel[$t1],$tokwor[$t2]); count_feature($tg);
  $tg=sprintf("T2_GP_%s_%s",$tokrel[$t1],$tokpos[$t2]); count_feature($tg);
  $tg=sprintf("T2_GG_%s_%s",$tokrel[$t1],$tokrel[$t2]); count_feature($tg);
  $tg=sprintf("T2_GL_%s_%s",$tokrel[$t1],$toklem[$t2]); count_feature($tg);
  $tg=sprintf("T2_LW_%s_%s",$toklem[$t1],$tokwor[$t2]); count_feature($tg);
  $tg=sprintf("T2_LP_%s_%s",$toklem[$t1],$tokpos[$t2]); count_feature($tg);
  $tg=sprintf("T2_LG_%s_%s",$toklem[$t1],$tokrel[$t2]); count_feature($tg);
  $tg=sprintf("T2_LL_%s_%s",$toklem[$t1],$toklem[$t2]); count_feature($tg);
}

for($start=1;$start<scalar(@tokwor);$start++)
{
  my($ntypes);
  $t1=$start;

  if($t1<1 or $t1>scalar(@tokwor)){ $t1=0; }
  if($t2<1 or $t2>scalar(@tokwor)){ $t2=0; }
```

```
   $tg=sprintf("T1_W_%s",$tokwor[$t1]); count_feature($tg);
   $tg=sprintf("T1_P_%s",$tokpos[$t1]); count_feature($tg);
   $tg=sprintf("T1_G_%s",$tokrel[$t1]); count_feature($tg);
   $tg=sprintf("T1_L_%s",$toklem[$t1]); count_feature ($tg);
  }
}

sub get_richness
{
   my($position);
   @this_richness = ("FORM", "LEMMA", "RWR_LOC", "RWR_ALPH", "SENTENCELENGTH",
"WORDLENGTH");

   if($caseposition == 1)
      {
         print PLOT "INPUT\t";
         for($position = 0; $position < scalar(@this_richness); $position++)
      {
         my$thing = $this_richness[$position];
         my$mttr = $thing . "_TTR"; my$mphpx = $thing . "_PHPX"; my$mnonzipf = $thing .
"_NONZIPF"; my$mentropy = $thing . "_ENTROPY";
         print PLOT "$mttr\t$mphpx\t$mnonzipf\t$mentropy\t";
      }
      print PLOT "\n";
   }

   print PLOT "$inputfile\t";

   for($position = 0; $position < scalar(@this_richness); $position++)
   {
      my($thing, $nthingtot, $kind); my(%thingtype);
      my($ntype, $nhpx, $probability, $entropy, $rank, $totzipf, $freq);
      my($ttr, $phpx, $averzipf, $devzipf, $totdevzipf, $thiszipf, $nonzipf);
      $thing = $this_richness[$position];
      if($thing eq "FORM" or $thing eq "LEMMA")
      {
         $nthingtot = 700;
         if($thing eq "LEMMA"){ %thingtype = %lemmatype700; }
         else{ %thingtype = %formtype700; }
      }
      elsif($thing eq "RWR_LOC" or $thing eq "RWR_ALPH")
      {
         $nthingtot = $nrewrites;
         if($thing eq "RWR_LOC"){ %thingtype = %rewrites_position; }
         else{ %thingtype = %rewrites_alph}
      }
      elsif($thing eq "SENTENCELENGTH"){ $nthingtot = $nsentences; %thingtype =
%sentencelengths; }
      elsif($thing eq "WORDLENGTH"){ $nthingtot = scalar(keys (%formtype700)); %thingtype =
%wordlengths; }
```

```
    $rank = 0; $nhpx = 0; $entropy = 0; $probability = 0; $totzipf = 0;

    foreach $kind (sort ({$thingtype{$b} <=> $thingtype{$a}} (keys (%thingtype))))
    {
       $ntype++;
       if($thingtype{$kind} == 1){ $nhpx++; }
       $probability = $thingtype{$kind} / $nthingtot;
       $entropy -= $probability*log($probability)/log(2);
       $rank++;
       $totzipf += $rank * $thingtype{$kind};
    }

    $ttr = scalar(keys(%thingtype)) / $nthingtot;
    if($thing eq "WORDLENGTH"){ my$feature = "TTR_ALLWORDS_WORDLENGTH";
$feats{$feature} = scalar(keys(%thingtype)) / 700; }
    $phpx = $nhpx / scalar(keys(%thingtype));
    $averzipf = ($totzipf / $nthingtot) / scalar(keys(%thingtype));

    $rank = 0; $totdevzipf = 0;

    foreach $kind (sort ({$thingtype{$b} <=> $thingtype{$a}} (keys(%thingtype))))
    {
       $rank++;
       $thiszipf = $rank * $thingtype{$kind} / $ntype;
       $devzipf = $thiszipf - $averzipf;
       if($devzipf < 0){ $devzipf = -$devzipf }
       $totdevzipf += $devzipf;
    }

    $nonzipf = $totdevzipf / $ntype;
    my($featttr, $featphpx, $featnonzipf, $featentropy);
    $featttr = $thing . "_TTR"; $feats{$featttr} = $ttr;
    $featphpx = $thing . "_PHPX"; $feats{$featphpx} = $phpx;
    $featnonzipf = $thing . "_NONZIPF"; $feats{$featnonzipf} = $nonzipf;
    $featentropy = $thing . "_ENTROPY"; $feats{$featentropy} = $entropy;
    if(100 == 100
      #not $thing eq "WORDLENGTH" and not $thing eq "SENTENCELENGTH"
      ){ print PLOT "$ttr\t$phpx\t$nonzipf\t$entropy\t"; }
    #else{print PLOT "\t\t\t\t$nonzipf\t$entropy\t"; }
  #   print TOUT "$ttr\t$phpx\t$nonzipf\t$entropy\n";
    # print TOUT "$ttr\t$phpx\t$nonzipf\t$entropy\t";
    # print TOUT "$featentropy $entropy\n";
  }

  print PLOT "\n";

  get_partofspeech_richness();
}
```

```
sub get_partofspeech_richness
{
   my($rewrite, $pos, $posrewrite, $rank, $nhpx, $entropy, $probability, $totzipf, $position,
$totdevzipf);
   my($thiszipf, $devzipf, $nonzipf, $ttr, $phpx);
   my(%partsofspeech, %posrew);
   %partsofspeech = (); %posrew = ();
   foreach $rewrite (keys (%rewrites_position))
   {
      if($rewrite =~ m/POSITION_(..)/i)
      {
         $pos = $1;
         $partsofspeech{$pos} += $rewrites_position{$rewrite};
         $posrew{$pos}{$rewrite} += $rewrites_position{$rewrite};
      }
   }
   foreach $pos (keys (%partsofspeech))
   {
      $rank = 0; $nhpx = 0; $entropy = 0; $probability = 0; $totzipf = 0; $ntype = 0;

      foreach $rewrite (sort {$posrew{$pos}{$b} <=> $posrew{$pos}{$a}} (keys
(%{$posrew{$pos}})))
      {
         $ntype++;
        # print TOUT "$partsofspeech{$pos}\n";
       #  print TOUT "$posrew{$pos}{$rewrite}\n";
         if($rewrites_position{$rewrite} == 1){$nhpx++;}
         $probability = $rewrites_position{$rewrite} / $partsofspeech{$pos};
         $entropy -= $probability*log($probability)/log(2);
         $rank++;
         $totzipf += $rank * $posrew{$pos}{$rewrite};
      }

      $ttr = scalar(keys(%{$posrew{$pos}})) / $partsofspeech{$pos};
      $phpx = $nhpx / $partsofspeech{$pos};
      $averzipf = ($totzipf / $partsofspeech{$pos}) / scalar(keys(%{$posrew{$pos}}));
      $rank = 0; $totdevzipf = 0;

      foreach $rewrite (sort {$posrew{$pos}{$b} <=> $posrew{$pos}{$a}} (keys
(%{$posrew{$pos}})))
      {
         $rank++;
         $thiszipf = $rank * $posrew{$pos}{$rewrite} / $partsofspeech{$pos};
   #      print TOUT "$pos $thiszipf\n";
         $devzipf = $thiszipf - $averzipf;
         if($devzipf < 0){ $devzipf = -$devzipf; }
         $totdevzipf += $devzipf;
      }
```

```
      #print TOUT "$pos $ntype\n";

      $nonzipf = $totdevzipf / $partsofspeech{$pos};

      my($featttr, $featphpx, $featnonzipf, $featentropy);
      $featttr = "RWR_LOC_POS_TTR_" . $pos; $feats{$featttr} = $ttr;
      $featphpx = "RWR_LOC_POS_PHPX_" . $pos; $feats{$featphpx} = $phpx;
      $featnonzipf = "RWR_LOC_POS_NONZIPF_" . $pos; $feats{$featnonzipf} = $nonzipf;
      $featentropy = "RWR_LOC_POS_ENTROPY_" . $pos; $feats{$featentropy} = $entropy;
      print TOUT "$pos\tTTR: $ttr\t PHPX: $phpx\t NONZIPF: $nonzipf\t ENTROPY: $entropy\n";
   }

    %partsofspeech = (); %posrew = ();
   foreach $rewrite (keys (%rewrites_alph))
   {
      if($rewrite =~ m/POSITION_(..)/i)
      {
         $pos = $1;
         $partsofspeech{$pos} += $rewrites_alph{$rewrite};
         $posrew{$pos}{$rewrite} += $rewrites_alph{$rewrite};
      }
   }
   foreach $pos (keys (%partsofspeech))
   {
      $rank = 0; $nhpx = 0; $entropy = 0; $probability = 0; $totzipf = 0; $ntype = 0;

      foreach $rewrite (sort {$posrew{$pos}{$b} <=> $posrew{$pos}{$a}} (keys
(%{$posrew{$pos}})))
      {
         $ntype++;
        # print TOUT "$partsofspeech{$pos}\n";
       #  print TOUT "$posrew{$pos}{$rewrite}\n";
         if($rewrites_alph{$rewrite} == 1){$nhpx++;}
         $probability = $rewrites_alph{$rewrite} / $partsofspeech{$pos};
         $entropy -= $probability*log($probability)/log(2);
         $rank++;
         $totzipf += $rank * $posrew{$pos}{$rewrite};
      }

      $ttr = scalar(keys(%{$posrew{$pos}})) / $partsofspeech{$pos};
      $phpx = $nhpx / $partsofspeech{$pos};
      $averzipf = ($totzipf / $partsofspeech{$pos}) / scalar(keys(%{$posrew{$pos}}));
      $rank = 0; $totdevzipf = 0;

      foreach $rewrite (sort {$posrew{$pos}{$b} <=> $posrew{$pos}{$a}} (keys
(%{$posrew{$pos}})))
      {
         $rank++;
         $thiszipf = $rank * $posrew{$pos}{$rewrite} / $partsofspeech{$pos};
    #      print TOUT "$pos $thiszipf\n";
```

```
            $devzipf = $thiszipf - $averzipf;
            if($devzipf < 0){ $devzipf = -$devzipf; }
            $totdevzipf += $devzipf;
        }

        #print TOUT "$pos $ntype\n";

        $nonzipf = $totdevzipf / $partsofspeech{$pos};

        my($featttr, $featphpx, $featnonzipf, $featentropy);
        $featttr = "RWR_ALPH_POS_TTR_POS_" . $pos; $feats{$featttr} = $ttr;
        $featphpx = "RWR_ALPH_POS_PHPX_POS_" . $pos; $feats{$featphpx} = $phpx;
        $featnonzipf = "RWR_ALPH_POS_NONZIPF_POS_" . $pos; $feats{$featnonzipf} = $nonzipf;
        $featentropy = "RWR_ALPH_POS_ENTROPY_" . $pos; $feats{$featentropy} = $entropy;
        print TOUT "$pos\tTTR: $ttr\t PHPX: $phpx\t NONZIPF: $nonzipf\t ENTROPY: $entropy\n";
    }
}

sub finish_features
{
    my($feature, $ratio);

    calculate_verb_features();
    calculate_noun_features();
    calculate_unstable_features();

    foreach my$partofspeech (keys (%posheadfinal))
    {
        if($totalpos{$partofspeech} > 4)
        {
            $ratio = $posheadfinal{$partofspeech} / $totalpos{$partofspeech};
            $feature = "RATIO_POS_HEADFINAL_" . $partofspeech . "&&" . $ratio;
count_feature($feature);
        }
    }

    foreach my$adjective (keys(%adjarticle))
    {
        if($lemmatype{$adjective} > 4)
        {
            $ratio = $adjarticle{$adjective} / $lemmatype{$adjective};
            $feature = "RATIO_ADJ_ART_" . $adjective . "&&" . $ratio; count_feature($feature);
        }
    }

    foreach my$adjective (keys (%adjbeforearticle))
    {
        if($adjarticle{$adjective > 4})
        {
            $ratio = $adjbeforearticle{$adjective} / $adjarticle{$adjective};
```

```
          $feature = "RATIO_ADJ_BEFORE_ART_" . $adjective . "_" . $ratio;
count_feature($feature);
      }
   }

   foreach my$preposition (keys (%prepositionwitharticle))
   {
      if($lemmatype{$preposition} > 4)
      {
        $ratio = $prepositionwitharticle{$preposition} / $lemmatype{$preposition};
        $feature = "RATIO_PREP_WITH_ART_" . $preposition . "_" . $ratio;
count_feature($feature);
      }
   }

   foreach $thing (keys (%afterhead))
   {
      if($thing =~ m/[\p{Greek} ’]*/ and $lemmatype{$thing} > 4)
      {
         $ratio = $afterhead{$thing} / $lemmatype{$thing};
         $feature = "RATIO_AFTER_HEAD_TYPE_" . $thing . "&&" . $ratio;
count_feature($feature);
      }
      elsif($totalpos{$thing} > 4)
      {
         $ratio = $afterhead{$thing} / $totalpos{$thing};
         $feature = "RATIO_AFTER_HEAD_POS_" . $thing . "&&" . $ratio; count_feature($feature);
      }
   }

   foreach my$coordination (keys (%coordposposition))
   {
      $ratio = $coordposposition{$coordination} / $ncoord;
      $feature = $coordination . "&&" . $ratio; count_feature($feature);
     # print TOUT "$feature\n";
   }
   foreach my$coordination (keys (%coordlemposition))
   {
      $ratio = $coordposposition{$coordination} / $ncoord;
      $feature = $coordination . "&&" . $ratio; count_feature($feature);
      #print TOUT "$feature\n";
   }
   foreach my$coordination (keys (%coordposalpha))
   {
      $ratio = $coordposposition{$coordination} / $ncoord;
      $feature = $coordination . "&&" . $ratio; count_feature($feature);
      #print TOUT "$feature\n";
   }
   foreach my$coordination (keys (%coordlemalpha))
   {
```

```
        $ratio = $coordposposition{$coordination} / $ncoord;
        $feature = $coordination . "&&" . $ratio; count_feature($feature);
        #print TOUT "$feature\n";
    }
}

sub calculate_verb_features
{
    my($feature, $ratio);

    foreach my$relation (keys(%verbrelations))
    {
        $ratio = $verbrelations{$relation} / $totalverbs;
        $feature = "RATIO_VERB_RELAT_" . $relation . "&&" . $ratio; count_feature($feature);
    }

    foreach my$transverb (keys (%verblemmawithobject))
    {
        if($lemmatype{$transverb} > 4)
        {
            $ratio = $verblemmawithobject{$transverb} / $lemmatype{$transverb};
            $feature = "RATIO_VERB_LEMMA_TRANS_" . $transverb . "&&" . $ratio;
count_feature($feature);
        }
    }

    foreach my$transverb (keys (%verbformwithobject))
    {
        if($formtype{$transverb} > 4)
        {
            $ratio = $verbformwithobject{$transverb} / $formtype{$transverb};
            $feature = "RATIO_VERB_FORM_TRANS_" . $transverb . "&&" . $ratio;
count_feature($feature);
        }
    }

    foreach my$auxverb (keys (%verbformasaux))
    {
        if($formtype{$auxverb} > 4)
        {
            $ratio = $verbformasaux{$auxverb} / $totalauxverb;
            $feature = "RATIO_VERBFORM_AS_AUX_" . $auxverb; count_feature($feature);
        }
    }

    foreach my$auxverb (keys (%verblemmaasaux))
    {
        if($lemmatype{$auxverb} > 4)
        {
            $ratio = $verblemmaasaux{$auxverb} / $totalauxverb;
```

```
            $feature = "RATIO_VERBLEMMA_AS_AUX_" . $auxverb; count_feature($feature);
        }
    }

    foreach my$headverb (keys (%verbformwithaux))
    {
        if($formtype{$headverb} > 4)
        {
            $ratio = $verbformwithaux{$headverb} / $totalauxverb;
            $feature = "RATIO_VERBFORM_WITH_AUX_" . $headverb; count_feature($feature);
        }
    }

    foreach my$headverb (keys (%verblemmawithaux))
    {
        if($lemmatype{$headverb} > 4)
        {
            $ratio = $verblemmawithaux{$headverb} / $totalauxverb;
            $feature = "RATIO_VERBLEMMA_WITH_AUX_" . $headverb; count_feature($feature);
        }
    }


}

sub calculate_noun_features #dingen die minstens 4x voorkomen
{
    my($noun, $feature, $ratio, $nountotal, $adjtotal, $adjrel);

    #adjectives per noun
    foreach $noun (keys(%nounsadj))
    {
        if($nounsadj{$noun} =~ m/nountotal_([0-9]*)_adj_([0-9]*)/)
        {
            $nountotal = $1;
            $adjtotal = $2;
            $ratio = $adjtotal / $nountotal;
            $feature = "RATIO_ADJPERNOUN_" . $noun . "&&" . $ratio; count_feature($feature);
        }
    }

    $ratio = $totalpos{"Nb"}/$totalpos{"A-"};
    $feature = "RATIO_NOUNS_ADJ&&" . $ratio; count_feature($feature);

    #gem aantal articles per noun
    foreach $noun (keys (%nounformarticle))
    {
        if($formtype{$noun} > 4)
        {
            $ratio = $nounformarticle{$noun} / $formtype{$noun};
```

```
            $feature = "RATIO_NOUN_FORM_ART_" . $noun . "&&" . $ratio; count_feature($feature);
        }

    }

    foreach $noun (keys (%nounlemmaarticle))
    {
        if($lemmatype{$noun} > 4)
        {
            $ratio = $nounlemmaarticle{$noun} / $lemmatype{$noun};
            $feature = "RATIO_NOUN_LEMMA_ART_" . $noun . "_" . $ratio; count_feature($feature);
        }
    }

    foreach $noun (keys(%nounscase))
    {
        if($lemmatype{$noun} > 4)
        {
            my($position, $total);
            my(@cases);
            if($nounscase{$noun} =~ m/TOTAL_([0-9]*)/){$total = $1;}
            @cases = split(/_/, $nounscase{$noun});
            for($position = 2; $position < scalar(@cases); $position = $position+2)
            {
                my($relcase);
                $relcase = $cases[$position+1] / $total;
                $feature = "RATIO_CASEPERNOUN_" . $noun . "_" . $cases[$position] . "&&" .
$relcase; count_feature($feature);
            }
        }
    }

    foreach $noun (keys(%nounscasenopp))
    {
        my($position, $total);
        my(@cases);
        if($nounscasenopp{$noun} =~ m/TOTAL_([0-9]*)/)
        {
            $total = $1;
        }
        if($total > 4)
        {
            @cases = split(/_/, $nounscasenopp{$noun});
            for($position = 2; $position < scalar(@cases); $position = $position+2)
            {
                my($relcase);
                $relcase = $cases[$position+1] / $total;
                $feature = "RATIO_CASEPERNOUN_EXCLPP_" . $noun . "_" . $cases[$position] .
"&&" . $relcase; count_feature($feature);
            }
```

```
        }
    }

    foreach my$case (keys (%totalnounscase))
    {
        $ratio = $totalnounscase{$case} / ($totalpos{"Nb"} + $totalpos{"Pp"});
        $feature = "RATIO_NOUNCASE_" . $case . "&&" . $ratio; count_feature($feature);
    }

    foreach $noun (keys (%nounssing))
    {
        if($lemmatype{$noun} > 4)
        {
            $ratio = $nounssing{$noun} / $totalnouns;
            $feature = "RATIO_SING_NOUN_" . $noun . "&&" . $ratio; count_feature($feature);
        }
    }

    $ratio = $totalsingular / $totalnouns;
    $feature = "RATIO_SING&&" . $ratio; count_feature($feature);

    $feature = "RATIO_PRONOUN_TOTAL&&";
    $ratio = $totalpronouns / $ntokens;
    $feature .= $ratio; count_feature($feature);

    foreach my$pronoun (keys (%pronouns))
    {
        $feature = "RATIO_PRONOUN_" . $pronoun . "&&";
        $ratio = $pronouns{$pronoun} / $totalpronouns;
        $feature .= $ratio; count_feature($feature);
    }
}

sub calculate_unstable_features #dingen die minstens 4x voorkomen
{
    my($case);
    foreach $case (keys (%unstable))
    {
        my(@words);
        my($position, $total);
        @words = split(/_/, $unstable{$case});

        $total = $words[$#words];

        for($position = 0; $position < (scalar(@words) - 2); $position = $position+2)
        {
            my($feature);
            if($total > 4)
            {
                $words[$position+1] = $words[$position+1]/$total;
```

```
            $feature = "RATIO_UNSTABLE_" . $case . "_" . $words[$position] . "&&" .
$words[$position+1]; count_feature($feature);
          }
       }
   }
}

sub count_feature
{
   my ($feat) = @_;

   if($feat =~ m/RATIO_.*&&([0-9]\.*[0-9]*)/)
   {
      my$ratio = $1;
      $feat =~ s/&&([0-9]\.*[0-9]*)//;
      $feats{$feat} = $ratio;
      $monsterfeats{$feat} = $ratio;
   }
   elsif($feat =~m/RATIO_.*&&1/)
   {
      $feat =~ s/&&1//; $feats{$feat} = 1; $monsterfeats{$feat} = 1;
   }
   elsif($feat =~m/RATIO_.*&&0/)
   {
      $feat =~ s/&&0//; $feats{$feat} = 0; $monsterfeats{$feat} = 0;
   }
   else
   {
      $feats{$feat}++;
      $monsterfeats{$feat}++
   }

   #kijk soort feature:
}

sub report_features
{
   my($feature, $count);

   print MASTEROUT "$inputfile\t";
   $count = 0;
   #if($caseposition < 14)
   #{
   #   print MASTEROUT "Paul\t";
   #}
   #else
   #{
   #   print MASTEROUT "Notpaul\t";
   #}
   foreach $feature (sort({$feats{$b} <=> $feats{$a}} (keys(%feats))))
```

```
    {
        $count++;
        if($feature =~ m/^C[0-9]_/)
        {
            $feats{$feature} = $feats{$feature} / $nchars;
        }
        elsif($feature =~ m/REWRITE_/)
        {
            $feats{$feature} = $feats{$feature} / 700;
        }
        elsif(not $feature =~ m/RATIO/)
        {
            $feats{$feature} = $feats{$feature} / $ntokens;
        }

        print OUT "$feature $feats{$feature}\n";
    }

    #print MASTEROUT "\n";
    print "Aantal features in $inputfile : $count\n";

    if($caseposition == 22)
    {
        my$scalar = scalar(keys(%monsterfeats));
        print "Totaal aantal features: $scalar";
    }
}
```

Appendix B: list unstable features

Everything	ἅπας	πᾶς	
If	ἐάν	^εἰ$	
To do, make	ποιέω	πράσσω	
Three	τρεῖς	τρία	
So	διό	οὖν	οὕτως
But, and (more general: conjunction)	δέ	καί	ἀλλά
And not	μηδέ	μήτε	
And not	οὐδέ	οὔτε	
House	οἰκία	οἶκος	
To know, understand	οἶδα	γιγνώσκω	
Small	μικρός	ὀλίγος	
Life	ζωή	ψυχή	
Learn	διδάσκω	μανθάνω	
Love	ἀγαπάω	φιλέω	

To	εἰς	πρός	
Now	ἄρτι	νῦν	νυνί
To receive	δέχομαι	παραλαμβάνω	
In order to	ἵνα	ὅπως	
To bring/send a message	ἀγγέλλω	κηρύσσώ	
To want	βούλομαι	θέλω	
To send	ἀποστέλλω	πέμπω	
Behold	ἴδε	ἰδού	
To see, watch	βλέπω	θεωρέω	ὁράω