university of groningen / faculty of science and engineering / mathematics and applied mathematics

Computational Stemmatology: Reconstructing Text Phylogenies through Computer Assisted Methods

Master Research Project Applied Mathematics

Master in Applied Mathematics

12th July 2024

First Supervisor: Dr J. Koellermeier

Second Supervisor: Dr T. M. Tashu

Student: Darren Zammit s5284236

Abstract

Stemmatology is a branch of textual criticism whose aim is to reconstruct the history, or stemma, of a text given a collection of manuscripts imperfectly copied from each other. Typically, a philologist would have to read through all the manuscripts several times and from the small differences in between the texts, the stemma would be constructed. This can be a very long process since dates and textual features which expose the documents' evolutionary direction are not always available or reliable and thus in this project, a tool chain was developed to derive a preliminary stemma using computational tools. This project serves to describe a general framework for reliably reconstructing stemmata and some experiments involving synthetic data sets and corpora with known stemmata, to which a number of computer-assisted methods from phylogenetics and natural language processing were applied. Moreover, we applied these methods to a tradition whose stemma is unknown, provided to us by a philologist, leading to believable results.

Contents

List of Tables

List of Figures

Acknowledgements

I thank my supervisors Dr Julian Koellermeier and Dr Tsegaye Tashu who allowed me to work independently throughout the project as well as for their suggestions. This was quite a unique project and I consider myself to have been very lucky to have worked on it.

I am also grateful to Lois Bakker and Dr Andrew Irving for providing me with a digital version of the Phareta Fidei tradition, expecially Lois who spent weeks going through the nearly illegible manuscripts and typing them down into a computer readable format.

I am grateful to my friends and family who always encouraged me to push myself forward and especially all those who belayed me whilst climbing or held up a pad during martial arts classes.

The research work disclosed in this publication is partially funded by the Endeavour II Scholarships Scheme. The project is co-funded by the ESF+ 2021-2027

1 Introduction to Computational Stemmatology, Graphs and Phylogenetics

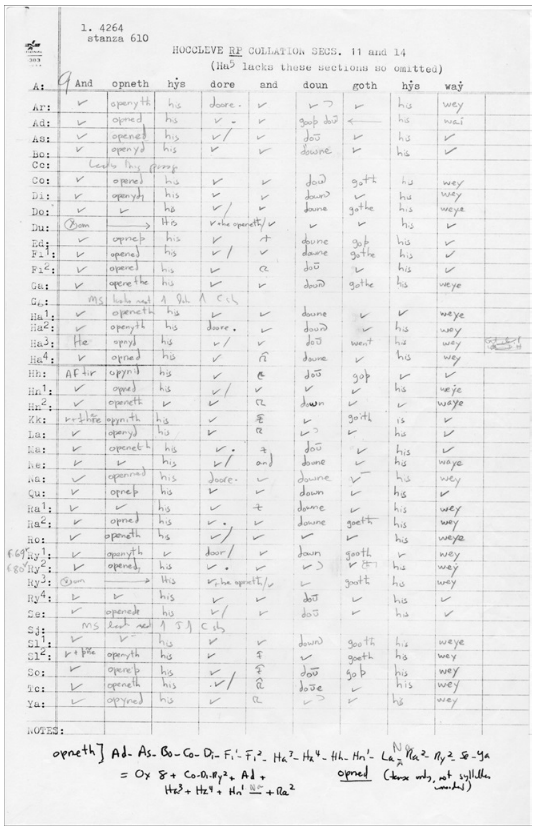
This section starts off as an introduction to stemmatology as a discipline. We then introduce some graph theoretical concepts and how they can be applied for phylogenetics and thus stemmatology. Finally, we discuss the main goals of this project.

1.1 Stemmatology

Copying a text multiple times over long periods of time usually results in changes in its content. Stemmatology [1] is a methodical approach to textual criticism and is based on the assumption that if two or more surviving witnesses have common errors then they were likely derived from a common intermediate ancestor (hyparchetype). Relations between the lost ancestors are determined similarly and such predicted ancestors are placed in with all surviving witnesses in a family tree or "stemma" descended from a single although not necessarily known common ancestor or "archetype". Given a set of manuscripts such as in Figure 1, the aim is to derive the order in which the manuscripts were copied from each other.

Figure 1: Some textual witnesses of Phareta Fidei tradition which is further discussed in section 3.2.

One way of deriving a stemma is to compare the scripts by aligning them word by word in a process called 'collation' as shown in Figure 2. This way it is fairly easy to analyse changes in vocabulary, from which the stemma may be constructed.



l. 4264
stanza 610

HOCCLEVE RP COLLATION SECS. 11 and 14
(Ha5 lacks these sections so omitted)

A:	And	opneth	hẏs	dore	and	doun	goth	hẏs	waẏ
Ar:	✓	openyth	his	doore.	✓	✓	✓	his	wey
Ad:	✓	opned	his	✓ .	✓	goþ doū	←	his	wai
As:	✓	opened	his	✓/	✓	doū	✓	his	✓
Bo:	✓	openyd	his	✓	✓	downe	✓	his	✓
Cc:	lacks this passage								
Co:	✓	opened	his	✓	✓	doun	goth	his	wey
Di:	✓	openyth	his	✓	✓	down	✓	his	wey
Do:	✓	✓	his	✓/	✓	doune	gothe	his	weye
Du:	om	→	His	✓ + the openeth/	✓	✓	✓	his	✓
Ed:	✓	opneþ	his	✓	+	doune	goþ	his	✓
Fi1:	✓	opened	his	✓/	✓	downe	gothe	his	✓
Fi2:	✓	opened	his	✓	&	doū	✓	his	✓
Ga:	✓	opene the	his	✓	✓	doun	gothe	his	weye
Gg:	MS lacks next 1 Dh 1 Csh								
Ha1:	✓	openeth	his	✓	✓	doune	✓	✓	weye
Ha2:	✓	openyth	his	doore.	✓	doun	✓	his	wey
Ha3:	He	opnyd	his	✓/	✓	doū	went	his	wey
Ha4:	✓	opned	his	✓	&	doune	✓	his	wey
Hh:	Aftir	opynid	his	✓	&	doū	goþ	✓	✓
Hn1:	✓	opned	his	✓/	✓	✓	✓	his	weye
Hn2:	✓	openeth	✓	✓	&	down	✓	✓	waye
Kk:	✓ + there	opynith	his	✓	&	✓	goith	is	✓
La:	✓	openyd	his	✓	&	✓	✓	his	✓
Ma:	✓	openeth	his	✓ .	&	doū	✓	his	✓
Ne:	✓	✓	his	✓/	and	doune	✓	his	waye
Na:	✓	openned	his	doore.	✓	downe	✓	his	wey
Qu:	✓	opneþ	his	✓	✓	down	✓	his	✓
Ra1:	✓	✓	his	✓	&	downe	✓	his	wey
Ra2:	✓	opned	his	✓ .	✓	downe	goeth	his	wey
Ro:	✓	openeth	his	✓/	✓	✓	✓	his	weye
f.69r Ry1:	✓	openyth	✓	door/	✓	down	gooth	✓	wey
f.80v Ry2:	✓	opened	his	✓ .	✓	✓	✓	his	wey
Ry3:	om	→	His	✓ + the opneth/	✓	✓	gooth	his	wey
Ry4:	✓	✓	his	✓	✓	doū	✓	his	✓
Se:	✓	openede	his	✓/	✓	doū	✓	his	✓
Sj:	MS lacks next 1 SJ Csh								
Sl1:	✓	✓	his	✓	✓	down	gooth	his	weye
Sl2:	✓ + þere	openyth	his	✓	&	✓	goeth	his	wey
So:	✓	openeþ	his	✓/	&	doū	goþ	his	wey
Tc:	✓	openeth	his	✓/	&	doūe	✓	his	wey
Ya:	✓	opyned	his	✓	&	✓	✓	his	wey

NOTES:

opneth] Ad- As- Bo- Co- Di- Fi1- Fi2- Ha3- Ha4- Hh- Hn1- La Na Ra2- Ry2- Se- Ya
= Ox 8 + Co-Di-Ry2 + Ad + Ha3 + Ha4 + Hn1 Na + Ra2
opned (tense only, not syllables counted)

Figure 2: Example of a tabular collation: Thomas Hoccleve's Regiment of Princes, line 4264. Hoccleve Archive, University of Texas Libraries. Extracted from [1].

Once the stemma is derived, the text of the archetype is constructed by examining variants from the hyparchetypes closest to the archetype and selecting the most likely ones. This phase is called selection. The most common variant at each locus (position in the script) at the same level of the tree, is selected as the most likely variant. If multiple variants occur equally often, then the philologist must use their judgement to determine which makes the most sense. A common example is that scribes tend to replace more difficult and older words with simpler and newer words, thus whilst constructing the archetype, the philologist typically would favour the more complex and sometimes older words that became less often used over time.

The most informative changes or copying errors include exchanging a word with a synonym, skipping words or sentences (usually by accident), paraphrasing of sentences and abbreviating names. Spelling mistakes, changes in punctuation, doubling of sentences and words are also quite common however, they are far less informative as the next scribe is very likely to fix these mistakes.

The text must then undergo the process of "examination" during which the philologist must identify errors as in some loci it may be that none of the surviving variants preserve the original text. If the text is determined to have significant amounts of missing text, it must be corrected through "emendation". At the end of this process it is sometimes possible to reconstruct a text closer to the original than any of the surviving witnesses.

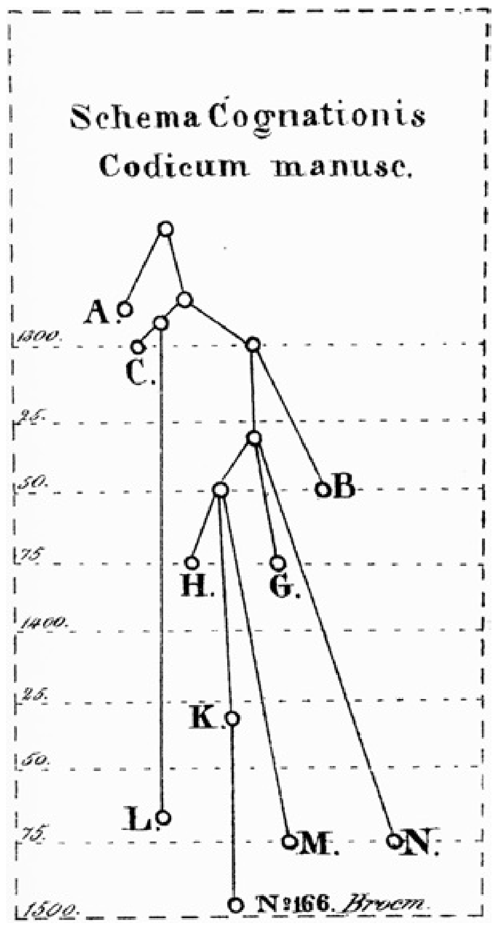


Figure 3: Schlyter's schema cognationis for the Västgötalagen (Schlyter and Collin 1827, appendix), which may be the first printed stemma. Extracted from [1]

The process of deriving a stemma has been studied in mathematical terms [1] since a stemma is essentially a graph as in Figure 3. However, some of the crucial steps in the process are largely impregnable to algorithmic analysis. Usually, familiarising oneself with a text and its contexts is instrumental in understanding its transmission which can be very time-consuming since the philologist often starts knowing very little about the original text or its author(s), their writing style, the environment where the text was written or even the time-period it was written in. This information plays a crucial role in determining the direction of copying and thus finding the root of the stemma. With each successive analysis of the scripts, the philologist identifies more subtleties. However, manual stemmatology can still lead to the philologist to mistakenly see patterns where there are none since every collection of manuscripts is unique and may follow very different and complicated patterns which humans are typically not well suited to fully understand. This non-linearity makes the process difficult to program fully, however, some steps in the process have been computerised to save time for the philologist or derive a hypothetical stemma when one deals with impregnable or very numerous scripts.

1.2 Computational Stemmatology

There exists a wide variety of approaches in dealing with textual and stemmatic reconstruction and an equally wide variety of tools are available [1]. Computer tools have simplified many steps in text editing. Moreover, computer simulations allow for the study of the evolution of large volumes of textual data. On top of this, computational stemmatology has potential applications in plagiarism detection [2], analysing the evolution of computer viruses [3] and content-based social network analysis [4] since any algorithm used in computational stemmatology in some way relies on comparing scripts.

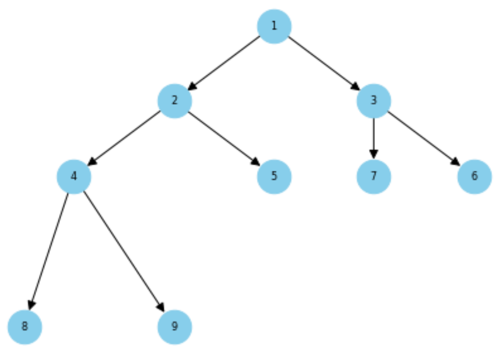
Graphical modelling takes a central role in constructing stemmata [1]. Contemporary methods – which are primarily analogous to those in phylogenetics, the study of the evolutionary history of species – can be split into distance-based [5], parsimony-based [6], and statistical methods [7]. In these methods every word in the text is equally important. In reality there are many different kinds of errors and changes, some more significant than others, leading to the algorithm failing to faithfully construct a stemma. For example, spelling mistakes or changing symbols should have far less impact on a script than adding or redacting text. Another issue which these methods struggle to address is that of contamination, that is when a text is copied from multiple manuscripts [1]. If different scripts were used for different chapters then this could lead to outputs which faithfully reflect the stemma of most chapters but not all if different chapters were copied from different manuscripts. Phylogenetic methods are usually only capable of generating bifurcating trees which are incapable of explaining contaminated stemmata.

Since the seminal paper [8] which first applied computational phylogenetics models in stemmatology some advances in computational stemmatology have been made. However, most papers in computational stemmatology are authored by computer scientists and philologists with mathematicians and statisticians representing a small minority of authors [1]. Moreover, algorithms designed specifically for stemmatology, even ones whose framework is adapted for stemmatolology, are very few in number [1]. Computational methods usually rely on transforming the data into analogues of genetic data and feeding it directly into a phylogenetics program as though it were phylogenetic data, with very few algorithms developed exclusively for stemmatology.

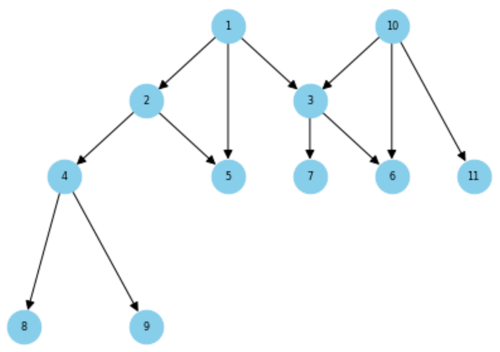
One area of computational stemmatology which has not been very well studied is the rooting of a stemma. So far most of the work done in computational stemmatology has been focused on generating the stemma. Rooting, however, is almost always done manually. This is due to the fact that whilst rooting a stemma, the philologist relies on context such as the historical era, the age of words, changes in spelling etc. Computerised methods are incapable of identifying without highly detailed datasets about such changes, which to the author's knowledge do not exist.

1.3 Graph Theory in Stemmatology and Phylogenetics

Approaches to formalising stemmata rely on graph theory as a framework. A graph G is a pair of a set of vertices (sometimes called nodes) V and a set of edges connecting the vertices E. Vertices represent entities, in the case of stemmata the variants and in phylogenetics the species, whereas edges represent relationships between the entities. A graph is said to be undirected if it consists of a set of vertices and a set of undirected edges, and directed if it consists of a set of nodes and a set of directed edges (ordered pairs of vertices). The two vertices forming an edge are said to be the edge's endpoints. An edge connecting vertices A and B can be denoted (A, B), in which case the vertices A and B are said to be adjacent. The neighbourhood of vertex A is the subgraph formed by all of its adjacent vertices. The degree of a vertex is the number of edges connecting to it. A vertex is called a leaf if it has degree 1. A graph is said to be connected if there exists a path (a sequence of edges which joins a sequence of vertices) between any two vertices. A cycle is a path of vertices and edges in which a vertex is both the start and the end of the path. A directed acyclic graph (DAG) is a directed graph with no directed cycles consisting of vertices and edges with each edge directed from one vertex to another in such a manner that it is impossible to start at any vertex and follow any sequence of edges looping back to the starting vertex as can be see in Figure 4 (b). A tree is an undirected graph in which any pair of vertices are connected by only one path. A polytree is a DAG whose undirected graph is a tree. In the absence of contamination, the stemma usually takes the form of a polytree [1]. Trees are useful in describing the relations between objects which evolve without interfering with one another. In biology, the leaves are usually extant species or groups thereof, whereas in stemmatology the leaves represent extant witnesses of a text. Most current phylogenetic models produce strictly bifurcating or "binary" trees wherein each vertex has at most two descendants as in Figure 4 (a).



(a) An bifurcating tree.



(b) A direct acyclic graph.

Figure 4: The main types of graphs in stemmatology and phylogenetics. (b) is a general stemma with more than one root, contamination and more than two children per node.

A rooted tree is a tree in which one vertex is taken as the root. The edges of such a tree can be directed, either all away from or all towards the root. In phylogenetics, unrooted trees or networks can only describe the relations between the leaves and their ancestries. To transform the unrooted tree into a rooted tree, one must determine the direction of change. The root of the tree would represent the most recent common ancestor of all leaves.

The most common method for rooting phylogenetic trees is by using an outgroup [1], that is, a taxon or set of taxa which is a relative of the group of taxa under study (ingroup) which is related to the ingroup but is lacks a sufficient number of characteristics common amongst the ingroup. The outgroup should be similar enough to allow inference, yet different enough to be distinguished from the ingroup. The ingroup members should be more closely related to one another than to the outgroup. Points where an outgroup is connected to the rest of the tree is taken as the root. Analogues to phylogenetic outgroups are sometimes found in traditions which incorporate texts or parts of texts from other traditions or early translations. Unfortunately, stemmatological outgroups are rare and thus philologists must determine the root via other methods. Reticulation is when two taxa merge and create a new one and is largely analogous to hybridisation, where two species interbreed or in horizontal gene transfer, where genes migrate from one species to another (such as from viruses to bacteria). Contamination is analogous to reticulation and can only be explained via networks rather than trees and it may even necessitate multiple roots rather than just one such as when multiple variants are published at once or when manuscripts which include the root have been lost.

In phylogenetics trees the taxa are placed as leaves at the end of the tree and are separated by internal nodes which correspond to the common ancestors of the taxa. The number of internal nodes in a phylogenetic tree depends on the number of taxa in the tree. The relationship between the number of taxa n and the number of internal nodes i in a binary phylogenetic tree is $i = n - 1$ since each taxon contributes one leaf node and each internal node has two child nodes. However, the root node does not have an incoming edge, thus it is not counted as an internal node. In Figure 5 the inner nodes are the points separating the taxa. This is also an example of an ultrametric tree where the taxa are all equidistant from the root. In such a tree, the branch lengths can be though of as time since the species all evolve at the same rate. Such an assumption is not compatible with stemmatology since the rate of copying errors made by scribes changes drastically between scribes and the conditions in which they were copying. Moreover, manuscripts usually produce multiple daughter documents throughout their sometimes very long lifespans, thus, time does not necessarily correlate with larger dissimilarities.

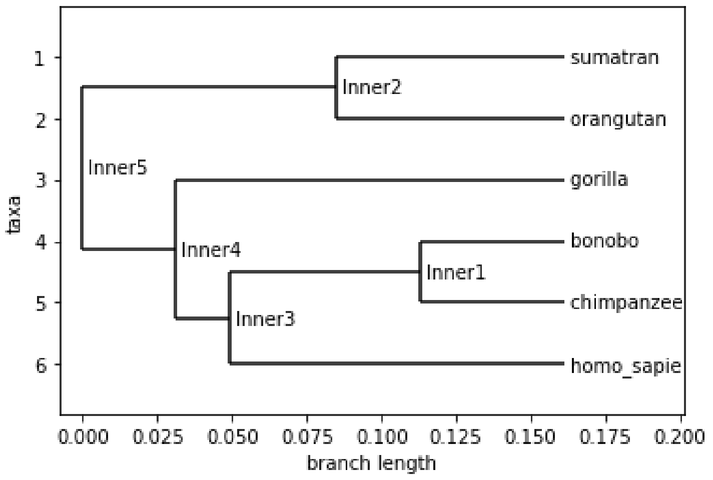


Figure 5: An phylogenetic tree of some primates with internal nodes and ranch lengths labelled.

1.4 Outline

The aim of this thesis is to create a pipeline to reconstruct stemmata using phylogenetic methods and compare them to some standard methods for natural language processing. The latter has been explored fairly recently in [9] and [10] with some positive results. Phylogenetic methods have been used for stemmatics for quite some time [8], however, papers on the subject are very scarce. In this thesis we build a pipeline using ready made tools and packages for text processing, reconstruction of stemmata and accuracy measurements.

Another objective is to test various stemma reconstruction methods for different numbers of manuscripts, text lengths and for missing data such as different numbers of missing manuscripts and missing text.

The report is structured as follows: In chapter 2, we introduce a variety of methods from phylogenetics and natural language processing which can be used to reconstruct a stemma. In chapter 3 we discuss the pipelines used, experimental setups, datasets and data management, accuracy measures as well as implementation details. In chapter 4, we show our results which are then discussed in chapter 5. Finally, in chapter 6 we give a short summary of the work done and make suggestions for future work.

2 Distance Matrics Mehtods for Tree Construction

This section introduces the methods used to reconstruct a stemma's structure. The first half is concerned with calculating the dissimilarity matrix between texts whilst the other is concerned with reconstructing the stemma's structure from the dissimilarity matrix. In this project we consider both natural language processing and phylogenetics methods to calculate the distance matrices.

Distance based methods are two-step processes. In the first step one calculates a symmetric dissimilarity matrix with zeros on the diagonal for the input whilst the second step involves constructing a tree from the distance matrix by clustering the closest data points together. Such matrices do not generally satisfy the triangle inequality and thus are sometimes referred to as dissimilarity matrices rather than distance matrices. Many distance-based tree estimation methods are polynomial time and efficient, and thus are favoured over most other algorithms which tend to be computationally expensive [1]. As an input, phylogenetics distance matrix methods require a sequence alignment and produce as output a pairwise distance matrix. In natural language processing, all that is required is the (processed) text.

Definition 2.1. *The measure d is said to be a distance metric of similarity if and only if:*

- $d(x,y) \geq 0$ *with equality attained* $\iff x = y$
- $d(x,y) = d(x,y)$
- $d(x,y) \leq d(x,z) + d(z,y)$

Distance metrics are the most common methods used for measuring the similarity between words and documents. In natural language processing such techniques are often used for spell-checking, auto-completing sentences or correcting words and sentences. There exist plenty of distance metrics which can be used to compute and measure similarities between texts. In sections 2.1 to 2.3 we introduce methods used to encode text documents. Following this, we describe some methods used in phylogenetics in sections 2.4 to 2.7.

2.1 Word Representation

A primary challenge for any natural language processing problem is representing text in a computer-readable form [11]. The standard way of achieving this is through the creation of a vector space model of the text, where vectors are used to represent text sequences via a variety of methods. This section aims to introduce some of the most notable methods.

Before encoding text, the text is usually first preprocessed. Punctuation, stop-words (words that are common such as "a", "the"etc.) suffixes and prefixes are removed for computational efficiency as well as to remove uninformative words. In the case of stemmatology, the texts are distinguished via very small details, thus removal of suffixes, prefixes and stop-words is not done. Punctuation can sometimes be informative, although it depends heavily on the texts, the era in which they were written and the scribes. Older manuscripts typically have more inconsistent punctuation [1].

2.1.1 Count Vectorization

Arguably the simplest method for representing text as a vector space is by taking each unique word as a separate dimension, and one-hot encode it as a vector, which is usually quite sparse, with length equal to the number of unique words [11]. The bag-of-words model is very commonly used to represent entire texts. It is a count vector derived from the sum of the one-hot encoded word vectors of all words in the document. A drawback is that it completely ignores contextual meaning. For example, the bag-of-words vector for the sentence “Jesse likes cats” is orthogonal to that of the sentence “Jack has an affinity for felines”, even though the sentences have similar meaning.

Given two documents including text ”Jesse likes cats. Jill also likes cats.” and “Jack has an affinity for felines.” the count vectors would be:

	Jesse	likes	cats	Jill	also	Jack	has	an	affinity	for	felines
Document 1	1	2	2	1	1	0	0	0	0	0	0
Document 2	0	0	0	0	0	1	1	1	1	1	1

2.1.2 *n*-grams

n-grams are sequences of n adjacent ordered symbols which may be n adjacent characters, which includes punctuation marks and blanks and also syllables, or sometimes whole words found in a language dataset. These are collected from a text corpus or speech corpus. In natural language processing, *n*-grams allow bag-of-words models to capture information about the word ordering which is completely lost when using bag-of-words models made up of 1-grams.

As an example, the 2-gram or ”bigram” of ”To be or not to be” would be ”To be, be or, or not, not to, to be”. When training on large corpora one can for example calculate the probability of two words appearing next to each other. Usually up to 5-grams are used as the higher the n the more prone to over-fitting the model would be.

2.1.3 Term frequency–inverse document frequency

Term frequency–inverse document frequency (TF-IDF) [12] is an efficient numerical statistic which reflects the importance of a word in a document relative to a collection of documents [11]. It is commonly used in natural language processing and information retrieval.

The term frequency $TF(t, d)$ is the relative frequency of term t in document d:

$$TF(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}, \tag{1}$$

where $f_{t,d}$ is the the number of occurrences of t in document d. As such this is the number of occurrences of a word in a document divided by the total number of words in the document. This naturally leads to common words such as stop words (commonly used but uninformative words such as ”a”, ”the” etc.) to have higher scores since they have higher rates of occurrence. This is counteracted by the inverse document frequency, which decreases with the number of documents the word appears in.

The inverse document frequency measures the informativeness or rarity amongst documents of a term and is defined as:

$$IDF(t, D) = log \frac{N}{|d : \{d \in D \text{ and } t \in d\}|}, \tag{2}$$

where N is the number of documents, D is the set of documents.

Combining these two measures yields the TF-IDF:

$$TF - IDF(t, d, D) = tf(t, d) \times idf(t, D). \tag{3}$$

The weight of a term in a document increases with the number of occurrences in that document and decreases with the number of documents the term occurs in. Thus, the weights tend to give less importance to common terms.

In stemmatology this measure is quite useful since most of the text is identical between scripts except for a few words, thus the mutual text is given less importance than the distinguishing words. However, if the differences between the texts are commonly occurring words this would backfire. Another feature of this measure is that it treats all differently spelled words as separate. It does not treat misspelled words as different from the correctly spelled word and it does not distinguish between synonyms. Another major disadvantage of this method is that it loses the positional information, however, this could be alleviated by taking the TF-IDF of n-grams rather than just the words.

2.2 Word Embeddings

Word embeddings were invented to alleviate the curse of dimensionality problem in one-hot word encodings. Such techniques were designed to transform large one-hot encoded word vectors to a dense continuous vector space capturing contextual relations between words. Thus, embeddings of semantically similar words should be expected to be close in the vector space. Words which frequently appear in the same sentences or contexts are likely to be semantically related. The models are trained on large corpora to determine the statistical properties of the words.

A weakness of this approach is that words which do not occur in the training corpus, often referred to as out-of-vocabulary words, are not explained by the model. If for example one wants to exchange a synonym with one which is not in the training corpus, the model would assign a likelihood of zero to the out-of-vocabulary synonym occurring. Misspellings, which are common in stemmatology, would not usually appear in the training set and thus are taken as unknowns.

2.2.1 Word2Vec

Word2vec [13] consists of two variants: a continuous bag-of-words and a skip-gram model. The continuous bag-of-words model is trained to predict a word given its context as shown in 6. Essentially, the input is the one-hot encoding of words surrounding a target word and the output predicts what the target is. The projection layer is shared between all of the input words, and thus the input word vectors are averaged in the projection layer into the same position. Contrary to the standard bag-of-words model, this model relies on continuous distributed representation of the context, hence the name.

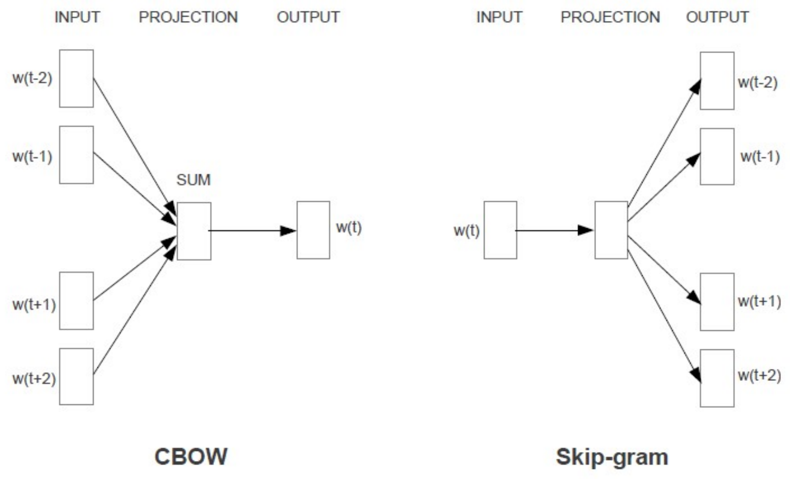


Figure 6: The continuous bag of words and skip-gram models suggested in [13]. Extracted from [13].

In this project we use the second architecture proposed in [13], that is, the continuous skip-gram model which, as shown in Figure 6, is in essence a reversed continuous bag-of-words architecture. Rather than predicting a word given the context, the continuous skip-gram model is trained to predict the words surrounding a particular word. The word vector for each word in the vocabulary is initially random for each word. The algorithm then goes through each locus t using a context window of arbitrary length say c, learning the relations between the word at t and the words between $t-c$ and $t+c$. The larger the window used, the higher the accuracy as and the longer the training time. Finally, once all the context words at locus t are run through, the next step is to find the maximum likelihood of the neighbouring words (from $t-c$ to $t+c$) given the word at the locus t. This likelihood can be represented using the formula:

$$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-c \le j \le c \\ j \neq 0}} P\left(w_{t+j} \mid w_t; \theta\right). \tag{4}$$

To turn this into a minimisation problem, the negative log likelihood is taken. Given a sequence of T words, each word vector is trained to maximise the log probability of neighbouring words in a corpus, i.e., given a sequence of words $w_1, \ldots, w_T$:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-c \le j \le c \\ j \neq 0}} \log P\left(w_{t+j} \mid w_t; \theta\right), \tag{5}$$

where $P\left(w_j \mid w_t\right)$ is the softmax function of the word vector representations of w_j and w_t. The conventional softmax function takes a k-dimensional vector as input and outputs a probability distribution of k possible outcomes. Effectively, it is a generalisation of the logistic function to multiple dimensions.

In the skip-gram model, each word is assigned d-dimensional vector representations to calculate the conditional probabilities. For any word with index i in the vocabulary, the vector when used as a centre word is denoted as $\mathbf{v}_i \in \mathbb{R}^d$ and $\mathbf{u}_i \in \mathbb{R}^d$ is the vector when it is used as a context word. The conditional probability of generating a word w_o given the center word w_c, the basic skip-gram model uses a softmax operation on vector dot products:

$$P\left(w_o \mid w_c\right)=\frac{\exp\left(\mathbf{u}_o^{\top} \mathbf{v}_c\right)}{\sum_{i \in \mathcal{V}} \exp\left(\mathbf{u}_i^{\top} \mathbf{v}_c\right)}. \tag{6}$$

In the numerator is the dot product of the centre word and the context word vectors, thus giving their similarity. The denominator is a normalising function which ensures that the sum of all outputs adds up to 1, thus yielding a probability distribution. It is essentially a generalisation of the logistic function to multiple dimensions and is often used as a so called activation function in artificial neural networks.

During training, the θ parameter vector is optimised. This consists of the stacked word vector representations of all target words and context words.

Taking the derivative of $J(\theta)$ with respect to the centre vector word representation $\boldsymbol{v}_c$ yields:

$$\frac{\partial J(\theta)}{\partial \mathbf{v}_c}=\frac{\partial}{\partial \mathbf{v}_c}\left(\log\left(\exp\left(\mathbf{u}_o^T \mathbf{v}_c\right)\right)\right)-\frac{\partial}{\partial \mathbf{v}_c}\left(\log \sum_{w \in Vocab} \exp\left(\mathbf{u}_w^T \mathbf{v}_c\right)\right). \tag{7}$$

The first term can easily be calculated as $\mathbf{u}_0$ i.e. the context word representation. For the second term take $Z=\sum_{w \in Vocab} \exp\left(\mathbf{u}_w^T \mathbf{v}_c\right)$. Taking the derivative of $Log(Z)$:

$$\frac{1}{\sum_{w \in Vocab} \exp\left(\mathbf{u}_w^T \mathbf{v}_c\right)} \sum_{Z \in Vocab} \frac{\partial}{\partial \mathbf{v}_c} \exp\left(\mathbf{u}_Z^T \mathbf{v}_c\right)=\sum_{Z \in Vocab} \frac{\exp\left(\mathbf{u}_Z^T \mathbf{v}_c\right)}{\sum_{w \in Vocab} \exp\left(\mathbf{u}_w^T \mathbf{v}_c\right)} \mathbf{u}_Z \tag{8}$$

$$=\sum_{Z \in Vocab} P(Z \mid c) \mathbf{v}_Z. \tag{9}$$

Thus, we end up with:

$$\frac{\partial J(\theta)}{\partial \mathbf{v}_c}=-\mathbf{u}_o+\sum_{Z \in Vocab} P(Z \mid c) \mathbf{u}_Z. \tag{10}$$

In the above the first term is the 'current' word vector representation of the context word whereas the second term is the expected context word vector representation should be according to the model. In essence, the difference between the actual and expected representations is used to determine the direction towards which the vector should move by updating the weight vector $\boldsymbol{v}_c$ with the aim of maximising the likelihood.

In similar fashion, one can take the derivative of the cost function with respect to the context vectors. In this case there are two cases: when the context word is the same as the centre word:

$$\frac{\partial J(\theta)}{\partial \mathbf{u}_w} = -\mathbf{v}_c + \sum_{Z \in Vocab} P(Z \mid c)\mathbf{v}_c, \tag{11}$$

and when it is not:

$$\frac{\partial J(\theta)}{\partial \mathbf{u}_w} = \sum_{Z \in Vocab} P(Z \mid c)\mathbf{v}_c. \tag{12}$$

Calculating the denominator consists of taking the exponential of the dot product of all words in the vocabulary and thus, this is very computationally intensive for large corpora. Instead, [13] suggests using hierarchical softmax functions which, instead of calculating the full softmax for all words, breaks down the computation into a series of binary decisions using a binary tree structure, significantly reducing the computational complexity from $O(V)$ to $O(log_2 V)$. This function represents the output layer by a binary tree with the V words as its leaves. For every node it explicitly represents the relative probabilities of its descendent nodes, essentially using a random walk assigning probabilities to the words. There is a path from the root of the tree to each word w in the vocabulary. Take $n(w, j)$ to be the j-th node on the path from the root to w, and let $L(w)$ be the path length such that $n(w, 1)-$ root and $n(w, L(w)) = w$. Moreover, for any inner node n, take an arbitrary fixed child node $\mathrm{ch}(n)$ and let $[\![x]\!] = 1$ if x is true and -1 otherwise. Then the hierarchical softmax approximates the conditional probability $P\left(w_o \mid w_c\right)$ as follows:

$$P\left(w_o \mid w_c\right) = \prod_{j=1}^{L(w_o)-1} \sigma\left([\![n\left(w_o, j+1\right) = \mathrm{ch}\left(n\left(w_o, j\right)\right)]\!] \mathbf{u}_{n(w_o,j)}^{\top} \mathbf{v}_c\right), \tag{13}$$

where function σ is the sigmoid function $\sigma(x) = \frac{1}{1+\exp(-x)}$.

The tree structure used by the hierarchical softmax significantly effects performance. In [13] a binary Huffman tree is used to train the google news 300 dataset, since it assigns short codes to the frequent words speeding up training. The theory behind Huffman trees is beyond the scope of this project.

A major disadvantage of word2vec is that to generate a word embedding for a particular word, the model must be trained on a dataset which contains that word. Thus, spelling mistakes, rare words and compound words which never occur in the vocabulary would not have a word vector. Thus FastText [14] was developed to alleviate this issue.

2.2.2 FastText

FastText is a newer extension of Word2vec designed to alleviate some of the limitations deriving from ignoring word morphology, thus allowing the use of out-of-vocabulary words in the test data [14].

In this case, the softmax function discussed in the previous section is not applicable since if it is given word w_t only one context word w_c can be predicted. Instead, the problem of predicting context words is framed as a set of independent binary classification tasks where the goal is to independently predict the absence or presence of context words. For w_t take all context words as positive examples and sample negatives at random from the dictionary. Given w_c, using the binary logistic loss, the negative log-likelihood can be computed as:

$$\log\left(1+e^{-s(w_t,w_c)}\right)+\sum_{n\in\mathcal{N}_{t,c}}\log\left(1+e^{s(w_t,n)}\right), \tag{14}$$

where $s(w_t,w_c)=\boldsymbol{u}_{w_t}^{\top}\boldsymbol{v}_{w_c}$ is the dot product of the word vector representations of w_t and w_c whereas $\mathcal{N}_{t,c}$ is a set of negative examples sampled from the vocabulary. The logistic loss function $\ell : x \mapsto \log\left(1+e^{-x}\right)$, allows the objective to be expressed:

$$\sum_{t=1}^{T}\left[\sum_{c\in\mathcal{C}_t}\ell\left(s\left(w_t,w_c\right)\right)+\sum_{n\in\mathcal{N}_{t,c}}\ell\left(-s\left(w_t,n\right)\right)\right]. \tag{15}$$

Through the use of unique vector representations for every word, the skip-gram model ignores the spelling of words. In [14], a different scoring function s is proposed in order to take into account this information.

The embeddings are trained on the words' constituent character n-grams, including word boundaries denoted as "<" for the word's start and ">" for its end, together with the whole word itself to learn a representation for each word. The word vector is then computed from the sum of the embeddings of all n-grams. As an example, the set of n-grams representing the word 'self' with $n=3$ is $\{<\mathrm{se},\mathrm{sel},\mathrm{elf},\mathrm{lf}>,<\mathrm{self}>\}$. The word boundaries ensure that the word 'elf' with corresponding sequence $<\mathrm{elf}>$ does not get confused with the 'elf' in 'self'. This approach allows the model to generate accurate embeddings for rare, misspellings, compound words, or words which occur in different forms or tenses. Moreover, it does not need to learn separate vectors for different morphological forms of the same word. If a word is particularly rare in a corpus, but still shares character n-grams with more common words it can be assigned a robust embedding.

Given a size G dictionary of n grams and a word w, denote the set of n-grams appearing in w as $\mathcal{G}_w\subset\{1,\ldots,G\}$ and to each n-gram g associate a word vector $\mathbf{z}_g$. Each word vector is calculated as the sum of its character n-gram vector representations, leading to the score function:

$$s(w,c)=\sum_{g\in\mathcal{G}_w}\mathbf{z}_g^{\top}\mathbf{v}_c, \tag{16}$$

which shares the representations across words and is thus able to learn word embeddings for rare words, misspellings as well as compound words.

2.2.3 GloVe

Context-based methods such as Word2vec only learn relationships from local context and capture word location and ordering. Statistical methods leverage the information contained in the corpus since they are derived from the entire corpus. GloVe [15] is a competitive recent word embedding model which uses the statistics of the word co-occurrence matrix rather than word frequencies on the entire corpus and attempts to keep some of the advantages of both statistical and context based methods, although it is at its core a statistical model. Since it uses co-occurrences rather than frequencies, some information is kept about the context of words.

Take X_{ij} to be the frequency of a word i in the windows of a word j. P_{ij} and X_i are defined as:

$$P_{ij} = P(j \mid i) = \frac{X_{ij}}{X_i} = \frac{X_{ij}}{\sum_k X_{ik}}. \tag{17}$$

The scores depend on two target words and one context word leading to to the most general form of the model:

$$F\left(w_i, w_j, \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}, \tag{18}$$

with word vectors $w \in \mathbb{R}^d$ and context vector $\tilde{w} \in \mathbb{R}^d$. This ratio is better at distinguishing relevant from irrelevant words and distinguishing relevant words compared to the raw probabilities. In the above equation, the right hand side is derived from some large corpus. The definition of F is based on some constraints. F must encode the information in the right hand side in the vector space. Since vector spaces are linear, F can be constricted to use only the difference between two target words:

$$F\left(w_i - w_j, \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}. \tag{19}$$

To prevent F from mixing vector dimensions non-linearly, the arguments' dot product is taken:

$$F\left(\left(w_i - w_j\right)^T \cdot \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}. \tag{20}$$

Now, for the word-word co-occurrence matrices, the distinction between a word and a context word is arbitrary meaning their roles are interchangeable. Thus, it must also be ensured that this is invariant to swapping $w \leftrightarrow \tilde{w}$ and that $X \leftrightarrow X^T$, since the difference between a context and a regular word should be exchangeable. Hence, F is taken to be a homomorphism for the groups $(\mathbb{R}, +)$ and $(\mathbb{R}_{>0}, \times)$, meaning that any function $F(A+B)$ such that $A, B \in \mathbb{R}$ can be recast as $F(A) * F(B)$ whilst keeping the same output. Moreover, this then means that $F(A-B)$ can be expressed as $F(A)/F(B)$, allowing equation (20) to be rewritten as:

$$F\left(\left(w_i - w_j\right)^T \cdot \hat{w}_k\right) = \frac{F\left(w_i^T \cdot \tilde{w}_k\right)}{F\left(w_j^T \cdot \tilde{w}_k\right)}. \tag{21}$$

From equation (20) we can see that this can be solved with:

$$F\left(w_i^T \tilde{w}_k\right) = P_{ik} = \frac{X_{ik}}{X_i}. \tag{22}$$

It is obvious that equation (21) can be solved by $F = exp$ or:

$$w_i^T \cdot \tilde{w}_k = \log\left(P_{ik}\right) = \log\left(X_{ik}\right) - \log\left(X_i\right), \tag{23}$$

where the $\log(X_i)$ avoids invariance to swapping by enforcing asymmetry. Since this is independent of the context term k, it can be exchanged with bias b_i. To retain symmetry, a similar bias $\tilde{b}_k$ is added:

$$w_i^T \cdot \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik}) . \tag{24}$$

Unfortunately, this is an ill-defined function since the logarithm diverges for zero argument. To resolve this, it is possible to add an additive shift to the logarithm:

$$\log(X_{ik}) \to \log(1 + X_{ik}) , \tag{25}$$

to maintain X's sparsity whilst at the same time avoiding the divergences. A major disadvantage to this is that all co-occurrences are weighed equally, including ones which never or rarely occur. Now, rare co-occurrences introduce noise and are less informative. Even word pairs which appear only once in the entire corpus are weighed equally to pairs which occur multiple times. Thus, in [15], equation (24) can be recast into a least squares problem, and a weighting factor is added to the cost function. Given a vocabulary of size V:

$$J = \sum_{i,j=1}^{V} f(X_{ij}) \left(w_i^T \cdot \tilde{w}_j + b_i + \tilde{b}_j - \log(X_{ij}) \right)^2 . \tag{26}$$

The weighting function $f(X_{ij})$ is taken such that it has the following properties:

- $f(0) = 0$. If f is viewed as a continuous function, it should vanish as $x \to 0$ fast enough that the $\lim_{x\to 0} f(x) \log_2 x$ is finite.
- $f(x)$ should be non-decreasing so that rare co-occurrences are given too much weight.
- $f(x)$ should be relatively small for large values of x, such that frequent co-occurrences are not given too much weight.

There exist many such functions which satisfy these properties, but the paper by [15] recommends:

$$f(x) = \begin{cases} \left(\frac{x}{x_{\max}}\right)^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{else} \end{cases} , \tag{27}$$

where $x_{\max}$ is a predetermined cutoff point constant and α is a parameter.

Like Word2Vec, Glove is not capable of predicting out-of-vocabulary word vectors since it does not take sub-word information into account.

2.3 NLP Document dissimilarity Calculation

In the previous section, we introduced some methods in which texts can be encoded into vectors. In this section we introduce some common methods to derive the dissimilarity between the document vectors from which dissimilarity matrices can be computed. It is from the dissimilarity matrix that the texts can be later clustered into a tree-like structure.

2.3.1 Cosine Distance

The cosine similarity is a similarity measure between two non-zero vectors and is defined as the cosine of the angle between the vectors:

$$(cos)(\theta) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{|\boldsymbol{x}||\boldsymbol{y}|} \tag{28}$$

Cosine similarity is a popular metric used to measure the text-similarity between vector representations of two documents irrespective of their size in text analysis. Words are represented in a vector and the text documents are represented as an n-dimensional vector space. Cosine similarity depends only on the angle between the vectors rather than on the vectors' magnitudes.

A major advantage of this measure is its simplicity, in particular for sparse vectors since only the non-zero coordinates are considered. For TF-IDF in section 2.1.3, the document is represented as a vector where the elements correspond to the TF-IDF statistic for each word in the corpus, thus the dissimilarity between the documents can be calculated as the angle between the document vectors. For word embeddings, one approach would be to take the document vector to be the sum of the word vectors of all words in the document, and then take the cosine distance of this. This was attempted for this task in [10]. In this project we use the more modern word mover's distance.

2.3.2 Word Mover's Distance

The word mover's distance [16] computes document dissimilarity by using the optimal alignment between the sets of word embeddings or the minimal cumulative distance that the embedded words of one document need to "move" to reach the embedded words of the other. Word mover's distance is inspired by the Earth Mover's Distance which is a very well studied transport problem. The main advantage of this metric is that it captures the distance between individual words. Take the example in Figure 7. The two sentences have no common words yet contain nearly identical information which cannot be explained using word frequencies or the bag of words model.

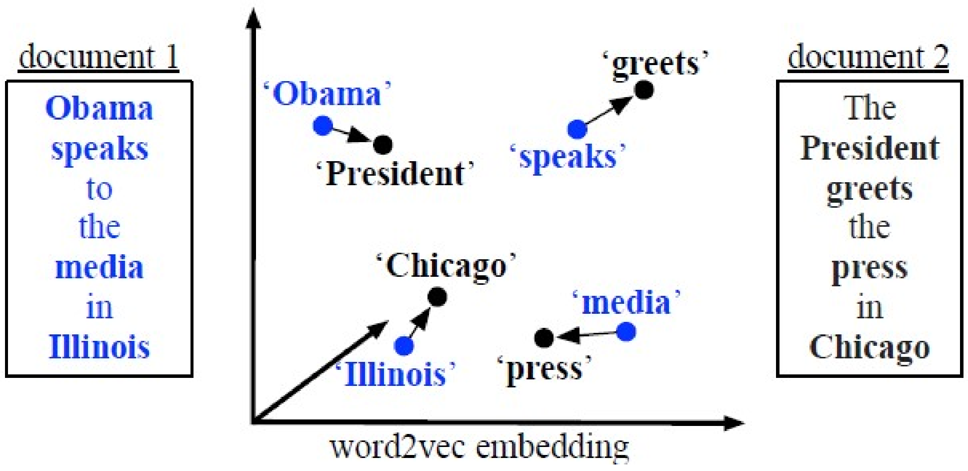


Figure 7: An illustration of the word mover's distance where all non-stop words (bold) of both documents are embedded into a word2vec space.

The word mover's distance is widely used due to its interpretability, lack of hyperparameters and high accuracy. Having said that, the best average time complexity scales to $O\left(n^3 \log n\right)$, where n is the number of unique words in the documents, making it impractical for long scripts datasets. Another advantage it has is the fact that it naturally incorporates the knowledge encoded in a word2vec space. As discussed in section 2.2 each word vector is trained to maximise the log probability of neighbouring words in a corpus, i.e., given a sequence of words $w_1, \ldots, w_T$:

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{j \in nb(t)} \log p\left(w_j \mid w_t\right), \tag{29}$$

where $nb(t)$ is the set of neighboring words of word w_t and $p\left(w_j \mid w_t\right)$ is the hierarchical softmax of the word vectors $\boldsymbol{v}_{w_j}$ and $\boldsymbol{v}_{w_t}$.

Given a word embedding matrix of embeddings $\boldsymbol{X} \in \mathbb{R}^{d \times n}$ for n words, the i^{th} column, $\boldsymbol{x}_i \in \mathcal{R}^d$ represents the embedding of the i^{th} word in d-dimensional space. Assuming the text documents are represented as normalised bag-of-words vectors $\boldsymbol{d} \in \mathbb{R}^n$ (which are usually very sparse since most words do not appear in any given document.) thus, given that a word i appears f_i times in a document the word's weight (which signifies its importance) is taken as:

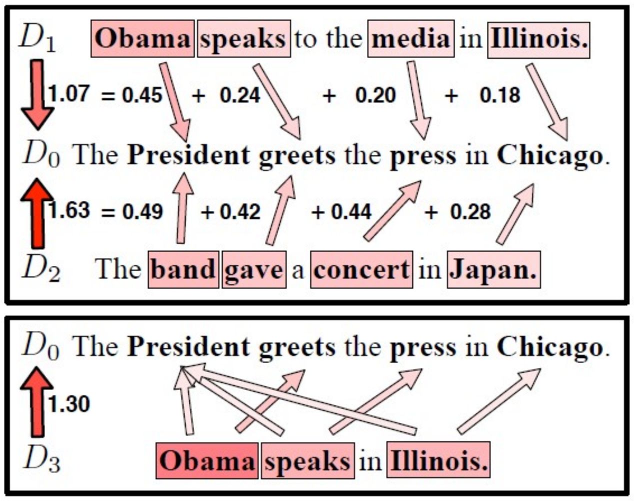
$$w_i = \frac{f_i}{\sum_{j=1}^{n} f_j}, \tag{30}$$

where the f_j's are the frequencies of all words and n is the number of unique words in the document. To incorporate semantic similarity between pairs of words into the document distances the Euclidean distance over the word embedding space is used. The dissimilarity between word w_i and word w_j can be computed as

$$c\left(w_i, w_j\right) = \left\|x_i - x_j\right\|^2, \tag{31}$$

where x_i and x_j respectively correspond to the word embeddings of w_i and w_j. Let $\boldsymbol{d}$ and $\boldsymbol{d'}$ be normalised bag-of-words representations of documents d and d' and $T \in \mathbb{R}^{n \times n}$ be a flow matrix, where $T_{ij} \geq 0$ denotes how much the word w_i in d has to "move" to reach the word w_j in d' as shown in Figure 8. To transform $\boldsymbol{D}$ to $\boldsymbol{D'}$ the complete flow $\sum_j \boldsymbol{T}_{ij}$ from the word w_i should equal d_i and similarly the incoming flow $\sum_i \boldsymbol{T}_{ij}$ to the word w_j should equal d'_j. The word mover's distance is then defined as the minimal weighted cumulative cost needed to move all words from d to d':

$$\min_{\boldsymbol{T} \geq 0} \sum_{i,j=1}^{n} \boldsymbol{T}_{ij} c\left(w_i, w_j\right) \tag{32}$$



D_1 Obama speaks to the media in Illinois.
1.07 = 0.45 + 0.24 + 0.20 + 0.18
D_0 The President greets the press in Chicago.
1.63 = 0.49 + 0.42 + 0.44 + 0.28
D_2 The band gave a concert in Japan.
D_0 The President greets the press in Chicago.
1.30
D_3 Obama speaks in Illinois.

Figure 8: The components of the word mover's distance between a query D_0 and two sentences D_1 and D_2 (with equal bag-of-words distance). The arrows represent flow between two words and are labeled with their distance contribution. (Bottom:) The flow between two sentences D_3 and D_0 with different numbers of words. This mismatch causes the measure to move words to multiple similar words. Extracted from [16].

Given these constraints, the solution of the transformation problem is calculated through:

$$\min_{\boldsymbol{T} \geq 0} \sum_{i,j=1}^{n} \boldsymbol{T}_{ij} c(w_i, w_j) \tag{33}$$

$$\text{subject to: } \sum_{j=1}^{n} \boldsymbol{T}_{ij} = d_i \quad \forall i \in \{1, \ldots, n\}, \tag{34}$$

$$\sum_{i=1}^{n} \boldsymbol{T}_{ij} = d'_j \quad \forall j \in \{1, \ldots, n\}. \tag{35}$$

The best case computational complexity for this algorithm is $O(Vocab^3 \log(Vocab))$ [16].

2.3.3 Relaxed Word Mover's Distance

For corpora with many unique words or large corpora calculating the word mover's distance can become very computationally demanding. In our case, since the documents are very similar having many unique words from different documents is usually not a problem since most words appear in all documents, however, one may have to deal with long texts or many witnesses. [16] also shows how the word mover's distance can be sped up by relaxing the distance optimisation problem by removing one of the two constraints leading to the optimisation problem:

$$\min_{\mathbf{T} \geq 0} \sum_{i,j=1}^{n} \mathbf{T}_{ij} c(i,j) \tag{36}$$

$$\text{subject to: } \sum_{j=1}^{n} \mathbf{T}_{ij} = d_i \quad \forall i \in \{1, \ldots, n\}. \tag{37}$$

This results in a lower bound to the word mover's distance. The optimal solution is for each word in d to move all its probability mass to the most similar word in d$'$. The optimal T^* matrix is defined as:

$$\mathbf{T}_{ij}^{*} = \begin{cases} d_i \text{ if } j = \operatorname{argmin}_j c(i,j) \\ 0 \text{ otherwise.} \end{cases} \tag{38}$$

If $\mathbf{T}$ is a feasible matrix for the relaxed problem, the contribution to the objective value for any word i, with closest word $j^* = \operatorname{argmin}_j c(i,j)$, cannot be smaller:

$$\sum_j \mathbf{T}_{ij} c(i,j) \geq \sum_j \mathbf{T}_{ij} c\left(i, j^*\right) = c\left(i, j^*\right) \sum_j \mathbf{T}_{ij} \tag{39}$$

$$= c\left(i, j^*\right) d_i = \sum_j \mathbf{T}_{ij}^{*} c(i,j). \tag{40}$$

Hence, T^* results in a minimum objective value. To caclulate this solution one needs only to identify $j^* = \operatorname{argmin}_i c(i,j)$ via a nearest neighbor search in the Euclidean word2vec space. For every word vector x_i in document D, one needs to find the closest word vector x_j in document D'. Removing the first constraint reverses nearest neighbor search. The lower bounds depend on the pairwise distances between the word vector representations. Such calculations from removing either constraint can be combined with the other to obtain both bounds. Let the two relaxed solutions be $\ell_1\left(\mathbf{d}, \mathbf{d}'\right)$ and $\ell_2\left(\mathrm{~d}, \mathbf{d}'\right)$. Taking the maximum of the two, $\ell_r\left(\mathbf{d}, \mathbf{d}'\right) = \max\left(\ell_1\left(\mathbf{d}, \mathbf{d}'\right), \ell_2\left(\mathbf{d}, \mathbf{d}'\right)\right)$ results in an even tighter bound which is referred to as the "relaxed" word mover's distance. In this case the complexity is $O(Vocab^2)$

Whilst the relaxed word mover's distance is faster, it is less accurate than the word mover's distance since it only considers the "best-case" scenario for moving words without accounting for the balanced distribution of words across documents.

2.4 Sequence Similarity Metrics

In this subsection we introduce spelling based measures. These are used during the alignment process and the derivation of the distance matrices in the phylogenetics methods through the PAUP* package (see section 3.1.2).

2.4.1 Hamming Distance

The Hamming distance [7] between two strings or vectors of equal length is the number of positions at which the corresponding symbols differ i.e. the minimum number of substitutions required to change one string into the other.

$$HD(x, y) = \sum_i \nVdash(x_i \neq y_i) \tag{41}$$

In texts, the strings are usually not of the same length, as in multiple sequence alignment, and thus this metric cannot be used directly. The Hamming distance can be used if the words in a collection of scripts are aligned and turned into individual characters to produce a multiple alignment sequence as in bioinformatics, however, one loses much information in this manner. This method is used in PAUP* (section 3.1.2) to calculate the number of mismatches there are in the multiple aligmnent sequences.

To calculate the edit distance between words of differing lengths, this concept was then extended into the Levenshtein distance.

2.4.2 (Demerau-)Levenshtein Distance

The Levenshtein distance is a generalisation of the Hamming Distance used for measuring the difference between two sequences of arbitrary lengths. It is the minimum number of insertions, deletions or substitutions required to change one word into the other. This method is used in in the CollateX package (see section 3.1.1) to align the words according to similarities in spelling.

This can be further extended to the Demerau-Levenshtein distance by allowing for transpositions for adjacent characters. The Demerau-Levenshtein distance [17] between two strings x and y and corresponding character positions i and j is defined as:

$$LD_{x,y}(i, j) = \begin{cases} 0 & \text{if } i = j = 0, \\ LD_{x,y}(i-1, j) + 1 & \text{if } i > 0, \\ LD_{x,y}(i, j-1) - 1 & \text{if } j > 0, \\ LD_{x,y}(i-1, j-1) + \nVdash(x_i \neq y_j) & \text{if } i, j > 0, \\ LD_{x,y}(i-2, j-2) + \nVdash(x_i \neq y_j) & \text{if } i, j > 0 \text{ and } x_i = y_{j-1} \text{ and } x_{i-1} = y_j, \end{cases}$$

The first case is the trivial case of empty strings, the second case corresponds to a deletion from x to y, the third case to an insertion from x to y, the fourth case corresponds to a match or mismatch, depending on whether the respective characters are the same whereas the final fifth case (which distinguishes the Demerau-Levenshtein from the regular Levenshtein distance) corresponds to a transposition between two successive characters. A typical implementation via a recursive algorithms is shown in Algorithm 1. This algorithm is best used to compare individual words rather than documents as it is very computationally intensive [17].

Algorithm 1 Demerau-Levenshtein Distance Algorithm

```
procedure DemerauLevenshteinDistance($s, t$)        ▷ $s$ and $t$ are the input strings
    $m \leftarrow$ length of string $s$
    $n \leftarrow$ length of string $t$
    Initialize a 2D array $D$ with dimensions $(m+1) \times (n+1)$
    for $i$ from 0 to $m$ do
        $D[i][0] \leftarrow i$
    end for
    for $j$ from 0 to $n$ do
        $D[0][j] \leftarrow j$
    end for
    for $i$ from 1 to $m$ do
        for $j$ from 1 to $n$ do
            if $s[i] = t[j]$ then
                $cost \leftarrow 0$
            else
                $cost \leftarrow 1$
            end if
            $D[i][j] \leftarrow \min(D[i-1][j]+1, D[i][j-1]+1, D[i-1][j-1]+cost)$
            if $i > 1$ and $j > 1$ and $s[i] = t[j-1]$ and $s[i-1] = t[j]$ then
                $D[i][j] \leftarrow \min(D[i][j], D[i-2][j-2]+cost)$        ▷ Transposition
            end if
        end for
    end for
    return $D[m][n]$        ▷ Return the Demerau-Levenshtein distance
end procedure
```

2.5 Clustering Algorithms

Once the distances between the different texts are calculated, the distance matrix can then be used to generate the network or tree structure. In this section we introduce some phylogenetic and general clustering methods which could be used to reconstruct a stemma. We begin by introducing least squares for phylogenetics, followed by two phylogenetics simple clustering algorithms (UPGMA and Neighbour Joining). The main disadvantage of phylogenetic algorithms is that they are usually limited to bifurcating trees. Thus, we also introduce minimum spanning trees.

2.5.1 Least Squares

We begin by introducing the least squares method for phylogenetics following [7]. This method is used to calculate the branch lengths in the PAUP* software for the aforementioned clustering algorithms given the multiple alignment sequences. Given observed distances D_{ij} and expected distances d_{ij}, the least squares measure is defined as:

$$Q = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \left(D_{ij} - d_{ij}\right)^2, \tag{42}$$

where the w_{ij} are weights depending on the particular variant of least squares with the most common three being $w_{ij} = 1$, $w_{ij} = \frac{1}{D_{ij}^2}$, $w_{ij} = \frac{1}{D_{ij}}$ [7].

If we number all branches of the tree and introduce an indicator variable $x_{ij,k}$, which is 1 if branch k lies in the path from species i to species j and else 0. The expected distance between i and j will then be

$$d_{i,j} = \sum_{k} x_{ij,k} \lambda_k. \tag{43}$$

Equation (42) then becomes

$$Q = \sum_{i=1}^{r} \sum_{j:j \neq i} w_{ij} \left(D_{ij} - \sum_{k} x_{ij,k} \lambda_k\right)^2 \tag{44}$$

Taking the derivative of Q with respect to v_k, and equating to zero:

$$\frac{dQ}{d\lambda_k} = -2 \sum_{i=1}^{n} \sum_{j:j \neq i} w_{ij} x_{ij,k} \left(D_{ij} - \sum_{k} x_{ij,k} \lambda_k\right) = 0 \tag{45}$$

Stacking the D_{ij}'s into a $\frac{n(n-1)}{2}$-dimensional vector $\boldsymbol{d}$ and rearranging the coefficients $x_{ij,k}$ into a $\frac{n(n-1)}{2} \times K$ matrix where K is the number of branches in the tree and where each row corresponds to the D_{ij} in that row of $\boldsymbol{d}$ and containing a 1 if branch k occurs on the path between species i and j:

$$\boldsymbol{X}^T \boldsymbol{d} = \left(\boldsymbol{X}^T \boldsymbol{X}\right) \boldsymbol{\lambda} \tag{46}$$

Multiplying on the left by the inverse of $\boldsymbol{X}^T \boldsymbol{X}$, we can solve for the least squares branch lengths:

$$\boldsymbol{\lambda} = \left(\boldsymbol{X}^T \boldsymbol{X}\right)^{-1} \boldsymbol{X}^T \boldsymbol{d} \tag{47}$$

For diagonal weight matrix $\boldsymbol{W}$:

$$\boldsymbol{X}^T \boldsymbol{W} \boldsymbol{d} = \left(\boldsymbol{X}^T \boldsymbol{W} \boldsymbol{X}\right) \boldsymbol{\lambda}, \tag{48}$$

with solution

$$\boldsymbol{\lambda} = \left(\boldsymbol{X}^T \boldsymbol{W} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^T \boldsymbol{W} \boldsymbol{d} \tag{49}$$

2.5.2 Unweighted Pair Group Method with Arithmetic Mean (UPGMA)

UPGMA is an agglomeration algorithm which computes a rooted tree from an input distance matrix [7]. In the first iteration, it finds the pair of taxa which have the smallest distance, clusters them together and the distance from the new cluster to every other taxon is calculated as the average of the distance from each element of the cluster to the other taxa. In each iteration, the pair of clusters with the smallest distance are clustered together. The distance matrix is then updated by removing the rows and columns for the aforementioned clusters and replacing them with a row for the new cluster. This process is repeated until all the taxa are merged into a single cluster which produces a rooted tree. The distance matrix can be updated in a variety of ways such as by simply removing the row and column corresponding to one of the closest two taxa and this has an impact on the output tree. Algorithm 2 shows the implementation of UPGMA.

Algorithm 2 UPGMA Clustering Algorithm

```
1: Initialize each species as a separate group with $n_i = 1$ for all $i$.
2: while more than one group remains do
3:     Find the pair of groups $(i, j)$ with the smallest distance $D_{ij}$.
4:     Create a new group $(i, j)$ with $n_{(ij)} = n_i + n_j$ members.
5:     Connect $i$ and $j$ on the tree to a new node corresponding to the new group $(ij)$.
6:     Set the branch lengths from $i$ to $(ij)$ and from $j$ to $(ij)$ to $D_{ij}/2$.
7:     for each group $k$ not equal to $i$ or $j$ do
8:         Compute the distance between the new group $(ij)$ and group $k$ using:
               $D_{(ij),k} = \left(\frac{n_i}{n_i + n_j}\right) D_{ik} + \left(\frac{n_j}{n_i + n_j}\right) D_{jk}$
9:     end for
10:    Delete the rows and columns of the distance matrix corresponding to groups $i$ and $j$.
11:    Add a new row and column for the new group $(ij)$.
12: end while
```

The UPGMA algorithm produces rooted trees and assumes an ultrametric tree in which the distances from the root to every branch tip are equal, a condition which is almost always violated in stemmata.

2.5.3 Neighbour Joining

Neighbor joining is one of the most widely used distance-based method due to its accuracy in constructing unrooted phylogenetic trees [18]. Neighbor joining is also an iterative agglomerative technique wherein the tree is built from the bottom up. The input is an $n \times n$ dissimilarity matrix d and in the first iteration, the n leaves are all in their own clusters. In subsequent iterations, each cluster is a set of leaves, however, the clusters are disjoint. At the beginning of each iteration the taxa are partitioned into clusters each of which has a rooted tree that is leaf-labeled by the elements in the cluster. During each iteration a pair of clusters is selected to be made siblings resulting in the cluster's respective trees to be merged into a rooted tree by making their roots siblings. Given three sub-trees, the three sub-trees are merged into a tree on all the taxa by adding a new node, and making the roots of the three sub-trees adjacent to it. Once the tree is generated the root is ignored such that neighbor joining yields an unrooted tree. The advantage of Neighbor joining over UPGMA is that it chooses which pair of clusters to make siblings (and how to update the distance matrix) using a more sophisticated strategy:

First it computes the $n \times n$ Q matrix:

$$Q_{ij} = (n-2)d_{ij} - \sum_{k=1}^{n}(d_{ik} + d_{jk}). \tag{50}$$

Whilst $n > 2$ it then finds the pair i, j minimising Q_{ij}, say l, m. The rooted trees associated with taxa l and m are made siblings. The distance matrix is updated by deleting the rows and columns for l and m, and including a new row and column for the formed tree's root u and set $d_{u,k} = \frac{d_{lk}+d_{mk}-dlb}{2} \forall k \neq u$. Subtract n by 1.

When there are only two elements left in the distance matrix i.e. $n = 2$; the star tree with a single internal node v where the roots of the three rooted trees are all adjacent to v is outputted.

Neighbour Joining is very efficient compared to other methods which makes it useful when dealing with large data sets as well as for bootstrapping. Unlike UPGMA it is statistically consistent under many models of evolution meaning that given enough data, it reconstructs correct phylogenetic trees with high probability [18]. Another advantage neighbor joining has over UPGMA is that it does not assume ultrametricity which also makes it far more compatible in a stemmatological context. Its main downside is that it lacks tree search and optimality criteria meaning there is no guarantee that the recovered tree is optimal. Thus, usually Neighbour Joining is used to produce a tree which is then used as an initial guess in a tree search using some optimality criterion [7]. Below, in Algorithm 3, is an implementation of the neighbour joining algorithm.

Algorithm 3 Neighbour Joining Algorithm

```
1: procedure NeighbourJoining($D$)                    ▷ $D$ is the distance matrix
2:     $n \leftarrow$ number of taxa in $D$
3:     Initialize an empty tree $T$
4:     Initialize a list of taxa $L$ containing all taxa
5:     while $|L| > 2$ do                             ▷ Until only two taxa remain
6:         Compute total branch lengths $L_i$ for each taxon $i$
7:         Compute the $Q$ matrix from distance matrix $D$
8:         Find the minimum element $q_{ij}$ of $Q$
9:         Create a new node $u$ with edge lengths $d_{iu} = \frac{D_{ij}+L_i-L_j}{2}$ and $d_{ju} = D_{ij} - d_{iu}$
10:        Remove taxa $i$ and $j$ from $L$
11:        Attach taxa $i$ and $j$ to node $u$
12:        Update distance matrix $D$ with $u$
13:        Add node $u$ to tree $T$
14:    end while
15:    Attach the two remaining taxa to $T$ with the appropriate edge lengths
16:    return $T$                                     ▷ Return the phylogenetic tree
17: end procedure
```

Such methods can only yield bifurcating trees, however, stemmata are usually multifurcating.

2.5.4 Minimum Spanning Trees

In [9], instead of phylogenetically inspired agglomerative algorithms, minimum spanning trees are constructed using Kruskal's algorithm as shown below in Algorithm 4. A minimum spanning tree is a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices without any cycles and with the minimum possible total edge weight. A feature of this approach is that it does not attempt accounting for missing taxa, resulting in a tree consisting only of the observed variants. This creates a more faithful tree structure in the case of complete datasets with no contamination since it allows for multifurcations, unlike phylogenetics algorithms. To generate minimum spanning trees we use Kruskal's algorithm which has complexity $O(E\log(E))$ where E is the number of edges.

Algorithm 4 Kruskal's Algorithm for Minimum Spanning Tree

```
 1: procedure KRUSKAL($G$)                              ▷ $G$ is the graph
 2:     $T \leftarrow \emptyset$                        ▷ Initialize an empty set for the MST
 3:     Sort all edges of $G$ by weight
 4:     for each vertex $v$ in $G$ do
 5:         MakeSet($v$)
 6:     end for
 7:     for each edge $(u, v)$ in $G$ in sorted order do
 8:         if FIND($u$) $\neq$ FIND($v$) then          ▷ If adding $(u, v)$ doesn't create a cycle
 9:             $T \leftarrow T \cup \{(u, v)\}$        ▷ Add edge $(u, v)$ to MST
10:             UNION($u, v$)                           ▷ Merge the sets containing $u$ and $v$
11:         end if
12:     end for
13:     return $T$                                      ▷ Return the MST
14: end procedure
```

2.6 Optimality Criterion Based Methods

Distance based methods are not considered state of the art in phylogenetics anymore but are instead used to generate an initial guess tree which is then used to infer the correct tree using some optimality criterion. In real cases in phylogenetics, the real tree is unknown and thus there is no way to measure the correctness of the predicted trees. Thus, selection criteria are often used. These selection criteria are also inference methods in of themselves when combined with search algorithms. Starting from a tree constructed using distance matrix methods, the tree's structure is iteratively rearranged and has an optimality criterion computed at each iteration, optimising for the criterion selected. The tree is then taken as the one with the highest/lowest criterion value.

In this project, these methods were only applied to the benchmark datasets and Phareta Fidei since they are too computationally intensive to perform on the large numbers of artificial datasets we generated. Thus, we only give a brief description of the methods.

2.6.1 Parsimony Based Methods

Maximum parsimony is one of the oldest methods to construct trees [7]. The output is a tree T in which the input sequences are placed at the leaves of T and additional sequences are placed at the internal nodes of T such that the total number of changes over the entire tree, is minimised. Essentially it is the 'Hamming Distance Steiner Tree Problem' wherein the input is a set of sequences and the output is a tree connecting these sequences at the leaves with optimised sequences at the internal nodes, minimising the total of the Hamming distances on the tree's edges. The Hamming distance between two sequences of the same length is the number of positions in which they disagree, thus the sum of the Hamming distances (section 2.4.1) on the edges of the tree is equal to the tree length. Determining the optimal tree under this criterion is an NP-hard problem [7], and hence heuristics are used to find suitable solutions which are not necessarily globally optimal.

The aim is to find character state assignments to the internal nodes of the tree so as to minimise the number of edges with different states at the edge's endpoints. This is the "fixed tree parsimony problem" or the "small parsimony problem". Inferring the optimal tree and its internal node labels which yield the optimal parsimony score is called the "large parsimony problem". The tree root does not impact the parsimony score as only this depends only on the total number of changes.

2.6.2 Minimum Evolution

Minimum evolution is a collection of methods [7] whose goal is to find the optimal tree from the set of possible trees. As an input it takes an $n \times n$ dissimilarity matrix D on n leaves, and it assigns edge weights using least squares (section 2.5.1). The total sum of branch lengths of each individual tree topology considered is then computed by adding up the branches lengths in the tree. The tree whose edge weights minimise the total sum of branch lengths is returned. Such methods are distinguished by how they define the edge weights of each tree. For example, given a tree and an input dissimilarity matrix, the optimal weights on the edges could be based on minimising ordinary least squares distances or weighted least squares distances.

2.7 Reliability

Once a tree is generated, it cannot be assumed to be reliable. Typically, scholars use various different methods constructing the tree and place more confidence in subgraphs that are consistently recovered with different methods. A consensus tree is a summary or representation of the phylogenetic relationships derived from multiple individual phylogenetic trees. Phylogeneticists often generate multiple trees to explore the uncertainty and variability in evolutionary relationships among a group of organisms. These individual trees can result from different methods, datasets, or variations in the analysis parameters.

The process of constructing a consensus tree involves combining information from multiple trees to produce a single, summarized representation that reflects the common or well-supported aspects of the phylogenetic relationships.

2.7.1 Bootstrapping

Bootstrapping is a very commonly used resampling method to estimate the statistical reliability of a tree's sub-graphs by taking the original dataset and picking the same number of sites randomly from within the dataset with replacement, some sites are sampled once or more than once whereas others are not sampled at all [7]. This is repeated using the original dataset to generate a large number of different auxiliary datasets. The tree inference method is then applied to each of the auxiliary datasets, as can be seen in Figure 9, and the corresponding output trees are compared to the tree from the original dataset. The percentage of sub-graphs which occur in the auxiliary trees and in the original tree are then calculated and assigned to the nodes which define the sub-graphs in the original tree. A high bootstrap value indicates a high level of confidence that a particular sub-graph is independent of the sites used to calculate the tree. Applying the tree inference method to each auxiliary dataset takes the same amount of time as calculating the original tree, hence there are computational constraints for large datasets or computationally demanding models.

```
                 Observed       Resampled
Homo Sapiens  A-CAATGGAG-A--    AA-AATGCAG-A-C
Pan           A-CAATA-AGCAAA    AAAAATACAGCAAC
Gorilla       ATCAA-A-AGCGG-    AA-AA-ACAGCGGC
```

Figure 9: Bootstrapping: Resampling from one's sample with replacement. The resampled sequence consists of copies of the observed columns with replacement.

2.7.2 Consensus Tree

A consensus tree is a synthetic phylogenetic tree that summarizes the information from a set of different trees [7]. This type of tree is commonly used in evolutionary biology to represent the relationships that are most commonly observed across a collection of trees, often generated from different datasets, bootstrap samples, different methods of tree construction or even trees generated from the same method. The latter case is useful for methods which could potentially yield ties or else trees with very similar selection criterion values such as in maximum parsimony. Consensus trees help in identifying the relationships that are most strongly supported across multiple studies. Moreover, creating consensus trees from bootstrap samples can improve the result [7].

There are two main types of consensus trees: strict and majority-rule. Strict consensus trees include only the groupings which appear in all of the input trees. If there are many conflicting signals in the input trees this can sometimes result in a tree with many polytomies, where multiple branches emerge from a single node. A majority-rule consensus tree includes all clades which appear in more than 50% of the input trees.

2.8 Rooting A Stemma

The root of a tree represents the most recent common ancestor of all taxa in the tree [1]. Thus, it is the oldest part of the tree and defines the the direction of evolution, with the information flowing from the root and towards the leaves.

There are two main methods of rooting a phylogenetic tree [7]: using an outgroup or assuming a molecular clock. In the outgroup method one includes a taxon that is close enough to the set of taxa under investigation (ingroup) to make comparisons however, farther apart from all ingroup taxa are to each other.

An alternative is to use a "molecular clock" where it is assumed that the same amount of changes are observed on all lineages implying that there should be a point in the tree with equal branch lengths from that point to all leaves. The root is the node which makes the amounts of change approximately equal on all lineages. The midpoint method consists of identifying the longest path between two leaves and placing the root at the midpoint of the path.

In stemmatology, the outgroup is rarely available and a molecular clock assumption would almost always be violated since scripts evolve much more dynamically than genetics [1]. Thus, different methods must be used. Typically, a stemmatologist would use their intuition and knowledge to determine which manuscript is older. Older manuscripts would contain older spellings and words. To root a stemma this way, one would require some database containing information about when spellings and words were used. To the author's knowledge, there is no such database in existence, thus other methods must be used.

2.8.1 Minimum Cost Heuristic

In [9], a rooting method which may be applicable to stemmatology is suggested. Given a non-oriented tree, one can assign any of the nodes as the root and calculate the tree cost, that being the sum of the edge weights of all edges on the path from the potential root to all the other nodes in the tree. The root is then taken as the node corresponding to the minimal cost tree. If two or more nodes generate trees of minimal cost, one of them is randomly selected as the root of the reconstructed tree.

The depth-first search algorithm (see Algorithm 5 below), is used to traverse or search trees. Starting at the root node (which is defined using the minimum cost heuristic), the algorithm explores each branch fully all the way down to the leaves and then backtracks.

Algorithm 5 Depth-First Search to Create a Directed Tree

```
1: procedure DFS(node, parent)
2:     for neighbour ∈ neighbours(node) do
3:         if neighbour ≠ parent then
4:             direct_edge(neighbour, node)
5:             DFS(neighbour, node)
6:         end if
7:     end for
8: end procedure
```

Depth first search takes two parameters: node, which is the current node being visited and the parent node from which the current node was accessed. This avoids revisiting the parent node and thus avoids cycles. This is done for all neighours of the current node. The algorithm then checks if the current neighbour is not the parent of the node to avoid going back to the node it came from, which is especially important in undirected graphs where each edge is bidirectional by nature. The edges are then directed away from the parent node at each step and the function is recursively called with the current neighbour as the new node to visit and the current node as its parent allowing it to dive deeper into the graph from the current neighbour. The algorithm then explores its sub-tree until it reaches the leaves, and finally backtrack to explore other branches.

This algorithm is polynomial in time. In a tree with n nodes, the number of pairs of nodes is $\binom{n}{2} = \frac{n(n-1)}{2}$. For each pair of nodes (u, v), there is a unique path in the tree. To compute the sum of edge weights on the path between each pair (u, v) one needs to traverse the path between u and v, summing the weights of the edges on this path, in our case, using depth first search which has a computational complexity of $O(|V| + |E|)$ which is asymptotically

$O(n)$. Now, this has to be done for every node pair and thus this leads to a complexity of $O(n \times \frac{n(n-1)}{2} n) = O(n^3)$. In Figure 10 we plot the time it takes to calculate the root heuristic for randomly generated trees of N nodes. The shape is clearly polynomial and it is feasible to compute for thousands of nodes.

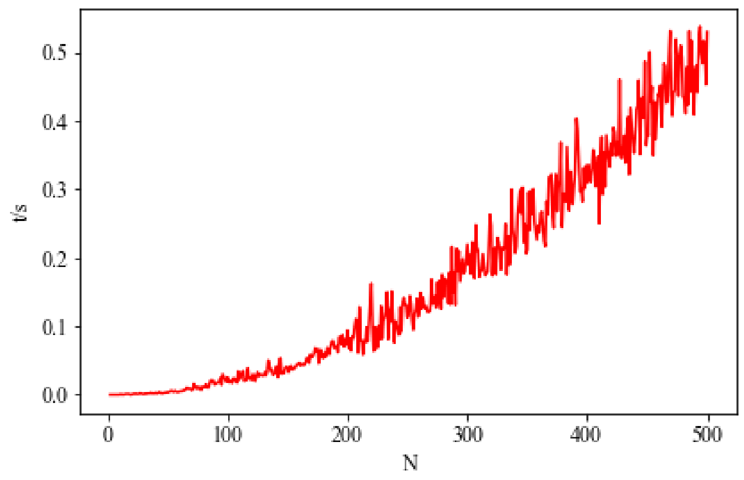


Figure 10: The runtime for calculating the root heuristic vs the number of nodes.

To illustrate how this heuristic is calculated consider the simple example in Figure 11:

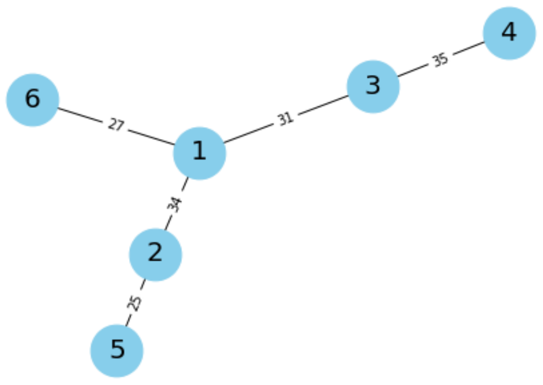


Figure 11: An example of a weighted tree.

In Figure 11, the total cost to traverse all paths starting from node 2 can be calculated as the sum of the cost required to traverse each path from node 2 as:

$$\text{Total Cost}(2) = \Sigma \left\{ \begin{array}{ll} \text{Cost}(2 \to 1) & = 34 \\ \text{Cost}(2 \to 1 \to 3) & = 65 \\ \text{Cost}(2 \to 1 \to 3 \to 4) & = 100 \\ \text{Cost}(2 \to 5) & = 25 \\ \text{Cost}(2 \to 1 \to 6) & = 61 \end{array} \right\} = 285.$$

Repeating, the calculation for all nodes:

$$\begin{aligned}
\text{Total Cost}(1) &= 217, \\
\text{Total Cost}(2) &= 285, \\
\text{Total Cost}(3) &= 279, \\
\text{Total Cost}(4) &= 419, \\
\text{Total Cost}(5) &= 385, \\
\text{Total Cost}(6) &= 325.
\end{aligned}$$

In this example, node 1 happens to yield the smallest value and thus it is taken as the root.

One drawback is that this method assumes that the stemma has only one root which may not necessarily be the case in stemmatology. Another disadvantage of this method is that it assumes that textual evolution leads to trees which tend to balance out over time in terms of the differences between the nodes. If a document's stemma does not branch out, this method becomes inaccurate since a tree with no branchings will have higher costs than one with branchings. To see this, suppose all edges have weight 1 and that each node can have only one child, (a linear tree as in Figure 12). Then the root would be calculated as the node at the middle of the tree. If, however, the tree branches and evens out in the sense that it is more symmetrical with the root close to the centre of the graph, such as when each node has 2 child nodes as in Figure 13, then a node close to the true root would be chosen. In Figure 12, node 3 would have the minimum total cost whilst in Figure 13, the minimum cost node would be the correct node 0.

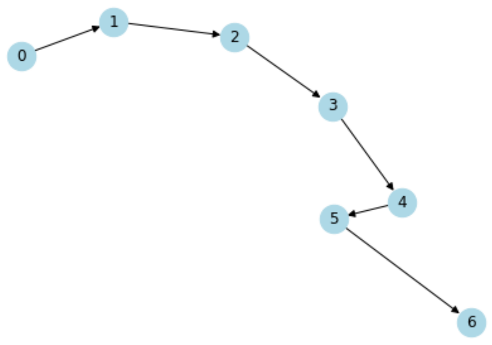


Figure 12: A linear tree where each node can have at most one child nodes.

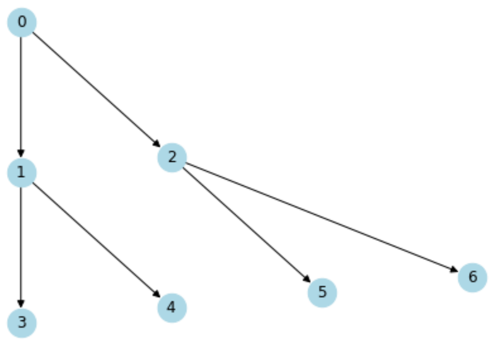


Figure 13: A symmetric binary tree where each node can have at most two child nodes.

To show this, rooted linear (one child per node) and (nearly) symmetric binary trees of with the number of nodes ranging from 0 to 500 were generated. The root was predicted for each using the minimum cost heuristic and the depth (see 3.9.3) was plotted against the number of nodes in the trees. For matching total costs, the first predicted root in a list of predicted roots is selected as the root. As can be seen in Figures 14 and 15, the depth depends on the tree structure. For a linear tree, the root is at the midpoint of the graph.

Since the binary trees were designed to ”balance out”, the predicted root is always at or very close to the actual root.

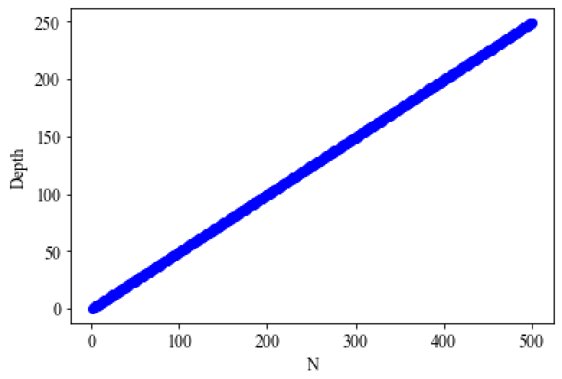


Figure 14: The number of nodes between the correct root and the predicted root for linear trees against the number of nodes.

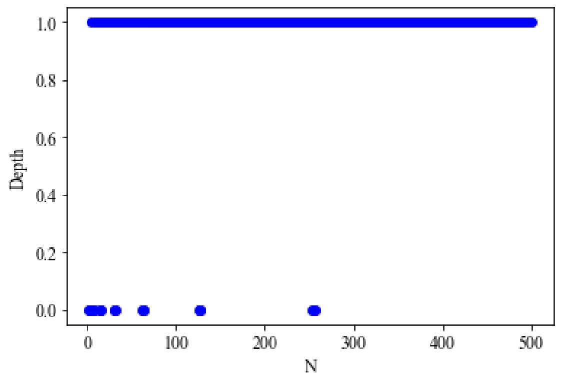


Figure 15: The number of nodes between the correct root and the predicted root for even binary trees against the number of nodes.

3 Methodology and Experiment Design

The main goal of this project was to test various methods to reconstruct the stemma of a given set of manuscripts. Thus, several of the mentioned methods from the previous chapters were benchmarked both on real-world and artificially generated datasets. This chapter serves to detail the experimental setup used for each case, what datasets were used, how the artificial datasets were generated, the models' configuration and the packages used.

3.1 Computational Tools

This project heavily relied on the use of different and often unrelated tools and packages to create pipelines for computing and generating stemmata. The main ones used were:

3.1.1 CollateX

CollateX [19] is a software tool designed for collating, aligning and visualizing multiple versions of a text, especially for tasks related to textual criticism and historical text analysis. It is often used to compare different manuscript versions, editions, or translations of a text and identify variations or differences between them.

The input texts are assumed to have been tokenized. During alignment, CollateX matches tokens across different text versions, aligning identical tokens and inserting placeholders for unmatched ones to ensure that the sequences line up. The alignment might also involve identifying and handling transpositions, where parts of the text have been moved.

3.1.2 PAUP*

PAUP* [20] is one of the most widely used phylogenetic program applied to stemmatology. It offers an easy-to-use graphical interface with many options, algorithms, and parameterisation options. Methods include, parsimony based methods (section 2.6.1), distance matrix methods such as neighbour joining (section 2.5.3), UPGMA (section 2.5.2), least squares (section 2.5.1), minimum evolution (section 2.6.2) as well as bootstrap and consensus trees (section 2.7). One major advantage of PAUP* is that it can readily read and process arbitrary numbers of character states rather than just the "ACTG" used in DNA sequencing. Another advantage of PAUP* is that it is highly optimised to be very computationally efficient and has an easy to use interface with little need for coding skills making it an ideal tool for humanities departments, although it does have a scripting language. This has been used for stemmatology before in [34].

3.1.3 Biopython

To read the PAUP* tree outputs which are in the Newick format, the BioPython package [21] is used to then turn the trees into Networkx objects allowing for the computation of accuracy measures. BioPython is a powerful toolset for biological computation, specialising in phylogenetics. It is an open-source collection of Python libraries and modules that allow researchers, developers, and enthusiasts to handle biological data more efficiently. This package could be used instead of PAUP*, however, we relied on PAUP* since most of the papers on computational stemmatology use PAUP* and PAUP* is also quite efficient [1]. Biopython also cannot read the pseudo-DNA sequences we created, thus methods which require the sequences themselves rather than the distance matrices would not be available.

3.1.4 Networkx

NetworkX is a powerful Python library designed for the creation, manipulation, and study of complex networks of nodes and edges. With its versatile range of tools and functionalities, NetworkX is suitable for analysing both large-scale and small-scale network data across diverse applications. This package was used to generate the artificial stemmatas' structures, process trees and in defining the accuracy measures.

3.1.5 Gensim

Gensim [23] is a robust open-source library designed specifically for unsupervised semantic modeling from plain text, making it an ideal tool for the purposes of this project. It is particularly well-suited for tasks involving large-scale text analysis, such as document similarity and has efficient implementations of word2vec, GloVe, FastText, word mover's distance and much more. Gensim is written in Python and optimized for performance with the use of Cython, making it a highly efficient option for natural language processing tasks that need to handle large volumes of text. This package was mostly used for the calculation of the distance matrices using the methods described in sections 2.1 till 2.3.

3.2 Data Sets

To test the methods artificially generated stemmata were used along with, two stemmatology datasets which were previously used in a computational stemmatology challenge [24] (Parzival and Heinrichi). [24] also included two other traditions, however, the true stemmas are not given and thus these were not used. On top of this, some variants of the Phareta Fidei tradition (provided to us by the philologists mentioned in the Acknowledgements) were studied as a test case for when the true stemma is unknown. In the latter case, heuristics were used to measure the results.

3.2.1 Artificial Datasets

Synthetic datasets consisting of near-duplicate documents were created according to a variety of of parameters. Subsets of the Reuters_50_50 [25] training dataset were used depending on the word-count of the article. Artificial stemmas were generated by taking a document to be the root and generating a set of child documents with randomised edits. Each child document is modified up to an editing limit. The edit operations were selected such that the overall meaning of the child documents does not stray far away from the original. These are:

- Synonym exchange: A given word is replaced by a randomly selected synonym using the WordNet dictionary [26].

- Misspelling and Correction: A word is misspelled or a misspelling is corrected using the Birkbeck [27], Holbrook [28], Aspell [29] and Wikipedia [30] misspelling datasets obtained from [27].
- Insertion and removal of adjectives and adverbs: Adverbs and adjectives are either added or removed before nouns using the NLTK [31] package by checking their part-of-speech tags.
- Paraphrasing. The T5 (text to text transfer transformer), pre-trained on the PAWS (Paraphrase Adversaries from Word Scrambling) dataset from [32] was used to paraphrase a sentence at random.

Whilst these transformations do occur often in stemmatics, some are more impactful than others. Misspellings are often corrected (especially if they are very obvious to the next scribe) and happen more frequently when the scribe is not a native speaker of language in which the manuscript is written. Usually, scribes are quite literate which means they rarely make spelling mistakes and usually only for very complicated words, however, since the misspellings datasets used are extracted from very poor spellers (people with learning disabilities and young children), the kind of spelling mistakes may not be too realistic. Synonym exchange or change in spelling also occur very frequently in stemmatics, especially if the scripts are written centuries apart. Performing synonym exchange using ready-made packages always has the risk of changing the context of the script as synonyms sometimes have different meanings in different contexts. Another common mistake is the doubling of words or of entire sentences. This is usually spotted by the next scribe.

Typically, very little of the text differs from manuscript to manuscript. It is assumed that the errors made during the copying of a manuscript are statistically independent events, although in reality this may not necessarily be the case. Moreover, it is assumed that the errors are rare and therefore far apart enough to be independent in terms of the logical, grammatical, and poetic relationships between the words.

We performed this experiment for synonym, adjective and adverb transformations together, misspellings and paraphrasing separately. For each case we randomly select 100 news articles from the Reuters dataset and generate a predetermined number of child nodes as described above, with each differing by up to a predetermined edit limit. For the first two cases, the edit limit was based on word changes and the number of nodes was increased from 10 to 50 in steps of 10 i.e. for each article we generate 5 different stemmata with the only difference being the number of nodes. For paraphrasing, the edit limit was based on the number of sentences which were paraphrased whilst the edit limit was increased from 10% to 50%.

3.3 Parzival

This dataset is comprised of 21 documents consisting of the first part of Wolfram von Eschenbach's German poem Parzival, translated to English by A.T. Hatto, and copied by hand by scribes. The true stemma has no cases of contamination (i.e. all nodes have at most one parent). Moreover, 5 of the 21 documents are missing.

3.4 Heinrichi

This data set is comprised of 67 variants of an old Finnish text (Piispa Henrikin surmavirsi 'The Death-Psalm of Bishop Henry'). It was artificially constructed by volunteer scribes, who copied a given text by hand, according to an imaginary stemma. To simulate the situation where a portion of the manuscripts are missing 30 of the variants were left out of the dataset and some of the manuscripts also had some significant passages deleted to simulate the situations where manuscripts are partially lost. In our collation this dataset yielded 1090 loci with 147 being constant. Since this is written in Old Finnish, word embedding methods are not applicable as Old Finnish pre-trained vectors seem not to exist. The use of Old Finnish was done to simulate the situation where the scribe is not a native speaker of the language in which the manuscripts were written.

3.5 Phareta Fidei

This dataset is of a real Medieval anti-Semitic tradition written in Latin, with several hundred known variants from different libraries and archives. We were provided with two different datasets, the first containing 10 variants and an expanded version with 28 variants. After collation, the first set of texts had 225 loci about a quarter of which are non-constant. The second had 639 loci with 354 being constant. The texts vary in their spelling, abbreviations, sentence structure as well as word count. It should be noted that only a very small portion of the tradition was provided in both cases, thus results should not be expected to be definitive.

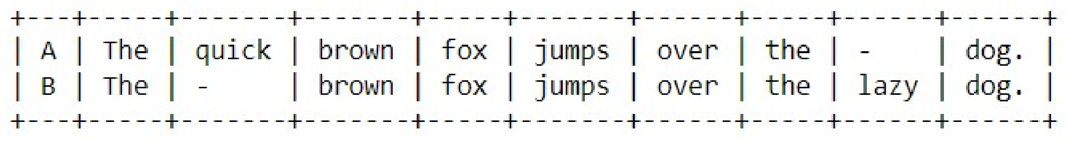
According to the philologist who provided the dataset, Rouen and Roma seemed closely related, the 2 Graz witnesses seemed related to Trier 964, Klosterneuburg and Kremsmunster are related as well and Prague X.B.II seems closely connected to Seitenstetten and Wien 4396. Wien 4396 seemed to be the oldest variant from this group of manuscripts, although this is not definitive.

The orthography and punctuation of the Corpus Christianorum Continuatio Mediaevalis (CCCM) is largely followed meaning that the letter "u" is used for "u" and "v", but the capital "V" for "U" and "V" is used. The letter "i" is used for "i" and "j", both in the case of capital and lowercase letters. E-caudata are represented by an "e", in accordance with normal medieval standards. The punctuation reflects modern usage: no attempt has been made to represent the irregular punctuation found in the manuscripts. Capital letters are only used at the beginning of a sentence, not in the case of proper names or nomina sacra. All numbers, either numeral or cardinal, are presented as ordinal, written in Roman or Arabic numbers depending on what is found in the manuscript. Abbreviations are silently expanded.

Most orthographic variants were kept, with the exception of "t"/"c". Orthographic variants kept are single or double consonants ("b"/"bb"; "c"/"cc" etc.), the addition of "h" after a consonant or before a vowel ("c"/"ch", "t"/"th", "a"/"ha", "i"/"hi", "y"/"hi" etc.) and letter combinations that sounded roughly equivalent in medieval Latin ("d"/"t"; "f"/"ph"; "i"/"y"; "m"/"n").

3.6 Phylogenetics Software Pipeline

The texts were all in ".txt" format and processed through the python package CollateX which generates the best alignment of the tokens in the text. As part of the preprocessing for using the ready-made phylogenetics package PAUP*, the texts were then reformatted into the Nexus format by ordering aligning the words at each position. Each variant of the token in each position was then assigned symbol $(A, B, C, ...)$, such that if say, three different variants of a given word appear in the texts, then the symbols A,B, and C appear in the Nexus file at the same position, identical words at the same position are assigned the same symbol, whereas missing tokens are represented by a "-". Modern philologists typically keep punctuation (hyphens, commas, periods etc.) since in some cases they may be informative. Having said that, punctuation is typically heavily dependent on the scribes' writing style and thus some philologists remove punctuation since its more random nature can be seen as adding noise to the data [1]. For the phylogenetic analysis we keep the punctuation as part of words as in Figure 16.



```
+---+-----+-------+-------+-----+-------+------+-----+------+------+
| A | The | quick | brown | fox | jumps | over | the | -    | dog. |
| B | The | -     | brown | fox | jumps | over | the | lazy | dog. |
+---+-----+-------+-------+-----+-------+------+-----+------+------+
```

Figure 16: Two sentences aligned using Collatex.

Figure 17 is an example of a Nexus datablock. "ntax" corresponds to the number of taxa, "nchar" corresponds to the number of characters in the aligned sequences which must be the same for all taxa (in our case the symbols correspond to the word variants and "-"'s correspond to gaps in the alignment) and "format symbols" is the list of symbols used to denote the words.

```
#NEXUS
BEGIN DATA;
    dimensions ntax = 5 nchar = 10;
    format symbols = "ABCD" labels = left;
    matrix
        Taxon1  AABAAAAAAB
        Taxon2  AABBA-BABA
        Taxon3  AAABAACACA
        Taxon4  AAA-AAABCA
        Taxon5  AAABAACBDA;
end;
```

Figure 17: Example of a Nexus File Data Block.

The nexus files are then fed into phylogenetics software such as PAUP* and the Newick tree outputs are then read using the Bio.Phylo package [21] in python, converted to Networkx

[22] objects and finally compared to the true stemma via the averaged signed distance since the other measures of accuracy are not ideal in this case due to the fact that phylogenetic tree outputs contain latent nodes.

In phylogenetics one often deals with missing data. Typically, these values are replaced by character states to obtain the best possible score. An alternative treatment of missing data replaces all the missing entries with the same new state, and then seeks a tree that optimises the criterion. If a dataset contains too many missing characters, the corresponding positions are removed before the tree is computed. In stemmatology missing data could be due to either destruction of part of a manuscript or due the scribe skipping certain words or punctuation, in which case the changes are informative. In the case that large sections of a manuscript is missing one may opt to either remove the variant, or to only use the available part. Another option would be to impute the missing sections from the closest manuscripts. In PAUP* distances are calculated based only on the available data, and positions with missing data are excluded from the calculation. For maximum parsimony, however, they are imputed such that the minimum total Hamming distance is achieved.

There does not seem to exist a stemmatological tool which allows the conversion of a collation to a distance matrix or to pseudo-DNA, which would be required to easily produce distance matrices or trees using bioinformatics software [1]. Thus, this pipeline could potentially be a useful setup for philologists. For the PAUP* software analysis we use the default settings which seemed to work quite well. These methods were least squares, minimum evolution, maximum parsimony, neighbour joining, neighbour joining consensus tree on 100 bootstrap replicates, UPGMA and UPGMA consensus tree on 100 bootstrap replicates.

3.7 NLP Distance Matrix Pipeline

To generate the synthetic stemmata we use a pipeline similar to that suggested in [9] as shown in Figure 18. Using a variety of different methods, the distance matrix is calculated from which the stemma's structure is reconstructed. The minimum cost heuristic is then used to root the stemma. For simplicity we assume that the stemmata have only a single root.

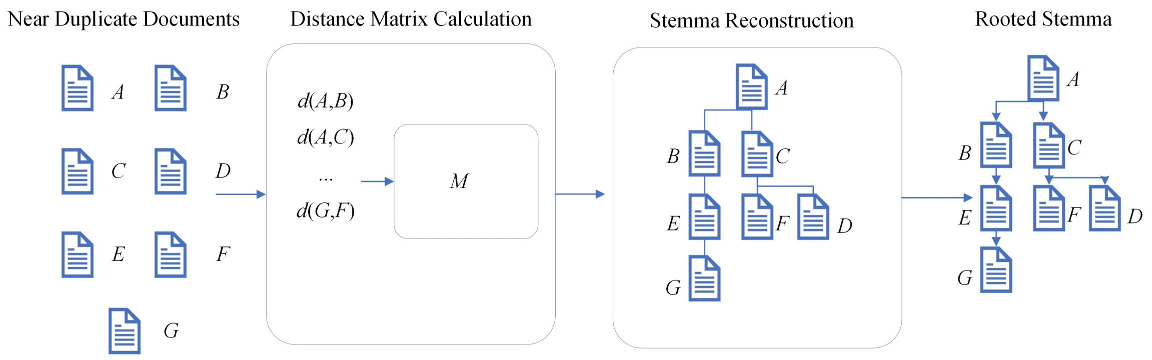


Figure 18: Pipeline for tree estimation using distance matrices calculation. The input is a set of documents, whilst the output is a rooted tree.

Given N near-duplicate documents, an $N \times N$ symmetric dissimilarity matrix M is calculated and used to construct an undirected unrooted tree using a minimal spanning tree. Finally, the minimum cost heuristic is used to root the tree using a depth first search.

One main bottleneck of these experiments is that it takes a considerable amount of time to generate the artificial datasets. A dataset has to be created from scratch for each set of transformation parameters as well as for different numbers of nodes. Higher editing limits and numbers of nodes require more time. Thus, a file management system was created to save the generated datasets so that one need not start from scratch each time. The correct stemma is also saved with the artificially generated texts. The reconstructed stemma is then computed from the texts and saved in the same folder. Finally, the reconstructed stemma is then compared with the correct stemma.

3.8 Pre-trained word vectors.

For Word2Vec a pre-trained the 300 dimensional 'google-news-300' model with 3 million tokens from [13] was used, for Glove, the 300 dimensional Glove embeddings trained on 42 billion tokens from [15] and for Fastext, the model was trained on the 'crawl-300d-2M-subword' Common Crawl sub-word dataset from [33]. We use these pre-trained models on such large datasets since the quality and robustness of the embeddings directly depend on the amount and variety of data they are trained on. Larger datasets provide a broader range of contexts in which words appear. This variety helps the model learn more accurate representations, capturing subtle differences in meaning and usage. In large corpora, words with multiple meanings (polysemes) appear in different contexts, aiding the algorithm in learning distinct representations for each sense of the word. Training on large datasets helps the model avoid over-fitting to idiosyncrasies in a small data sample. A well-trained model on a large corpus generalises better to new, unseen texts. Larger datasets are also more likely to include a wider range of vocabulary, including rare words and phrases which is particularly useful when using word embedding models which are incapable of generating embeddings for out of word vocabulary. A disadvantage of using such large models is that they require large amounts of memory and longer loading times.

3.9 Accuracy Measures

The tree construction methods are evaluated based on their success of finding a stemma that is close to the true stemma. For the case where there are missing manuscripts and non-tree-like structured stemmata we use the Average Signed Distance suggested in [34] and for complete datasets some of the measures used in multimedia phylogeny suggested in [35] and [9].

3.9.1 Root

This metric returns a 1 if the roots agree and 0 otherwise.

3.9.2 Leaves

This metric evaluates whether the leaves of the reconstructed stemma are the same as in the true stemma. A score close to 100% would mean a perfect match.

$$Leaves(T_1, T_2) = \frac{Leaves(T_1) \cap Leaves(T_2)}{Leaves(T_1) \cup Leaves(T_2)} \tag{51}$$

3.9.3 Depth

The depth distance is defined as the number of edges between the true root of the true stemma and the predicted root in the reconstructed tree. Defining $dist(i, j, T)$ to be the number of edges between nodes i and j on the tree T, the Depth is then:

$$Depth(T_1, T_2) = dist(Root(T_1), Root(T_2), T_2). \tag{52}$$

The less nodes between the true root and the predicted root, the better.

3.9.4 Indirected Edges

This measure evaluates the edges. Thus, should an edge connect two nodes in the reconstructed tree and in the the original, then it is considered correct:

$$Edges(T_1, T_2) = \frac{Edges(T_1) \cap Edges(T_2)}{n - 1} \tag{53}$$

A score close to 100% would mean a perfect match.

3.9.5 Directed Edges

This measure evaluates the edges along with their direction. Thus, should a directed edge connect two nodes in the reconstructed tree and in the the original, then it is considered correct:

$$DirectedEdges(T_1, T_2) = \frac{DirectedEdges(T_1) \cap DirectedEdges(T_2)}{n - 1} \tag{54}$$

A score close to 100% would mean a perfect match.

3.9.6 Ancestry

This measure evaluates whether the ancestors (parents, grandparents etc.) of each root are the same in the reconstructed tree and in the true tree.

$$Ancestry(T_1, T_2) = \frac{Ancestry(T_1) \cap Ancestry(T_2)}{Ancestry(T_1) \cup Ancestry(T_2)} \tag{55}$$

A score close to 100% would mean a perfect match. In our implementation, the intersection is computed between the nodes which have the same ancestors up to the root, but not necessarily in the same order. Moreover, all the ancestors of a particular node must match with those of the corresponding node in the true stemma.

3.9.7 Average Signed Distance

In situations where there are missing manuscripts or when one uses methods which include internal nodes, two arbitrary latent tree structures are being compared and thus the usual measures such as counting the number of shared edges do not apply since there is no one-to-one correspondence between the latent nodes in the true stemma and the estimated one. Moreover, the tree generating algorithms which place the extant taxa as leaves makes the 'Leaves' measure redundant. It is not guaranteed that the number of latent nodes will be the same, let alone their positions in the stemma. Thus, we employ the average sign similarity score [34].

This measure depends on the number of edges between pairs of nodes and ignoring the edge weights. For each pair of nodes A, B, the true distance, $d(A, B)$ is defined as the number of edges on the shortest path between A and B in the true stemma. Thus $d(A, B) = d(B, A)$. Similarly, the same quantity computed from the predicted stemma is denoted by $d'(A, B)$. Whenever the predicted stemma is correct, $d'(A, B) = d(A, B)$ for all pairs of nodes, and otherwise the two values differ for some A and B. Given three nodes A, B, and C, one can measure the distances $d(A, B)$ and $d(A, C)$. We consider which one of these two distances is greater than the other, or whether they are equal. Let $sign(d(A, B) - d(A, C))$ be the sign of the difference between the two distances, so that the index:

$$u(A, B, C) = 1 - \frac{1}{2}|\text{sign}(d(A, B) - d(A, C)) - sign((d'(A, B) - d'(A, C))|, \tag{56}$$

so that:

$$u(A, B, C) = \begin{cases} 1 & \text{if } sign(d(A, B) - d(A, C)) = sign(d'(A, B) - d'(A, C)) \\ 0 & \text{if } sign(d(A, B) - d(A, C)) = -sign(d'(A, B) - d'(A, C)) \\ 1/2 & \text{otherwise.} \end{cases}$$

The average sign similarity score is the average of over all distinct observed nodes corresponding to the extant manuscripts (but not the latent nodes) given that none of the three nodes are the same. This measure can be used with any pair of graphs, given that both include all the observed nodes and may include any number of additional nodes, which need not be identical for the correct and the proposed stemma. The computation of the distance requires that the pairwise distances between all nodes are computed and then used to calculate the average. This has a computational complexity of $O(n^3)$ [34].

4 Results

This section presents the results from the experiments described in chapter 3. The first half deals with different experiments conducted with the artificial data sets. Due to computational constraints, only shorter texts were used. The second half of this chapter deals with the benchmark dataset Parzival and Phareta Fidei.

4.1 Artificial Datasets

In this section the results for the synthetic datasets generated by editing the Reuters news articles via various transformations are described. We start with the synthetic datasets generated from Reuters_50_50 news articles, then proceed to the benchmark datasets Parzival and Heinrichi, and finally Phareta Fidei.

4.1.1 Misspellings with 5% Edit Limit

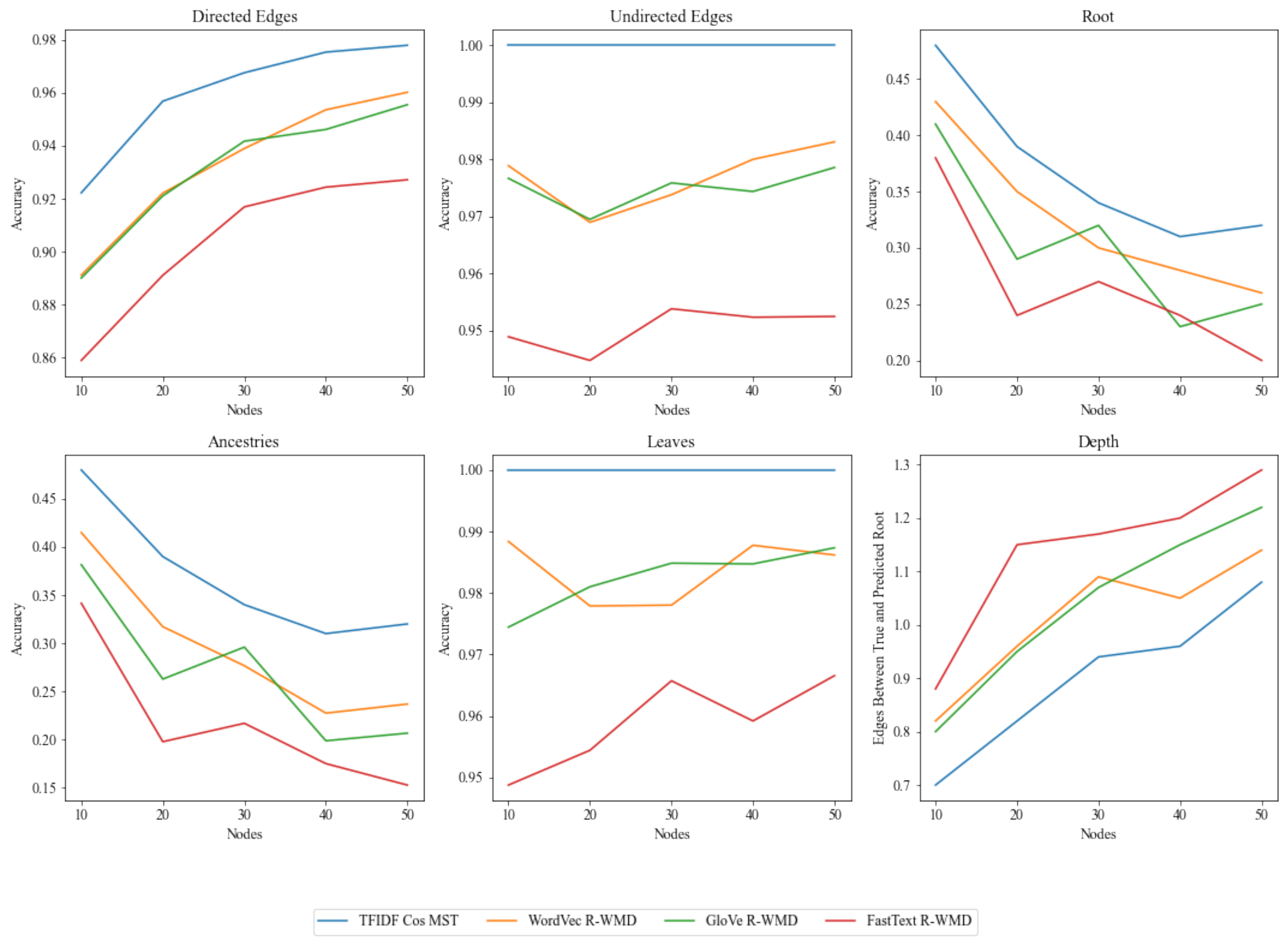


Figure 19: Accuracy rate averages of 100 examples per data point for the root, leaves, depth, directed and undirected edges vs number of nodes for stemmata where the differences between the texts are misspellings (at rate 60% of edits) and correction of misspellings (40%).

In Figure 19 the undirected edges and leaves seem not to vary too much against the number of nodes whereas, directed edges seem to improve whereas all others seem to worsen. For this dataset, the word embeddings performed nearly as well as TF-IDF, even managing to correctly predict the root in a good portion of the stemmata.

4.1.2 Synonyms and Additives with 5% Edit Limit

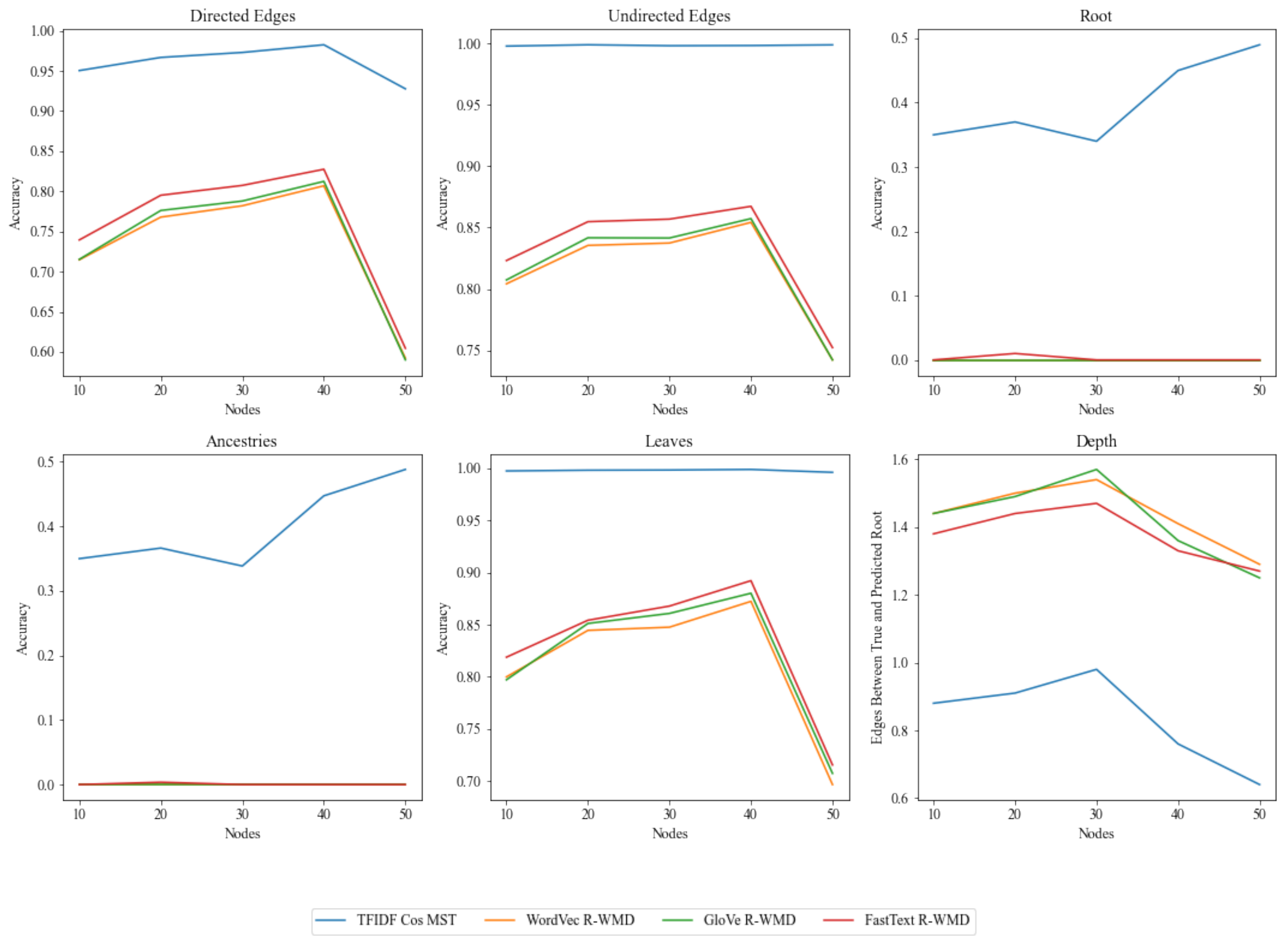


Figure 20: Accuracy rate averages of 100 examples per data point for the root, leaves, depth, directed and undirected edges vs number of nodes where the differences between the texts are synonym exchange (at rate 30% of edits), addition of adverbs/adjectives (40%) or removal of adverbs/adjectives (30%).

In Figure 20 the the number of nodes did not seem to make much of a difference to the results, with 50 node trees being an outlier. The root was almost never correctly guessed for the word embedding methods, thus, the ancestries were not correctly predicted. Still, the root is off by only a few nodes at most, meaning these methods are still quite feasible.

4.1.3 Paraphrasing of Sentences 20 nodes

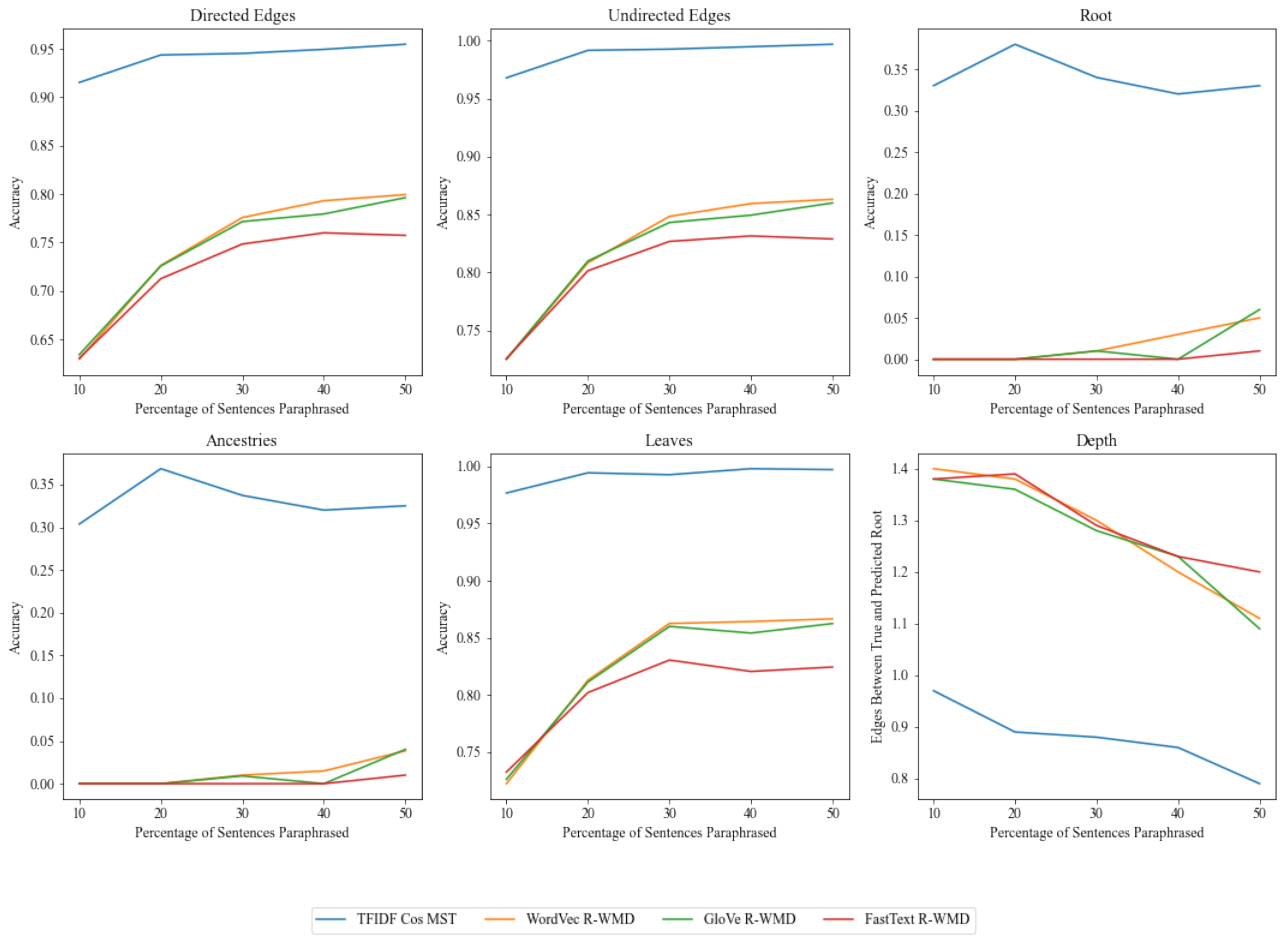


Figure 21: Accuracy rate averages of 100 examples per data point for the root, leaves, depth, directed and undirected edges vs portion of sentences paraphrased. All stemmata had 20 nodes, with the edit portion changed from 10% to 50% in steps of 10%.

In Figure 21 the edit portion seemed to improve the results. Once again, the root was almost never correctly guessed for the word embedding methods, thus, the ancestries were not correctly predicted. Still, the root is off by only a few nodes at most, meaning these methods are still quite feasible. The improving result for the other accuracy measures with the edit portion is probably due to the paraphrased sentences being very similar to the originals, sometimes with only a few words being changed, making the documents very difficult to tell apart. With higher portions of edited sentences, the documents become more distinguishable.

4.2 Parzival

This section serves to display the results for Parzival. In the PAUP* outputs, the internal nodes are labelled as arbitrary integers.

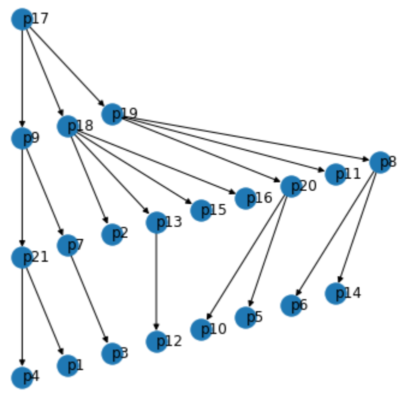


Figure 22: Parzival Correct Stemma. Nodes 17-21 are missing and are thus named arbitrarily.

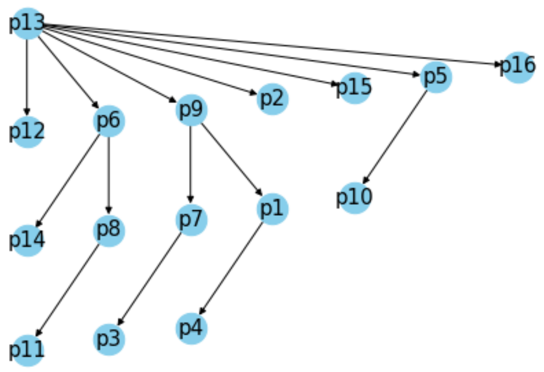


Figure 23: Parzival predicted stemma using TF-IDF with 1-,2-,3-grams.

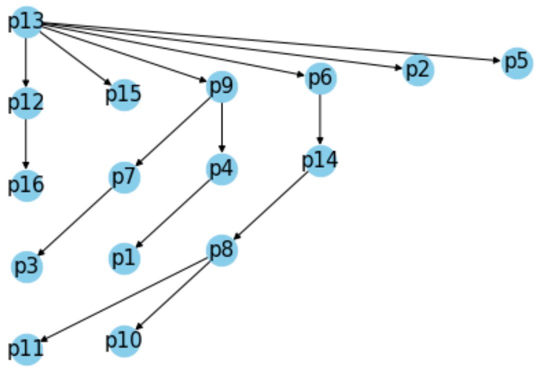


Figure 24: Parzival predicted stemma using the cosine distances between the Word2Vec word vectors.

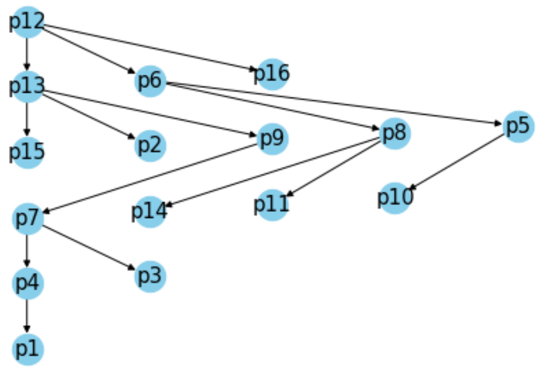


Figure 25: Parzival predicted stemma using the reduced word mover's distances between Word2Vec vectors.

In Figure 22 is the correct stemma for Parzival, where nodes 17-21 are missing. TF-IDF, Word2Vec with cosine and reduced word mover's distance in Figures 23-25 gave fairly believable results. Nodes which neighbour each other in the correct stemma, also tended to neighbour each other in these trees for example p_{12} neighbours p_{13} and p_7 neighbours p_3 in all of these. The main problem seems to be when the nodes are separated by missing nodes. Having said that, nodes separated by missing nodes in the true stemma usually neighbour the closest available observed node. For example p_1 neighbours p_4 in Figures 23-25. The root was predicted to be p_{12} by Word2Vec using reduced word mover's distance. All other NLP methods gave p_{13} as the root which is one node closer to the actual root in the correct stemma.

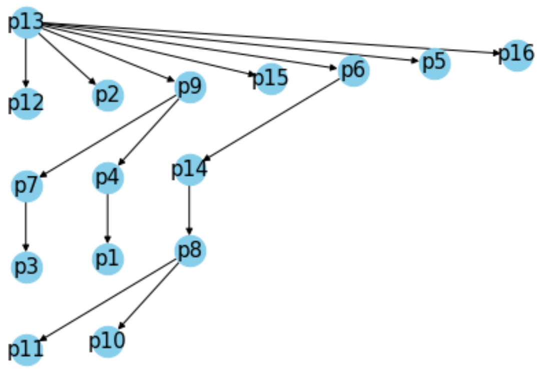


Figure 26: Parzival predicted stemma using the word mover's distances between Word2Vec vectors.

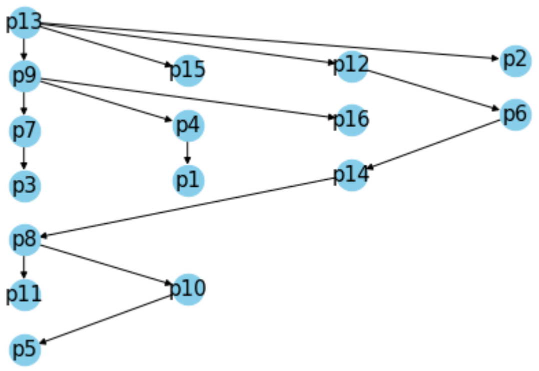


Figure 27: Parzival predicted stemma using the cosine distances between the FastText word vectors.

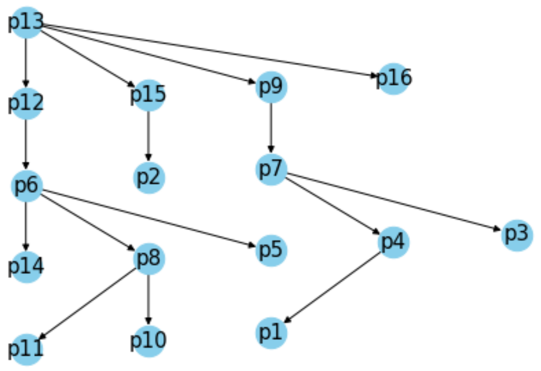


Figure 28: Parzival predicted stemma using pre-trained FastText word vectors and reduced word mover's distance.

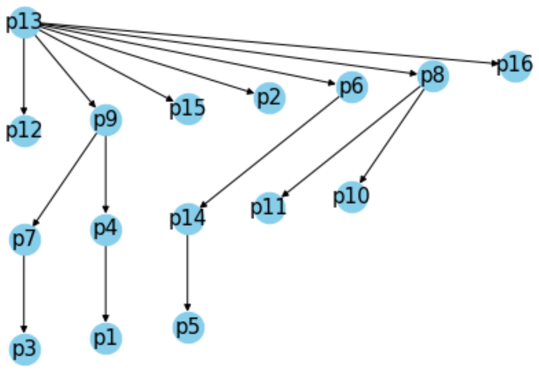


Figure 29: Parzival predicted stemma using pre-trained FastText word vectors and word mover's distance.

The results from Word2Vec with word mover's distance, FastText with reduced word mover's distance, word mover's distance and cosine distances shown in Figures 26-29 gave very similar results to Figures 23-25. Once again, nodes separated by missing nodes tend to end up paired with the closest available node. Node p_8 is taken as the parent of p_{11} and p_{10} in all of these examples. In the stemma they are separated by missing p_{19} and p_{20}.

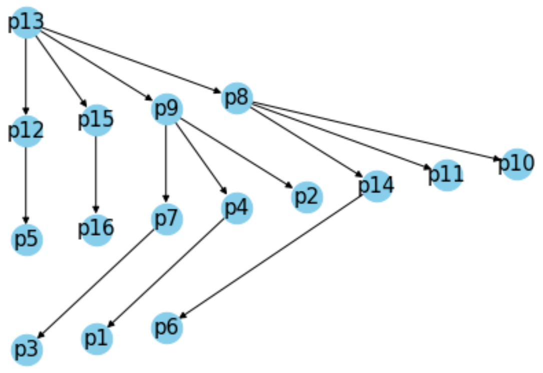


Figure 30: Parzival predicted stemma using the cosine distances between the Glove word vectors.

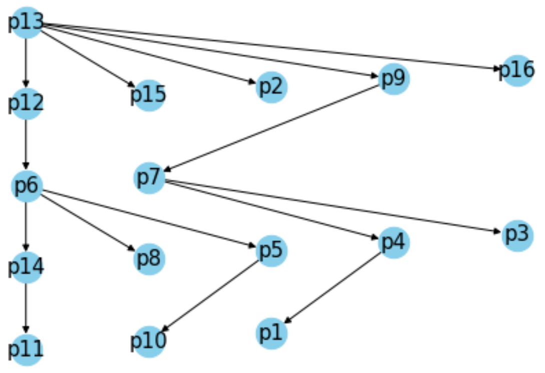


Figure 31: Parzival predicted stemma using pre-trained Glove word vectors and reduced word mover's distance.

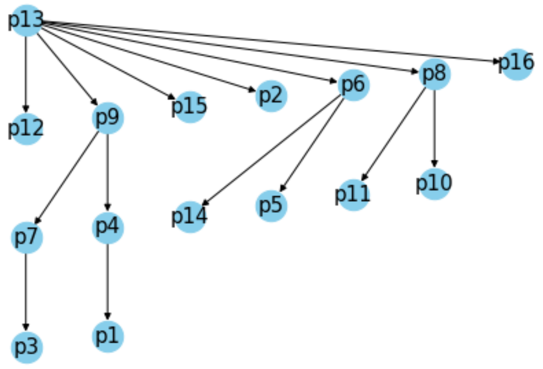


Figure 32: Parzival predicted stemma using pre-trained Glove word vectors and word mover's distance.

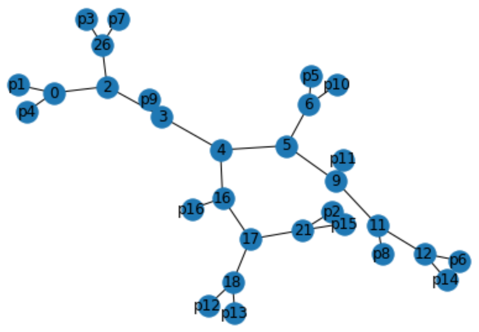


Figure 33: Parzival predicted stemma ordinary least squares in PAUP*.

The results from Glove with all three distance methods considered shown in Figures 30-32 gave very similar results to Figures 23-25. Once again, nodes separated by missing nodes tend to end up paired with the closest available node. Node p_8 is also taken as the parent of p_{11} and p_{10} in all of these examples. It seems that three clusters of nodes emerge from all the NLP method predictions. p_2, p_{12}, p_{13}, p_{15} and p_{16} tend to cluster together, so do p_9, p_1, p_4 p_7 and p_3. The third cluster seems to be p_5, p_{10}, p_6, p_8, p_{14}, p_{11}. These groupings also appear in the stemma.

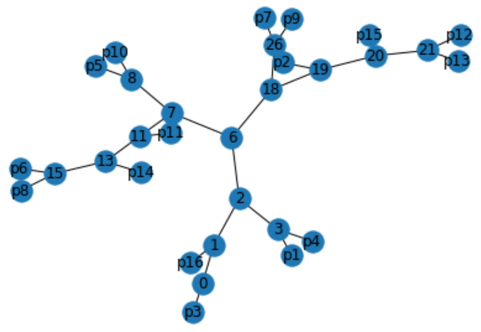


Figure 34: Parzival predicted stemma using UPGMA in PAUP*.

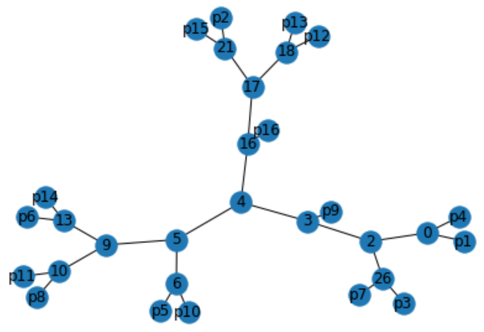


Figure 35: The predicted stemma for Parzival using neighbour joining in PAUP*.

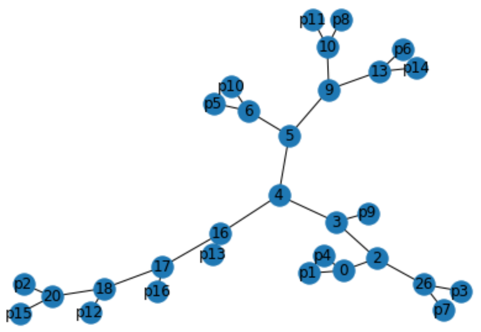


Figure 36: The predicted stemma for Parzival using maximum parsimony.

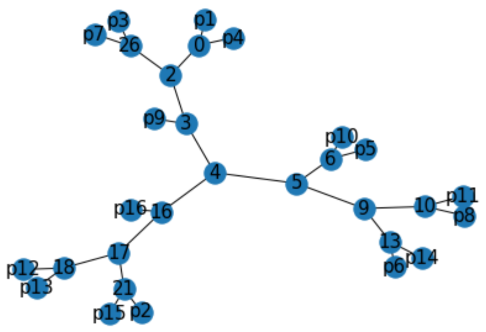


Figure 37: The predicted stemma for Parzival using minimum evolution in PAUP*.

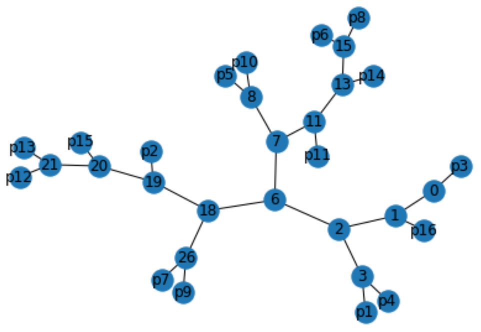


Figure 38: Parzival UPGMA consensus tree using on 100 bootstrap replicates in PAUP*.

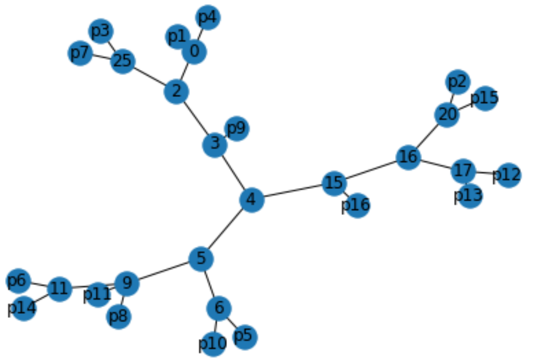


Figure 39: Parzival neighbour joining consensus tree using on 100 bootstrap replicates in PAUP*.

Figures 33-39 show the results for Parzival using PAUP* software. All results were very similar to one another. UPGMA is the only phylogenetic method which yields the root. In both the basic UPGMA and the bootstrap consensus, p_3 was found out to be the root (this is identified as the node connected to an internal node with degree 2). This is separated by the true root by p_9 and p_7. In all cases, nodes which appear as neighbours in the true stemma tend to appear close together in the trees generated by these methods. The same groupings tend to appear in all of these methods as well. UPGMA and UPGMA with bootstrap seems to be the worst violators of these patterns. p_7 and p_3 appear very far from one another. Meanwhile, p_{16} is grouped with p_3 even though they should be very far apart. This seems to be the main mistake this algorithm made, with all other nodes being close to nodes which occur close in the true stemma.

Method	**Average Signed Distance (%)**
Least Squares	83.3
Minimum Evolution	81.5
Maximum Parsimony	77.8
Neighbour Joining	81.5
Neighbour Joining + Bootstrap	84.4
UPGMA	71.6
UPGMA + Bootstrap	71.6
TF-IDF 1-,2-,3-gram	71.6
Word2Vec+Cosine	70.1
Word2Vec+WMD	71.6
Word2Vec+RWMD	76.8
Glove+Cosine	75.5
Glove+WMD	74.9
Glove+RWMD	77.9
FastText+Cosine	75.8
FastText+WMD	74.4
FastText+RWMD	76.9

Table 1: Average Signed Distance between the correct stemma of Parzival and the stemmata predicted by PAUP*, and minimum spanning trees using cosine distances between the word embedding sums of each document, Relaxed Word Mover's Distance (RWMD), Word Mover's Distance (WMD) and Text Frequency Inverse Document Frequency.

In Table 1 are the accuracy rates for all methods considered. The best performer is neighbour joining with bootstrapping using the default settings in PAUP* seemed to give very good results. The bootstraps reported here were done with 100 replicates since improving the number of replicates did not improve the result.

4.3 Heinrichi

Method	**Average Signed Distance (%)**
Least Squares	57.2
Minimum Evolution	59.5
Maximum Parsimony	67.2
Neighbour Joining	63.3
Neighbour Joining + Bootstrap	59.6
UPGMA	52.5
UPGMA + Bootstrap	56.7
TF-IDF 1-,2-,3-gram	82.3

Table 2: Average Signed Distance between the correct stemma of Heinrichi and the stemmata predicted by PAUP* algorithms, and minimum spanning trees using cosine distances between the and Text Frequency Inverse Document Frequency vectors.

Since the phylogenetic methods performed very poorly on this dataset, we only plot the result for the TF-IDF with 1-,2-,3-grams which gave a very good result as can be seen in Table 2. Moreover, the predicted root *Ba* is separated from the real root (which in this case is missing), by only two missing nodes as can be seen in Figures 40 and 41. Surprisingly, even though this tradition is very large, is missing nearly half of its nodes and has five cases of contamination (9 to *A*, 24 to 18, 24 to *Ca* and 24 to 27, TF-IDF with 1-,2-,3-grams gave an exceptionally good prediction.

The first main clusters in the correct stemma were *O*, *P*, *V*, *Ba*, *I*, *J*, *Da*, *I*, *J*, *T*, *S*, *Ae* and *W* (starting from 24). The second cluster was *N*, *X*, *H*, *Cd*, *E*, *C*, *Be*, *F*, *Ca*, *Bd* and *Bb* (starting from node 12). The third cluster was *Z*, *Ad*, *Cb*, *Cc*, *G*, *Ab*, *Ce*, *R*, *B*, *Cf*, *M*, *A*, *L*, *K* (starting from node 1).

In the TF-IDF tree there are three distinct clusters: *O*, *P*, *N*, *F*, *H*, *C*, *Cd*, *E*, *V*, *Ba*, *I*, *J*, *Da*, *I*, *J*, *T*, *S*, *Ae* and *W*. A second grouping was *Z*, *Ab*, *Ad*, *Cb*, *R*, *Ce*, *Ac*, *G*, *Cc*, *A*, *B*, *K*, *M*, *Cf* and *L*. *V*, *Ca*, *J* *I* and *Da* were also correctly grouped together. *P* neighbours *Ba* in the reconstructed stemma which makes sense since they are separated by one missing node in the true stemma. Thus, the high accuracy rate in Table 2 is clearly visible in this graph.

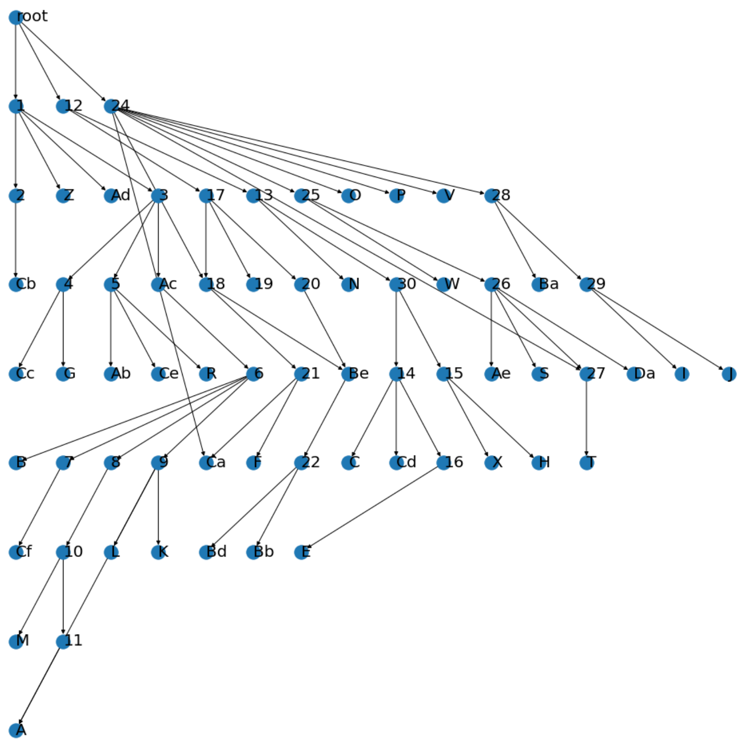


Figure 40: Heinrichi correct stemma. Missing manuscripts are labelled as arbitrary numbers whereas observed manuscripts are denoted by letters.

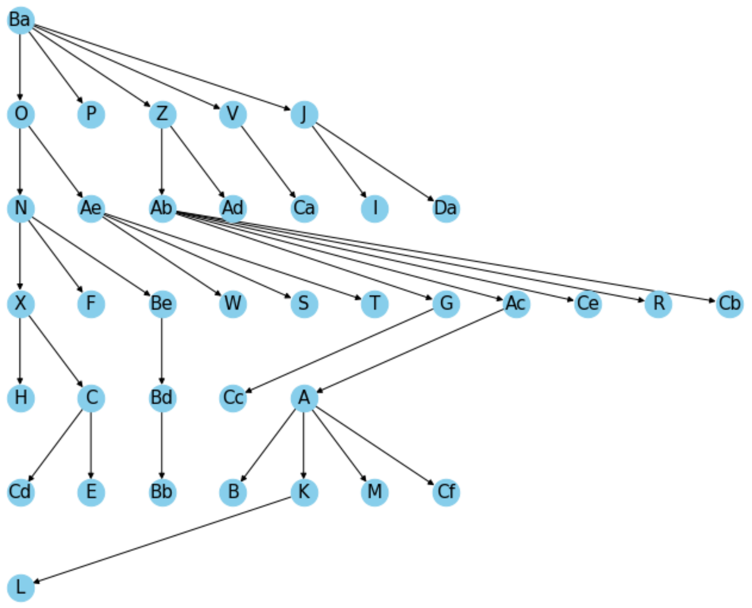


Figure 41: Heinrichi minimum spanning tree generated using TF-IDF with 1-,2-,3-grams.

4.4 Phareta Fidei

Witness	Name in Graph
Bamberg, Staatliche Bibliothek, Theol. 109, fol. 120r-131v (1401-1500, Bamberg)	Bamberg_Bamberg
Berlin, Staatsbibliothek Preussische Kulturbesitz, germ. Qu. 1577, 97r-112v (1437, Sulczmatt)	Berlin_Sulczmatt
Berlin, Staatsbibliothek Preussische Kulturbesitz, lat. Fol. 772, 185r-200v (1401-1500, Trier)	Berlin_Trier
Brno, Statni Vedecka Knihovna, Mk 43 (II. 148), 1r-17r (1301-1400, Rokycany)	Brno_Rokycany
Ceske Budejovice, Vyssí Brod, 123, 156-226 (1410)	CeskeBudejovice
Erfurt, Universitätsbibliothek, Ampl. 4 82, 131-148 (1333-1400, Erfurt)	Erfurt_Erfurt_13331400
Erfurt, Universitätsbibliothek, Ampl. 4 116, 204r-216r (1401, Erfurt)	Erfurt_Erfurt_1401
Graz, Universitätsbibliothek, 312, 2r-15 (1378-1418, Seckau)	Graz_Seckau
Graz, Universitätsbibliothek, 873, 129v-148v (1401-1450, Neuberg an der Mürz)	Graz_Murz
Klosterneuburg, Bibl. Des Chorherrenstifts, 933, no 49, 239-248 (1301-1500, Krems)	Klosterneuburg_Krems
Köln, Hist. Archiv d. Stadt, GB 4 66, 24r-39v (1401-1425, Köln)	Koln_Koln
Kremsmunster, Stiftsbibliothek, 99, 180r-202v (1401-1450, Stein bei Krems)	Kremsmunster_SteinbeiKrems
London, British Library, Royal 8.F.XI, 43r-55v (1401, Köln)	London_Koln
Mainz, Wissenschaftliche Stadtbibliothek Mainz, Hs I 130, 24v-55v (1401-1450, Mainz)	Mainz_Mainz
Madrid, Biblioteca Nacional de Espana, INC/2661 (1488-1490, Zagaroza)	Madrid_Zaragoza
München, Bayerische Staatsbibliothek, clm 12389, 196r-221v (1301-1500 Raitenbuch/Rottenbuch)	Munich_Raitenbuch
Prague, Národní knihovna, I.B.7, 312r-323v (1401-1500)	Prague_IB7
Prague, Národní knihovna, X.B.11, 1r-45r (1401-1500)	Prague_XB11
Rome, Biblioteca Casanatense, Ms. 159, 5v-35v (1511, Wien)	Rome_Wien
Rouen, Bibliotheque Municipale, Ms. Leber 59, 1r-107 (1301-1400)	Rouen
Seitenstetten, Stiftsbibliothek, 268, 157r-192v (1401-1450, Seitenstetten)	Seitenstetten_Seitenstetten
Trier, Stadtbibliothek, 964/1158, 211v-228r (1401-1500, Trier)	Trier_Trier_211v_228r
Trier, Stadtbibliothek, 1296/554, 78r-95v (1401-1500, Trier)	Trier_Trier_78r_95v
Wien, Österreichische Nationalbibliothek, 362, fol. 181v-188v (1301-1333, Lilienveld)	Wien_Lilienveld
Wien, Österreichische Nationalbibliothek, 812, 107r-121v (1301-1350)	Wien_812
Wien, Österreichische Nationalbibliothek, 4180, 26-42v (1401-1500, Essingen)	Wien_Essingen
Wien, Österreichische Nationalbibliothek, 4213, 134r-156r (1401-1500, Wien)	Wien_Wien
Wien, Österreichische Nationalbibliothek, 4396, 1r-24v (1401-1500)	Wien_4936

Table 3: The textual witnesses in Phareta Fidei 28 and their names in the generated graphs. The naming convention is ”City, Library, Variant ID, Year Written, Place of Origin”.

4.4.1 Phareta Fidei 10

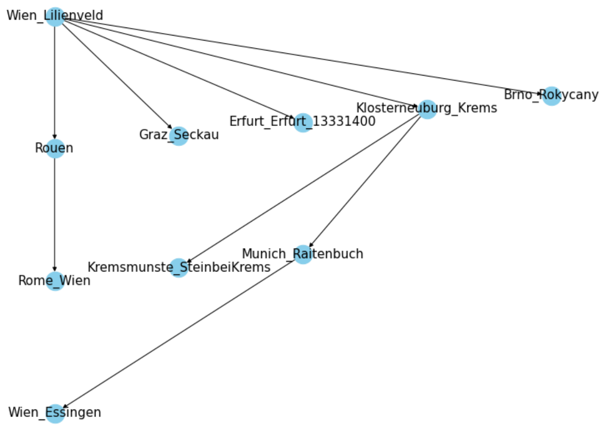


Figure 42: Phareta Fidei minimum spanning tree generated using TF-IDF with 1-,2-,3-grams.

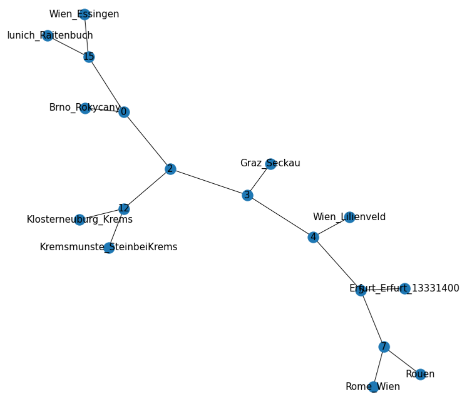


Figure 43: Phareta Fidei tree using least squares in PAUP*.

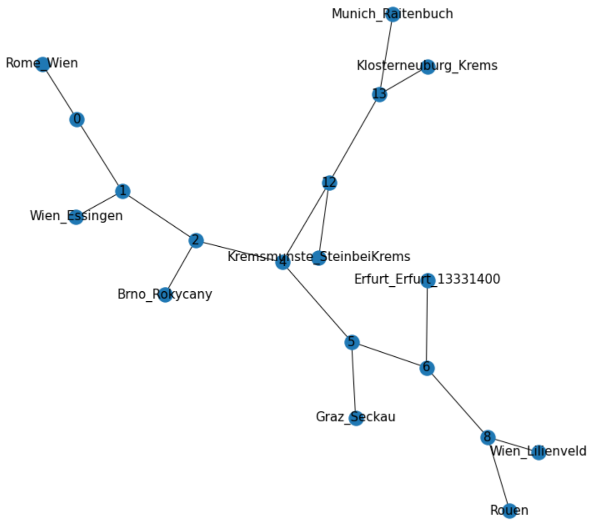


Figure 44: Phareta Fidei using UPGMA in PAUP*.

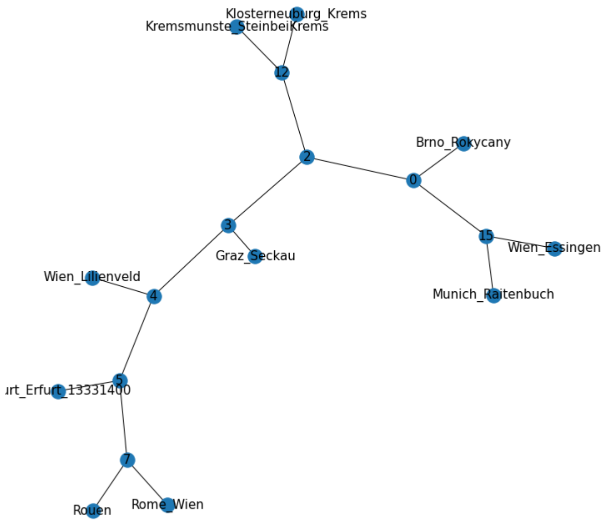


Figure 45: Phareta Fidei predicted stemma using neighbour joining in PAUP*.

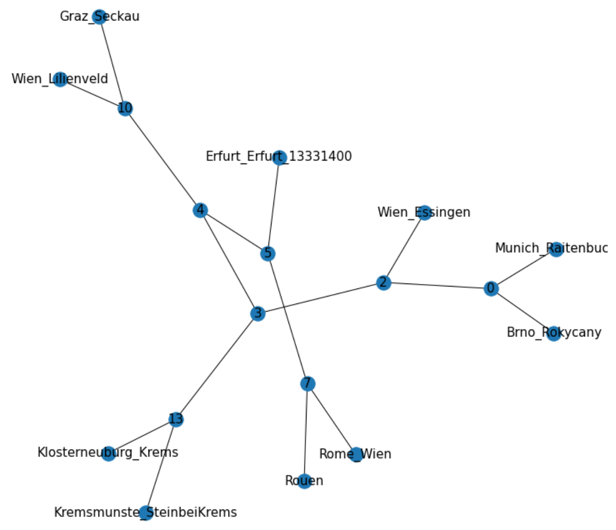


Figure 46: Phareta Fidei prediction using maximum parsimony in PAUP*.

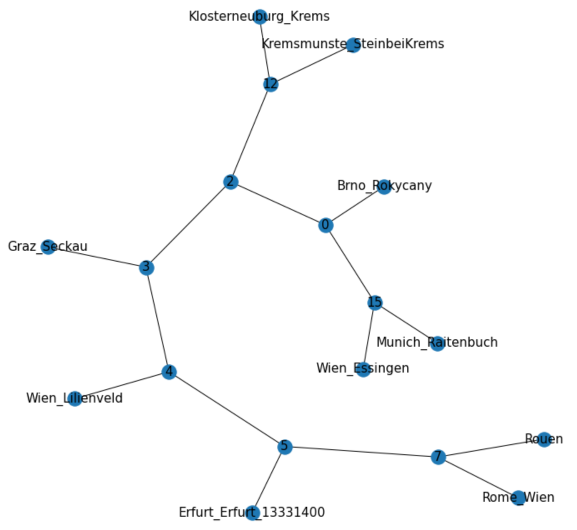


Figure 47: Phareta Fidei prediction using minimum evolution in PAUP*.

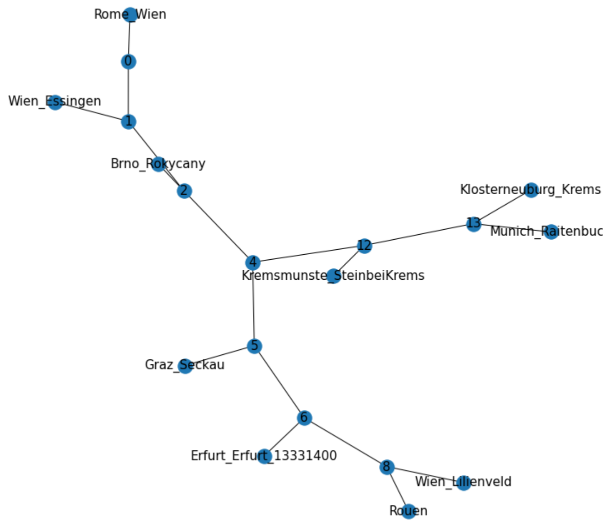


Figure 48: Phareta Fidei UPGMA consensus tree on 100 bootstrap replicates.

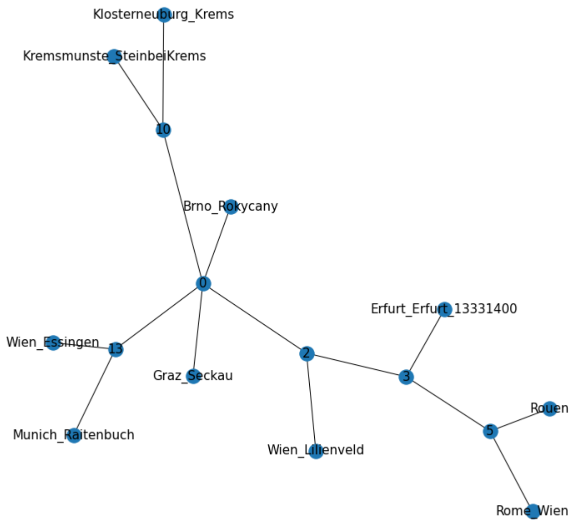


Figure 49: Phareta Fidei neighbour joining consensus tree on 100 bootstrap replicates.

Figures 42-49 show the results for the methods used to predict the stemma for the 10 variant Phareta Fidei corpus. The philologist who provided these manuscripts predicted that Rouen and Roma are related, Klosterneuburg and Kremsmunster are related. Encouragingly, Rouen neighbours Roma whilst Klosterneuburg neighbours Kremsmunster in all methods except UPGMA and UPGMA with bootstrap. Graz Sackau tends to appear close to Wien Lilienveld.

UPGMA gives the Rome variant as the root both with and without bootstrap. Out of this set the root was not obvious to the philologist thus little can be said about the accuracy.

4.4.2 Phareta Fidei 28

The philologist who provided these manuscripts predicted that Rouen and Roma are related, the 2 Graz witnesses seemed related to Trier 964, Klosterneuburg and Kremsmunster are related as well and Prague X.B.II is closely connected to Seitenstetten and Wien 4396. Wien 4396 seemed to be the oldest variant from this group of manuscripts. Figure 50 agrees with all of these observations, with the exception that Graz Seckau was the root, although this is only two nodes away from Wien 4396. UPGMA managed to correctly predict the root as Wien 4396 both with and without bootstrap. However, it must be emphasised that the root of this tradition is not known for certain.

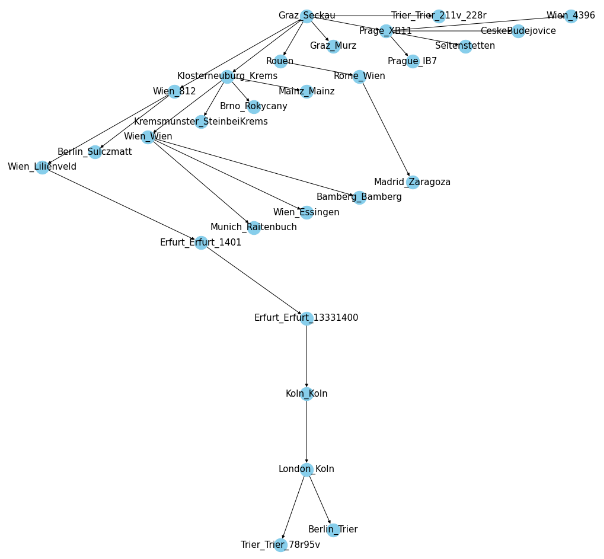


Figure 50: Phareta Fidei minimum spanning tree generated using TF-IDF with 1-,2-,3-grams.

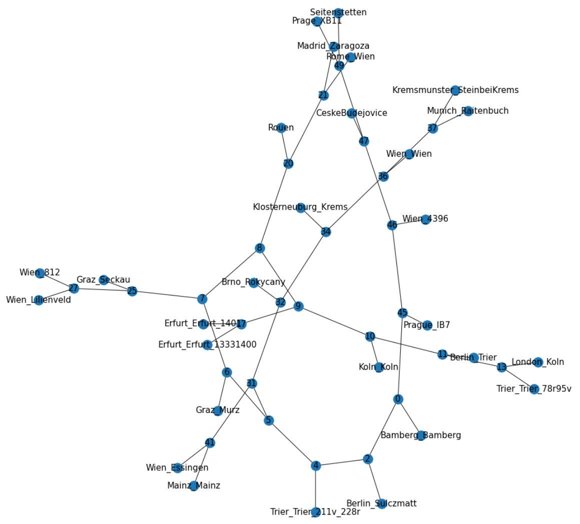


Figure 51: Phareta Fidei tree using least squares in PAUP*.

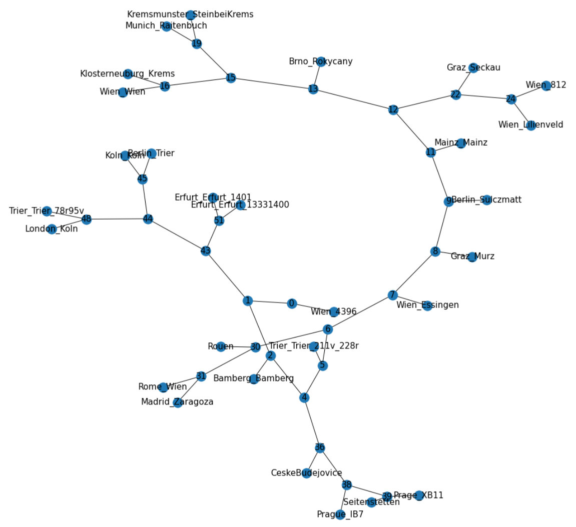


Figure 52: Phareta Fidei using UPGMA in PAUP*.

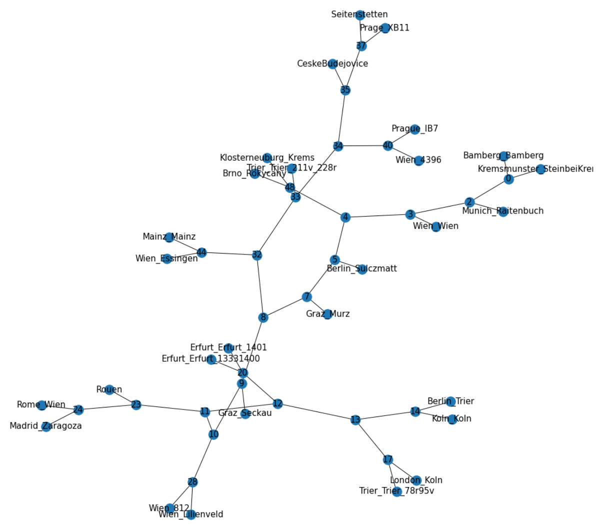


Figure 53: Phareta Fidei predicted stemma using neighbour joining in PAUP*.

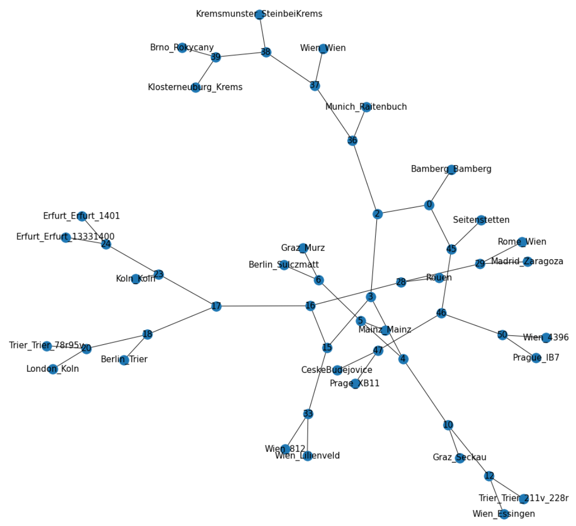


Figure 54: Phareta Fidei prediction using neighbour joining in PAUP*.

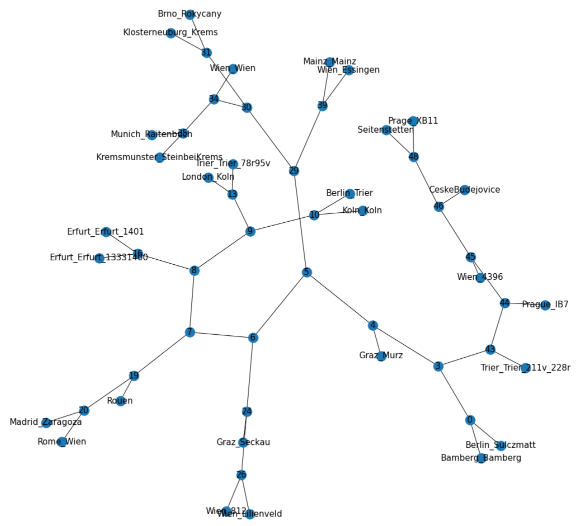


Figure 55: Phareta Fidei prediction using minimum evolution in PAUP*.

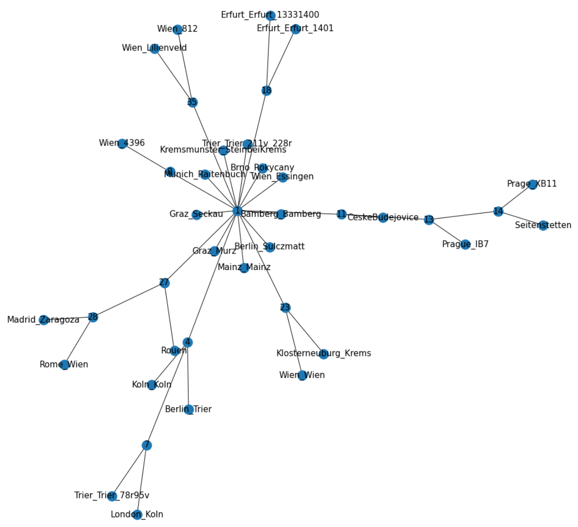


Figure 56: Phareta Fidei UPGMA consensus tree on 100 bootstrapreplicates.

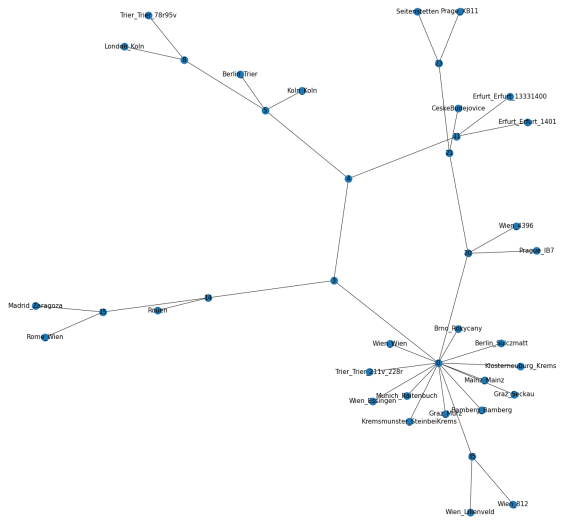


Figure 57: Phareta Fidei neighbour joining consensus tree on 100 bootstrap replicates.

In the bootstrap consensus graphs using neighbour joining and UPGMA (Figures 56 and 57), several nodes had their internal nodes collapsed, causing them to end up in a multifurcation, at the internal node leading to the root. This happens when these appear neighbouring the same node less than 50% of the trees generated from the bootstrap replicates since we use a 50% majority rule. Thus, they should be studied further by the domain expert. These are Trier-Trier 211v 228, Kremsmunster-Stein bei Krems, Brno-Rokycany, Wien-Essingen, Bamberg-Bamberg, Berlin-Suiczmatt, Mainz-Mainz, Graz-Murz, Graz-Seckau and Munich-Raitenbuch.

The manuscripts sometimes have the name of the city in which they were written, in the graphs this is specified as the second name. It seems that those written at the same place such as those written in Köln and Erfurt tend to appear either as neighbours or very close together which makes sense. The predictions made by the philologist did not seem to appear often in the PAUP* outputs. Klosterneuburg and Kremsmunster usually appear fairly close but not attached to the same internal node.

5 Discussion

In this section the results from the previous section are discussed, dataset by dataset. We start by discussing the artificial datasets generated from the Reuters news articles, followed by Parzival, Heinrichi and finally Phareta Fidei.

5.1 Artificial Datasets

In the synthetic data experiments (Figures 19-21), the performance of TF-IDF with 1-,2-,3-grams was compared to the trees based on the reduced word mover's distance between FastText, Word2Vec and Glove word embeddings. The tree complexity was increased from 10 to 50 nodes in steps of 10, and generated 100 trees for each increment at 5% edit limit for the misspellings and the synonym/additive exchange whilst for the paraphrasing experiment the the portion of sentences paraphrased was increased from 10 to 50% in steps of 10% with the number of nodes kept constant at 20. All three word embedding methods yielded very similar results in all cases. It seems that the accuracy increases with the number of nodes which is unexpected. TF-IDF with 1-,2-,3-grams seems to perform better than word embedding representations for all cases. Advanced methods such as pre-trained word embeddings with word mover distance did not yield the best results, although most of the indirected edges were correctly guessed in all experiments. This is most likely due to the fact that the synthetic texts generated were extremely similar to one another, thus, traditional text representations such as TF-IDF with n-grams are more able to identify the changes made via the transformations used. Word embeddings operate by mainly looking at changes in context, however, since the context is largely unchanged, the word vectors are not much affected overall, leading to the algorithm to struggle to tell the difference between the very similar texts. This can explain why the word embeddings did not perform well for the synonym exchange and adjective/adverb removal and addition dataset in 20.

Surprisingly, for the misspellings dataset in Figure 19, the word embeddings gave better results than for the paraphrasing and the synonym exchange dataset, even though word embedding methods are not designed to distinguish between spelling mistakes since these do not usually occur in the training vocabulary. Even more surprising was the fact that FastText achieved the lowest results in all categories, even though this is the only method which takes the spelling into account. This could be explained along similar lines to those in the previous paragraph. For misspellings, FastText should yield word vectors close to those of the correct spelling, thus, the word vectors do not change much from document to document, making it more difficult to distinguish them. For Glove and Word2Vec, out of vocabulary words are assigned zero vectors, thus it is easier to distinguish words and their misspellings.

Another surprise was that the word embedding methods also did not perform better on the paraphrased datasets as can be seen in Figure 21. This could again be explained by the fact that the context does not change much when paraphrasing and thus methods based on word frequency are better at detecting these changes. However, it was also noticed that the paraphrased sentences are usually very similar to the original sentences, usually only differing by a few words, sometimes with only one word being exchanged with a synonym. The edit proportion was based on the number of sentences rather than word count, thus the number of changes were not very consistent. In Figure 21, the accuracy measures for the word embedding methods seemed to improve with greater edit proportion, most likely due to the fact that when paraphrasing sentences, too few words were actually changed, leaving extremely similar documents which word embedding methods may not perform too well on.

It should be noted that the relaxed word mover distance was used for the artificial datasets. This was done due to computational constraints. Unfortunately, this reduces the accuracy of the measure [16]. In stemmatics, the tree is reconstructed using the tiniest differences which when using the relaxed word mover distance, may not be appropriately captured. In any case, its performance was not too disappointing. Most of the nodes were neighbours, however, the root was never correctly guessed, leading to bad performance for the other measures. Having said that, in stemmatology, the most important thing is that the indirect edges are correct since the root must always be derived by the scholar using prior knowledge such as the age of the words, the date of the documents and their own knowledge and intuition.

The ancestries measure seems to have been too strict for this case. Since it compares the ancestors of the nodes upto the root, if the root is predicted to be incorrect, then this guarantees an ancestry result of zero. The rooting heuristic seems to work quite well, even though it may not yield the root, it usually yields a node close to the root.

5.2 Parzival

For Parzival, the phylogenetic methods seemed to work best, especially neighbour joining and minimum evolution. UPGMA did not give too bad a result which is surprising since since it yields ultrametric trees, where the branch lengths are all equal from the root node and the leaves. Since different scribes make different mistakes at different rates, this assumption is rarely met.

The NLP methods did not perform as well at the PAUP* methods on Parzival, however, they were still very competitive. That they did not perform so well can be explained by the fact that most of the positional information is lost when encoding the texts as word vectors or documents as TF-IDF vectors. Having said that, the graphs produced by all NLP methods were quite similar and accurate, especially considering that the data had 5 nodes missing out of 21 and that minimum spanning trees cannot generate latent nodes. The mistakes tended to be where the latent nodes corresponding to the missing manuscript should have been, thus, had the entire dataset been available these algorithms probably would have performed much better. Surprisingly, the word embedding methods outperformed TF-IDF by a small margin which they did not for the synthetic experiments. This could be explained by the fact that when humans are copying texts, the changes they make may change the context more than when done randomly through some algorithm. Another surprise was that the relaxed word mover's distance outperformed the word mover's distance on this dataset when using all three word embeddings, although by very small margins. TF-IDF was already applied to this dataset in [9], however, the average signed distance was not applied. Word embeddings have never been applied to a real stemmatology dataset before, thus this is a new result.

One disadvantage of word vector methods, however, is that one needs to have large pre-trained models to get good results. For extinct languages such as Old Finnish these usually are not available and since spelling changed so much throughout history, the models may have been trained on datasets which lack the particular spellings used by the scribes in sufficient numbers. In any case it was encouraging to see modern NLP methods perform so well.

5.3 Heinrichi

The phylogenetic methods seem to not work very well in this case. This could be due to the large number of missing nodes, the contamination or the size of this dataset. On the other hand, Text Frequency - Inverse Document Frequency seems to have been able to predict the tree with a very high degree of accuracy as shown in Figure 41. Moreover, the predicted root *Ba* is separated from the real root (which in this case is missing), by only two missing nodes. This was already done in [9], however, the average signed distance was not applied. This may mean that for real world stemmata, NLP methods may be more robust than phylogenetics methods. This could also be due to the fact that in our implementation, the NLP methods yield minimum spanning trees which are more faithful to the multifurcating nature of stemmata.

5.4 Phareta Fidei

The expanded Phareta Fidei's minimum weight spanning tree computed from TF-IDF agreed with the relationships identified by the philologist who provided the dataset. There were only two variants in between the TF-IDF predicted stemma (Graz Seckau) and the root predicted by the philologist (Wien 4936). As discussed before, the location of the root should always be determined by a scholar using their knowledge and intuition. Having said that, the stemma for Phareta Fidei has never been rigorously studied and this only considers a very small portion of the entire tradition. Thus, the structure and the root are uncertain, however, it is quite encouraging to see that this method produces believable results. To the author's knowledge, this is the first real-world tradition to which natural language processing methods have been applied. Encouragingly, the relationships which the philologist noticed tended to occur in the stemmata as well, both in the 10 variant corpus and in the 28 variant corpus.

The PAUP* methods yielded similar results to TF-IDF for the 10 variant dataset. In this reduced dataset Rouen neighbours Roma whilst Klosterneuburg neighbours Kremsmunster in all methods but UPGMA with and without bootstrapping. The structure of the graphs was fairly consistent. Meanwhile, for the 28 variant dataset, the PAUP* predictions did not usually agree with the philologist's heuristics, although usually the number of nodes between the predicted pairs is not too large. For example Kremsmunster and Klosterneuburg tend to be a few nodes away from each other but never attached to the same internal node. This may suggest that for larger datasets, pylogenetic methods do not work as well as the NLP methods considered. This could be due to the fact that larger traditions are more likely to involve cases of contamination and have many multifurcations which cannot be predicted by phylogenetic methods. Surprisingly, UPGMA agreed perfectly with the philologist's prediction that the root was Wien 4936. It should be noted that these are all heuristics and not confirmed since this tradition was never studied in a stemmatology setting.

6 Conclusion

In this project we have developed a pipeline which aligns the words in a given corpus of text files, turns them into pseudo-DNA sequences and outputs NEXUS files which can be read by the phylogenetics PAUP* package for stemmatological analysis of any corpus of manuscripts. On top of this we built another pipeline which calculates the document dissimilarity matrix between the text files and predicts the stemma using Kruskal's algorithm to find the minimum spanning tree, expanding on the work done in [9], by including sentence paraphrasing in the generation of the artificial datasets and by using the reduced word mover's distance. All scripts and datsets were made available at https://github.com/DarZam/Masters-Thesis.

The minimum spanning tree calculated using the cosine distance and Text Frequency - Inverse Document Frequency with 1-,2-,3- grams or using word embeddings seems to yield good results for the synthetic datasets and the results are very easy to read. For Parzival, word vector methods also gave good results, even beating those from Text Frequency - Inverse Document Frequency albeit by a small margin, however the phylogenetic methods all managed to outperform the NLP methods with the exception of UPGMA and UPGMA with bootstrap (although Glove with the reduced word mover's distance achieved a higher average signed distance than maximum parsimony by a very small margin) 1. Moreover, the minimum cost heuristic [9] seems to yield consistently good results with all of the NLP methods considered. To the author's knowledge, this was the first time that the word mover's distance was used for stemmatology, demonstrating that natural language processing methods are indeed viable for this application. Having said that, the neighbour joining method with bootstrap yielded the best results for Parzival. It seems that TF-IDF with minimum spanning trees is a good starting hypothesis from which a skilled philologist would be able to reconstruct the true stemma.

Having said that, this is not a final solution for stemmatology, as there are several things which need to be considered depending on the corpus used. The methods used are incapable of incorporating information such as the date of publication, no assumptions on the directionality of the changes between the documents (such as newer words replacing older words) and the topology is restricted to trees.

6.1 Future Works

The main limitation of the artificial dataset experiments is that it is very time consuming to generate the datasets and thus it is difficult to perform these experiments on a mass scale. One idea would be to parallelise the process and generate the datasets using a computer cluster. This may also be useful when applying more time consuming methods such as the word mover's distance. Different tree/graph structures could be considered such as direct acyclic graphs to reconstruct contaminated texts or Steiner trees to predict where the location of the missing manuscripts should be [1].

Another direction would be to used different transformations to generate the datasets. The dataset generator used could incorporate new transformations quite easily. In particular it could be fruitful to conduct larger scale experiments with more varied text document lengths and larger numbers of nodes, missing nodes, missing texts and contamination. In our case, we only used text files of word counts of 500-750 due to computational constraints.

Using a wider range would allow a proper study of how the document length relates to the accuracy.

It could also be interesting to perform experiments conducted on datasets written by humans such as Parzival where the stemma is known for certain. Such open source datasets seem to be lacking.

One could also use other packages than PAUP* for phylogenetic inferencing. MrBayes [37] is a Bayesian phylogenetics program and is considered the standard model of choice across a wide range of phylogenetic and evolutionary models. MrBayes relies on variants of Markov chain Monte Carlo methods to estimate the posterior distribution of model parameters. One major limitation of MrBayes is that it is designed for Linux systems and that it can only read DNA and protein sequences rather than the pseudo-DNA sequences we generate in this project. The author of this script is aware of only a few papers which used Bayesian phylogenetic methods for stemmatology. In [38] BEAST 2 package [39] was used with collations encoded in TEI XML using the teiphy Python package to convert TEI XML to BEAST XML. Their results were consistent with established findings on the textual tradition they considered.

SplitsTree [36] is another popular program for computing unrooted phylogenetic networks from molecular sequence data. As an input it takes distance matrices or a set of trees and outputs a phylogenetic tree or network. Methods include split decomposition, neighbor-net, consensus networks, super networks methods or methods for computing hybridization or simple recombination networks. This makes it able to detect contamination however, the graph outputs are generally very difficult to interpret. Once again, this package was not able to read our pseudo-DNA sequences. A limitation of this package is that it can only read DNA and protein data and thus there is a much more limited number of character states that can be used than with PAUP*.

Another idea would be to investigate different methods for rooting the stemma. [40] suggests a method which relies on the directionality of changes. Correctly guessing all directions of the edges in a tree would naturally lead to the root. Some changes are less likely to occur than others. The addition of text is less likely to occur than transpositions and deletion of text and thus, from this one can guess the direction of textual evolution. In [40], it is shown that this method can be effective in predicting the root.

References

[1] P. Roelli. (2020). Handbook of stemmatology: History, methodology, digital approaches. Berlin, Germany: De Gruyter.

[2] C. Liu, C. Chen, J. Han, and P. S. Yu. (2006) "GPLAG: Detection of software plagiarism by program dependence graph analysis," Proc. KDD 2006, pp. 872–881.

[3] J. Sun, S. Papadimitriou, C.-Y. Lin, N. Cao, S. Liu, W. Qian. (2009). "MultiVis: Content-based social network exploration through multi-way visual analysis," Proc. SDM 2009, pp. 1063–1074

[4] S. Wehner. (2007). "Analyzing worms and network traffic using compression," J. Comp. Secur., vol. 15, pp. 303–320.

[5] M. Spencer and C. J. Howe. (2001). "Estimating distances between manuscripts based on copying errors," Literary and Linguistic Computing, vol. 16, p. 467–484.

[6] W. M. Fitch. (1971). "Toward defining the course of evolution: minimum change for a specific tree topology," Systematic Biology, vol. 20, p. 406–416.

[7] J. Felsenstein, (2004). Inferring Phylogenies. Sinauer Associates.

[8] P. Robinson, R. J. and O'Hara. (1992). Report on the textual criticism challenge 1991. Bryn Mawr Classical Review, 3(4): 331–7.

[9] G.D. Marmerola, M. A. Oikawa, Z. Dias, S. Goldenstein, A. Rocha. (2006). On the Reconstruction of Text Phylogeny Trees: Evaluation and Analysis of Textual Relationships. PLoS One. 2016 Dec 19;11(12):e0167822. doi: 10.1371/journal.pone.0167822. PMID: 27992446; PMCID: PMC5167249.

[10] B. Shen, C. W. Forstall, A. D. R. Rocha & W. J. Scheirer. (2018). "Practical Text Phylogeny for Real-World Settings," in IEEE Access, vol. 6, pp. 41002-41012, doi: 10.1109/ACCESS.2018.2856865.

[11] D. Jurafsky, & J. H. Martin. (2024). Speech and Language Processing (rd edition draft). Obained from https://web.stanford.edu/ jurafsky/slp3/.

[12] G. Salton & C. Buckley. (1988) Term-weighting approaches in automatic text retrieval. Information processing & management, 24(5):513–523.

[13] T. Mikolov, K. Chen, G. Corrado, & J. Dean. (2013). Efficient estimation of word representations in vector space. Proceedings of the International Conference on Learning Representations.

[14] P. Bojanowski, E. Grave, A. Joulin, & T. Mikolov. (2017). Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics, 5, 135-146.

[15] J. Pennington, R. Socher, & C. D. Manning. (2014). GloVe: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1532-1543. Data obtained from: https://nlp.stanford.edu/projects/glove/

[16] M. J. Kusner, Y. Sun, N. I. Kolkin, & K. Q. Weinberger. (2015). From word embeddings to document distances. In Proceedings of the 32nd International Conference on Machine Learning (pp. 957-966).

[17] F. J. Damerau. (March 1964), "A technique for computer detection and correction of spelling errors", Communications of the ACM, 7 (3): 171–176, doi:10.1145/363958.363994, S2CID 7713345

[18] N. Saitou, & M. Nei. (1987) "The neighbor-joining method: a new method for reconstructing phylogenetic trees": Molecular Biology and Evolution (Volume 4, Issue 4, Pages 406-425).

[19] R. H. Dekker, & G. Middell. (2011). Computer-Supported Collation with CollateX: Managing Textual Variance in an Environment with Varying Requirements. Supporting Digital Humanities 2011. University of Copenhagen, Denmark. 17-18 November 2011.

[20] D. L. Swofford. (2003). PAUP*. Phylogenetic Analysis Using Parsimony (*and Other Methods). Version 4. Sinauer Associates, Sunderland, Massachusetts.

[21] E. Talevich, B. M. Invergo, P. J. Cock, et al. (2012). Bio.Phylo: A unified toolkit for processing, analyzing and visualizing phylogenetic trees in Biopython. BMC Bioinformatics 13, 209. https://doi.org/10.1186/1471-2105-13-209

[22] A. A. Hagberg, D. A. Schult, & P. J. Swart. (2008). Exploring Network Structure, Dynamics, and Function using NetworkX, Proceedings of the 7th Python in Science conference (SciPy 2008), G. Varoquaux, T. Vaught, J. Millman (Eds.), pp. 11–15.

[23] R. Řehůřek, & P. Sojka. (2010). Software Framework for Topic Modelling with Large Corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks (pp. 45-50). Valletta, Malta: ELRA. Retrieved from http://is.muni.cz/publication/884893/en.

[24] https://www.cs.helsinki.fi/u/ttonteri/casc/data.html

[25] M. Lichman. (2013). UCI Machine Learning Repository. [Online]. Available: http://archive.ics.uci.edu/ml

[26] C. Fellbaum (Ed.). (1998). WordNet: An electronic lexical database. Cambridge, MA: MIT Press.

[27] R. Mitton. Corpora of misspellings for download; (1985). Available: http://www.dcs.bbk.ac.uk/ roger/ corpora.html

[28] D. Holbrook, (1964). English for the Rejected: Training Literacy in the Lower Streams of the Secondary School, Cambridge University Press.

[29] K. Atkinson: GNU Aspell spellchecker. Available: `http://aspell.net/`

[30] Wikipedia: The Free Encyclopedia. Common Misspellings. Available: `https://en.wikipedia.org/wiki/Wikipedia:Lists_of_common_misspellings/For_machines`

[31] S. Bird, E. Klein, E. Loper. (2009). Natural Language Processing with Python. O'Reilly.

[32] Y. Zhang, J. Baldridge, & L. He. (2019). PAWS: Paraphrase Adversaries from Word Scrambling. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) (pp. 1298-1308). Minneapolis, Minnesota: Association for Computational Linguistics. doi:10.18653/v1/N19-1131

[33] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, A. Joulin (2018). Advances in Pre-Training Distributed Word Representations. Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018). Data obtained from: https://fasttext.cc/docs/en/english-vectors.html

[34] T. Roos, T. Heikkilä. (2009). Evaluating methods for computer-assisted stemmatology using artificial benchmark data sets, Literary and Linguistic Computing, Volume 24, Issue 4, December 2009, Pages 417–433, https://doi.org/10.1093/llc/fqp002.

[35] Z. Dias, A. Rocha, S. Goldenstein. (2012). Image Phylogeny by Minimal Spanning Trees. IEEE Transactions on Information Forensics and Security. 2012; 7(2):774±788. doi: 10.1109/TIFS.2011.2169959

[36] D.H. Huson, & D. Bryant. (2006). Application of Phylogenetic Networks in Evolutionary Studies. Molecular Biology and Evolution, 23(2), 254-267.

[37] F. Ronquist, and J. P. Huelsenbeck. (2003). MRBAYES 3: Bayesian phylogenetic inference under mixed models. Bioinformatics 19:1572-1574.

[38] J. McCollum & R. Turnbull. (2024) Using Bayesian phylogenetics to infer manuscript transmission history, Digital Scholarship in the Humanities, Volume 39, Issue 1, April 2024, Pages 258–279, https://doi.org/10.1093/llc/fqad089

[39] R. Bouckaert, J. Heled, D. Kühnert, T. Vaughan, C. H. Wu, D. Xie, et al. (2014). BEAST 2: A Software Platform for Bayesian Evolutionary Analysis. PLoS Computational Biology, 10(4): e1003537. doi:10.1371/journal.pcbi.1003537.

[40] A. Hoenen. (2019). "Rooting through Direction - New and Old Approaches". In DHd 2019: Digital Humanities; Multimedial & Multimodal: Konferenzabstracts, edited by Patrick Sahle.