Simulating Mistakes

Using Agent Based Models to simulate and study the effects of scribal errors in classical text transmission.

Mees Gelein
T. Verhoef (supervisor)
T. A. Van Berkel (supervisor)
Master Thesis for Media Technology MSc. program
Leiden University, The Netherlands
www.mediatechnology.leiden.edu
February 2021

Abstract

The transmission of text by scribes introduces errors that often compound over time. Using the first book of Homer's Iliad as a case study, we would like to propose a method to simulate this process using agent based models. In this paper we will first briefly explain general information about text transmission and the nature of the text we are working with. Then we will discuss the creation of an agent based model (ABM) in detail, as well as the problems encountered during development. We will continue by evaluating the results of the model and studying the influence of its parameters. To end this thesis we will then contrast this method with other Natural Language Processing approaches such as Machine Learning and Evolutionary Algorithms. The results seem to indicate that while ABMs could be useful in a text transmission context, the surrounding research needs further development to improve the quality of the model. On the whole, the usage of ABMs in text transmission is deemed a fruitful approach worthy of further investigation.

Table of contents

1. Introduction: the Homeric Question and *hubris*

We, as scientists, can be proud, stubborn, and sometimes even foolhardy. It seems that every scientific field contains a seemingly minor detail that divides its peers into tribes who stand on either side of an issue. Whilst most of these problems are not really suited for actual scientific research but are a better fit for the dinner table or friendly discussion, some of them do spark interesting research.[1]

One such matter is the so-called *Homeric Question*, a subject which has been discussed among classical scholars for over 200 years (Fowler, 2004). Did the legendary Greek poet Homer actually exist? Or, more accurately: are the works we attribute to Homer the product of one author working in a similar way as modern day authors, or are they a product of centuries of oral composition and transmission? This is the question many classicists have fought over all this time. Over the millenia his works have survived, many versions have been known to exist. Some versions are almost completely with us to this day, some have only fragments remaining, while others are lost. According to Fowler (2004), these different variations caused scientists to start to analyze which verses and versions were really written by Homer himself and which by creative scribes.

To oversimplify centuries of research dramatically:[2] this practice of reconstructing the *true* homeric version continued until Milman Parry started publishing his works in which he stated that the different versions were products of an oral tradition. He hypothesised that the use of epithets[3] was a tool to make the verses easier to remember for poets who were orally composing and performing at the same time (Parry, 1930). As can be imagined: the existing scholarly tradition was not overly fond of this theory, since it basically invalidated most of their previous research. A divide in the classical field was born, and remains to this day.

Before we continue, let me introduce another relevant Greek term: *hubris*. Roughly translated it was generally believed to mean 'overconfidence', although modern insights provide a more nuanced view (Fisher, 2015). It is a thing many Greek heroes suffered from, and it is one of the driving factors that lead us to do this particular research. When centuries of the greatest classical minds cannot solve the *Homeric*

[1] An example of the more frivolous variety would be a hotly debated topic among computer scientists: Do you use tabs or spaces to indent your code.
[2] This really is a *dramatic* oversimplification and brushes aside centuries of scholarly research. We decided to rephrase this discussion in this manner since the history of the question was not a central part of this thesis. For a more extensive overview we can recommend the works of Fowler (2004), Nagy (2009) or West (2011).
[3] Short describing passages fitting a certain character. F.e. the *fast footed Achilles*, or the *rosy-fingered Dawn*.

Question, it is of course not *hubris* to think we can solve it with modern computational means.[4] Surely, in this thesis the *Homeric Question* will be answered once and for all.[5]

Maybe, however, we have to come up with some more reasonable goals and expectations. To simulate the errors of a scribe who is transcribing a work is no small feat, let alone to reverse them. In this research we wil try to build an agent based model that can simulate the behaviour of scribes as they copy the text. This model will allow us to study the effect of compounding error as the errors feed forward into new versions, and may provide us with some insights into the effect of individual scribes on a text tradition. Even more ambitiously, this model should be able to give us a hint as to what could have come before any of the versions we currently know of.

There are, of course, already many researchers who have looked into answering this question. In fact, reconstructing the so-called archetype, or at least providing insight into what it could have looked like, is one of the main goals of modern scholarly editions (West, 1973) (Wilson & Reynolds, 1991). There are also more modern approaches to publishing these editions, where digital means are used to provide more insight into the complicated history of a text (De Weerdt, Van Beijeren, and Gelein, 2019), (Alma & Berti, 2013).[6] This is a massive scientific field, with a rich tradition. Without a doubt we can use some of the expertise this field has collected over the centuries, which we will further discuss in the penultimate chapter.

While the construction of an agent based model for text transmission purposes seems to be a novel approach, the usage of agent based models in natural language context certainly is not. It is generally accepted that agent based models are useful in simulating human systems (Bonabeau, 2002), with research showing its strength in computational linguistics as well (De Boer, 2006). A doctoral thesis by Kaše (2019) even simulates transmission of religious ideas, at which point it looks like texts should be only a minor addition. The broad application of agent based models was one of the deciding factors in choosing it as the method with which we model our transmission.

There are many uses for such a text transmission model that we can think of. Using its reversing capabilities, editors of scholarly editions could use the models suggestions as inspiration for possible corruptions that have taken place. While the model itself doesn't prove anything in this case, it can provide an editor with interesting insight,

[4] Spoiler: it is.
[5] Spoiler: it won't.
[6] There are more projects, but these are just some that we have personal experience with. This is a rapidly evolving field with promising projects, such as the Sources Echo & Interactions of Protagoras project, in which we are also involved as part of the team: http://manmeasure.org/.

who then can turn this computer generated hypothesis into a scholarly researched theory. Maybe some brilliant scientist can actually prove the existence of a single manuscript of the Iliad by running this model on lots of different versions of the Iliad, only to find they converge to one root.

On the other hand, a model such as this could be used in its normal feed-forward mode to find probable spots for corruption. Given some set of provided parameters for the scribes who will transcribe it, what is the most likely location that errors will be found. This could even be used as extra proof to show the lineage of certain manuscript families.[7] With some minor alterations to the parameters it is even possible to simulate oral storytelling. While the nature of the errors would change between the two, the setup of the model would remain largely the same. Modelling scribes can also give us some insight into the probable nature of the transmission that leads to our surviving manuscripts. By connecting observations such as a diverse tradition with scribal tendencies we could document with this model, we can form hypotheses about the nature of the transmission, backed up by our simulation.

The possibilities for such a computational model seem very interesting. However before we dive into the specifics of the model we have created, we will first have to understand a bit more about the qualities of classical text transmission. We will talk about this in chapter 2 after which we will discuss the specifics of our chosen case study, the first book of the Iliad, in chapter 3. In chapter 4 we will discuss agent based models in general and our specific implementation will be described in more detail in chapter 5. We will then continue to evaluate our results and consider future work on this project in chapters 6 and 7 respectively, before we conclude with chapter 8.

All code for this project can be found on GitHub, but generally looking at this repository won't be necessary.[8] When deemed appropriate the code required to understand a passage has been included in this paper as a code block, such as the one shown below.

```
print(“Let’s make some agent based models!”)
```

[7] More about manuscripts, lineages and the *stemma* in chapter 2.
[8] https://github.com/MGelein/simulating-mistakes

2. Transmission of text

Storytelling is seen as one of the ways in which we form and maintain our social bonds, even dating as far back as our earliest ancestors (Dunbar, Gamble & Gowlett, 2014). To this day stories are one of the most important ways we make sense of the world around us, using the people who surround us and their views (Bietti, Tilston & Bangerter, 2019). However, together with the importance of storytelling through the ages comes the plethora of different ways stories have been told.

When talking about the transmission of a story, we thus first have to discuss the method with which the story is being transmitted. Roughly speaking we can make two distinct categories: stories can be transmitted orally or in writing. Until the advent of writing oral transmission would have been the only logical, or even possible, way of transmitting a story. As one can imagine this leads to small alterations every time the story is recited but it also impacts the way the narrative is presented. An example of this would be the use of formulaic phrases or epithets that Parry (1930) observed in Homer's works.[9] With written transmission we suddenly are no longer bound by our own memory.[10] The differences between versions are naturally much smaller, but as humans we still make mistakes (West, 1973).

It seems that we have our first important decision to make: what part of transmission should we simulate? The difference in nature between these two methods of sharing a narrative is big enough that they seem to warrant separate models. This initial thought, paired with the practical consideration that it is much easier to find and model written data, led us to the conclusion that we should focus on textual transmission.[11]

In the next paragraphs we will briefly discuss the basics of text transmission, with a minor focus on classical texts, since that is what we will be working on. This is largely based on the work of West (1973), with some minor additions based on Mathijsen (1995 & 2010) and Wilson & Reynolds (1991). According to Mathijsen we can divide text transmission into four periods. Of these periods we are mostly interested in the first: the scribal period in which handwritten copies were the only way of

[9] Interestingly, Goody (2006) argues that large narratively complex epics must necessarily be a product of an early *literate* society, since in an oral society nobody would have the time required to listen or perform such an enormous work of fiction.

[10] Or at least less so, as we will see in the following paragraphs when we start to discuss the way the agents in our model will use memory.

[11] By using digitised editions we are also not properly considering the visual aspect of a written manuscript, such as letters that are likely to be confused. These characteristics are likely language or even manuscript dependent. Modelling this highly unique behaviour, we decided, was not a priority for this thesis.

disseminating a story into the world. The advent of the printing press, which denoted the start of Mathijsen's second period, gave the option of creating exact copies, something that scribes were never capable of.

The process of copying a manuscript varied a lot over different periods and between different scribes. The general method however, remains largely the same. A scribe has access to a source manuscript and copies it, word by word, into a new one. In doing so he would inevitably introduce mistakes and sometimes well-meant alterations, thus spawning a new version of the manuscript.[12] [13] Although West (1973) mentions various ways of preventing these errors, such as direct comparison with the source manuscript after the copying was finished, this was still not enough to stop errors and alterations from creeping in.[14] These different versions can all spawn new versions, leading to something that resembles a family tree structure.

This family tree that so neatly, and visually, describes the relations between manuscripts is called a stemma. In textual criticism this stemma is created by careful analysis of all available manuscripts, and it is used to reason about the relations these manuscripts would have to the archetype, the root of the stemma (Wilson & Reynolds, 1991).[15] Below this paragraph, in figure 1, we can see an example of such a stemma that describes an imaginary relationship between nonexistent manuscripts. In this example the root of the stemma is manuscript A. This manuscript was copied three times, leading to manuscript B, C and D. However, as you can see, when copying manuscript D the scribe also had manuscript C there to double check. The resemblance of a stemma to a family tree, or a social structure, is what led us to first consider these manuscripts as products created by social agents, whose relationships could be digitally modeled.

[12] Going forward in this thesis we will use the word 'error' for both honest mistakes and conscious changes. An error is merely a deviation from the original text, and no value judgement is implied.
[13] Wilson and Reynolds (1991) have written extensively about the varying reasons these alterations or errors get introduced. The nature of the errors, as well as the reason for making them vary greatly depending on the time period and location the manuscript is created in, with great differences existing, for example, between the East and West Roman empire.
[14] West (1973) even mentions that this correctional use of the source manuscript would sometimes even lead to more errors. F.e.: If the original words were included next to an error in the manuscript, sometimes the next scribe would think both these variants were equally important and think of a middle ground to include them both partially.
[15] Wilson and Reynolds also argue that this reasoning back to the archetype is one of the limitations of this way of working. Some traditions are 'open' and contain multiple archetypes.

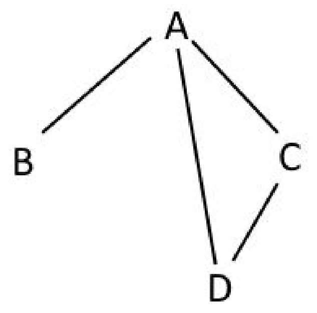

Figure 1. A graphical visualisation of an imaginary stemma.

Now that we have established how these versions of manuscripts come into existence, we need to look at the nature of the errors and alterations that get included. While some sources exist that mention various scribal tendencies for alterations, such as Royse (2013), Wilson & Reynolds (1991), and West (1973), none of them seem to be backed up with any kind of statistics or even something more than circumstantial evidence. We mostly know *that* things happen, we don't know how often or how large the chance of that error happening is. There simply seems to be not enough data currently available for such extrapolations.[16]

However, when creating a model of scribal transmission, we need to know what kinds of changes scribes might make. While a lot of this knowledge is widely spread among classical scholars and editors, much of it remains implicit, and hardly any of it is digitally available. The search for more tangible and condensed data led us to Dr. G. Boter, former professor of UvA Amsterdam. In 2013 he gave a guest lecture at Leiden University about scribal text transmission errors. During this lecture he gave participants a handout which contains a nicely organised list of the most common errors.[17] In a brief interview we conducted with Dr. Boter it was concluded that this list was one of the more complete summaries of scribal errors.[18] The usage of this document as a basis for this type of model would be sufficient for the abstract nature of our model.

[16] By studying modern scholarly editions of many works maybe such a list could be compiled however. These editions already contain remarks about possible alterations and the reason for their existence in their critical apparatus.
[17] This list is included, with permission, as Appendix A. This list is only an excerpt, the first two pages, of the complete handout. Unfortunately the contents of this handout are in Dutch. While the subject matter is Ancient Greek, most of the errors described in this document are applicable to other languages as well.
[18] This interview was conducted remotely online on the 4th of December 2020 due to the coronavirus pandemic.

The types of errors mentioned in the list, as well as those talked about in our interview, can be briefly summarized into the following categories:

- *Letter errors:* With these errors letters, or parts of words are changed. Depending on the language skill of the scribe this could lead to non-existent words. This is generally the biggest category of mistakes. These include things such as misread letters, and swapping of characters in a word.
- *Word errors:* Scribes would sometimes try to improve the text by adding in words they deemed more fit to the original manuscript. Alternatively they would replace words in dialects they did not recognize with the words in a 'correct' dialect (Wilson & Reynolds, 1991). One example of this is the replacement of synonyms.
- *Sentence errors:* There are cases where entire lines are added or removed because the scribe thinks notes are part of the main text, or the other way around: when a scribe deems a line of text unfit for the author.

Now, armed with this list of possible errors, we should take a quick look at the text we are going to be subjecting to our model; the first book of the Iliad by Homer.

3. Our subject: Homer’s Iliad, Book 1

The Iliad is the oldest surviving Western text (Mueller, 2013). It is an epic poem that tells the story of the wrath of Achilles, ultimately leading to his doom, set in the Trojan war. This story is told in 24 books, originally papyrus rolls, for a total of well over 10.000 lines. There are several reasons which led us to pick the first book of the Iliad as our case study.

The most obvious reason is that the text had to be part of a scribal tradition. This would make the already complex matter of evaluation of results a little easier.[19] Secondly, since due to a previous bachelor in classics we are familiar with the material as well as with the methods of obtaining the material. Most, if not all, examples of text transmission we know are from classics.

Thirdly, there was the matter of choosing between Latin or Greek texts, and after that between poetry or prose. The ideal text to work with would be a Latin prose text. Prose, since automated meter[20] detection for classical languages is another problem that at the time of writing has not been solved. If we wanted our model to stick to the meter as well, that would pose a significant increase in difficulty. Latin, because the Greek alphabet, along with its diacritics, is significantly harder and more memory intensive to digitally work with when compared to the Latin alphabet.

All of these choices and preferences then turned out to be useless when we realised that the only easily available, digitised, manuscript editions of a classical work known to us were the editions made by the Homer Multitext Project.[21] While it was an Ancient Greek text instead of Latin, and poetry instead of prose, we had little choice in the matter. Therefore this edition will be the source of our data after some data preparation described in chapter 5.

[19] More about evaluation in chapter 6.

[20] The rhythm contained within poetry lines. There is some research into automated detection of this rhythm (Abuata & Al-Omari, 2018), but none of this research is about classical meter.

[21] *“The Homer Multitext project seeks to present the Homeric Iliad and Odyssey in a critical framework that accounts for the fact that these poems were composed orally over the course of hundreds, if not thousands of years by countless singers who composed in performance. The evolution and the resulting multiformity of the textual tradition, reflected in the many surviving texts of Homer, must be understood in its many different historical contexts. Using technology that takes advantage of the best available practices and open source standards that have been developed for digital publications in a variety of fields, the Homer Multitext offers free access to a library of texts and images and tools to allow readers to discover and engage with the Homeric tradition.”* from http://www.homermultitext.org/about/ on January 9, 2021.

We are using the first book of the Venetus A manuscript as contained in their *hmt-2020i* release (Homermultitext, 2020).[22] This book is 611 lines long and contains the famous first words of Western literature.[23] The Venetus A manuscript was chosen since it is one of the best preserved, as well as one of the oldest manuscripts of the Iliad, dating back to the 10th century (Dué & Ebbott, 2014).

[22] While any other book, or even multiple books, could have been used, we decided to stick with just one, familiar, book to help the evaluation step.
[23] "Μῆνιν ἄειδε θεὰ Πηληϊάδεω Ἀχιλῆος", roughly translated that means: "*Sing, Goddess, of the wrath of Achilles, son of Peleus.*"

4. Agent Based Models

The nature of the data that runs the simulation necessarily prescribes the kinds of algorithms that are usable to model a system. As we have established in chapter 2, a stemma for a manuscript closely resembles a family structure, or a social structure. These observations provide use with two obvious candidates to simulate our system: agent based modelling or genetic algorithms. The difference, at least for our purposes, is how they handle the data in the system. A genetic algorithm operates by selecting for the most 'fit' candidate (Mitchell, 1998). Thus it operates directly on the object to be studied. By abstracting away the individual scribe's brain through which a text is read and reproduced we might lose a critical step in the transmission path. In case of our manuscript tree this would mean we would be looking at the most fitting text instead of simulating individual scribe. Such an evolutionary model does not seem to fit reality, since the reasons for manuscripts to survive are more often tied to chance than to their quality.[24][25] Moreover, according to de Boer (2006) genetic algorithms tend to converge towards a single optimal solution, while we want to simulate divergence with more optional convergence.

An agent based model on the other hand creates a population that operates on the data. This resembles the nature of transmission more closely. The individual scribes acting upon the manuscript closely resembles that of individual agents acting upon the text. In fact our system of transmission seems to easily fit some of the criteria that Bonabeau (2002) describes as good indicators to use Agent Based Modelling (ABM), such as the fact that we have a clear network effect, as well as the strong influence of individual agents, which Bonabeau describes as "the system is linearly stable but unstable to large perturbations". Thus we decided to go with agent based modelling to create our simulation.[26]

Another option that we briefly entertained was the use of machine learning, more specifically neural networks, to predict the corruptions and errors of scribes. There is even a recent project that has had major succes with restoring lost passages of ancient Greek inscriptions (Assael, Sommerschield & Prag, 2019). Unlike this project however,

[24] One only needs to look at the fire in the Great Library of Alexandria, 48 B.C. to see that even the greatest sources of information can be lost.

[25] According to Wilson and Reynolds (1991) censorship plays a remarkably small role, if any at all, in the survival of the classical texts. Censorship was generally reserved for contemporary pagan or sacrilegious writers, while the classical texts remained largely untouched. For a more complete overview, we recommend chapter 2 of their book: The Greek East.

[26] An added benefit of going with agent based models is their broad range of applications. As discussed in the introduction, these models have been used for a variety of linguistic or theological research, but even simulation of Covid19 transmission is something which an agent based model can tackle (Cuevas, 2020) (Gomez, 2020).

we do not have access to the wealth of digitized fragments that the papyrological community has built over the decades. When paired with the ‘black box’ nature of neural networks this turned out to be undoable. We decided to stick with our initial pick of agent based models. In chapter 7 we will, however, briefly discuss the possibilities a machine learning approach could provide, and the problems that need to be solved before such an approach would become feasible.

According to Bonabeau (2002), the idea of an agent based model is to simulate a population of autonomous *agents*. Each of these agents has individual properties that determine how they interact with their environment. This makes them a natural fit to many real-life problems and social structures (Helbinger, 2012). By interacting with their fellow agents they influence the shared world they live in. This shared world can, of course, also have some properties that have an effect on agent interaction. The most obvious property being spatiality; agents being separated by space to simulate geographical separation.

The tunable parameters are in the end what determine the outcome of the model. Therefore some careful thought needs to be put into what we should simulate. Our perfect model is completely accurate to an individual scribe, but since we actually want to make this model we will necessarily have to make some abstractions.[27] For our model, in which every agent is a scribe, we have decided on the following parameters:

Agent memory: a number that will determine how well an agent remembers things, such as the words or letters they just read. Lower values mean that there is a bigger change of misremembering letters or even entire words. This is a metaparameter. It controls the chance of all mutations happening.

Agent vocabulary: a number that will control how likely an agent is to recognize a word as part of their vocabulary. A recognized word then gets added to an agent’s list of known words.[28] Higher numbers mean there is less of a chance of correct words being rejected and incorrect words being included in a text.

Agent arrogance: this parameter tunes the likeliness of an agent actually acting upon things they perceive as incorrect. An agent with a low vocabulary and low arrogance actually makes a great scribe, although they won’t understand what they are copying, whereas high arrogance, in extreme cases, can lead to entire word substitutions.

[27] These abstractions are also paramount in making this model easier to draw conclusions from. The complexity of the real world is often the thing that prevents us from drawing conclusions, whereas abstraction lets us study the underlying systems.
[28] This is not a tunable parameter and therefore not included in this overview.

Agent position: the agent's position is determined by a one dimensional coordinate which denotes their position on the plane of the simulation. By spreading agents around we can limit interactions between agents. This allows us to simulate the limited availability of manuscripts which West (1973) describes.[29] If we give all agents identical positions we effectively remove the influence of spatial separation from the simulation.

Agent influence: this parameter can be used in conjunction with the spatial separation model to model the importance of certain scribes or manuscripts. Certain works could have a larger impact than others, which we can simulate by having an increased influence. The influence number will determine the fraction of the simulation space in which other agents will consult this manuscript.

Now that all of our requirements have been discussed it is time to continue onto the meat of the model: the implementation, which we will discuss in the next chapter.

[29] During early prototypes of the model agent influence and position was abstracted away as 'availability'. This method was discarded in favor of the more easily tunable 'position' and 'influence' parameters.

5. Implementation Details

One of the first decisions to make when implementing any kind of computer software is the technology to use. Over the course of the past few months we have developed several prototypes. We did this to gain familiarity with agent based models and learn about the various ways we could structure this project. During this period we have tried various languages and even frameworks for agent based modelling,[30] but ultimately we decided to stick with Python 3 with some external libraries.[31] The ease of use of Python development far outweighed any performance concerns such an interpreted language might cause.

The next thing we need is, of course, data. In chapter 3 we discussed our reasons for choosing the first book of the Iliad, specifically the digital database that contains a transcribed version the Venetus A (Homer Multitext, 2020). All the data in this database is in the form of the CITE architecture that was developed specifically for this project (Blackwell & Smith, 2019). Luckily the `.cex` format that was used for our specific database is a plain-text format, which meant we could easily extract the required data from the file. This straightforward process is documented in the `data_extraction` folder of the source code repository.[32] Our data was now one file per book of the Iliad in the Venetus A version, with each line in the file corresponding to a line of poetry in the original.

Since we would be needing quite a lot of tunable parameters we decided to group all of these together in a settings file. This file is the only command line parameter that the program accepts. For a brief moment we considered also allowing command line parameters as a way of controlling the simulation, but because of the amount of parameters this quickly got unmanageable. The `settings.ini` file that comes with the repository contains some sane defaults.[33] To run the application with the default settings file you run the following command in the terminal:

```
python main.py --settings settings.ini
```

[30] Frameworks such as Mesa proved to be either too complex for our use case, or not flexible enough. Furthermore, the basics of an agent based model were easy to implement ourselves.
[31] These libraries are *numpy* and *gensim*, both of which can be downloaded from the Python Package Index. The specific versions used can be found in the source code repository.
[32] To briefly summarize the process: each line was prefaced by a so-called CTS identifier for this specific line. By filtering only the identifiers that we needed we then saved only the Greek parts of these lines to separate files.
[33] A settings file is expected to be conforming to the standard of INI files, which Tyers (2016) explains quite well. The INI file is a very loose standard but it is basically just a key-value pair file.

By using this file, or creating your own using the default file as a guideline, we can easily manipulate the conditions of the simulation, or execute multiple different runs in parallel. All settings should have a pretty self-explanatory effect, especially for those who have read chapter 4. However, the default settings file contains comments that describe the parameters in more detail to make the process of modifying these settings easier.

As the text, in our case the first book of the Iliad, is loaded into memory, it is split by line, with each line being split by word. In some early prototypes we would just give the agents access to the complete book as a single string of text, but this soon turned out to be a performance bottleneck since strings in python are immutable (Van Rossum, 2007). This means that every time we make an alteration to the original string, we effectively create an entirely new copy of the original with a small alteration. Using larger texts this caused considerable slowdowns and unnecessary overhead. By splitting the text into a list of lines, and each line into a list of words we can easily manipulate only specific words, keeping the overhead to a minimum. In the figure below you can see an example of what the first two lines of the Iliad look like in this data format.

```
[[“Μῆνιν", "ἄειδε", "θεὰ", "Πηληϊάδεω", "Ἀχιλῆος"], ["οὐλομένην·", "ἡ",
"μυρί", "Ἀχαιοῖς", "ἄλγε", "ἔθηκεν·"]]
```

Now that we have our parameters set and the texts loaded into memory, our next hurdle is to set up the actual simulation. The abstract notion of an agent based model is rather easy to implement. We have a `Simulation` class that controls a population, a list, of `Agent` instances. The size of the population and the runtime of this simulation are both loaded from the settings we have provided in the beginning. At this point we let the agents interact with each other for the given amount of time, until we end the simulation. At this point we save the text, that every agent holds in memory, to an output directory and we are ready for some analysis of the results.

There is one large piece of the puzzle that is missing: the agent’s behaviour. The key of this whole simulation is that they are able to read, process and write versions of texts. This process introduces errors inspired by the ones in the list of Boter (2013). As we have discussed in chapter 2, there are three broad categories of errors: letter errors, word errors and sentence errors. These categories are shown here in order of importance.

By far most of the errors that Boter mentions are at the letter level.[34] For each of these levels we will need different methods of introducing them to resemble the errors that are mentioned in Appendix A. In the next few paragraphs we will talk through each of the possible mutations, starting at the letter permutations and ending at the more rare sentence changes.

One of the easiest errors to implement was the mutation of letters. By randomly varying one of the letters of a word we could simulate the agent misreading the word. There was one big issue with this: misreading a word usually has a reason. Depending on the language level of the scribe the misread word should probably be a word itself. There is of course a chance that the agent misreads the word but that they, despite not understanding the word, still put it in the text. So, to summarize, the agent has to mutate a letter in a word, and then compare the resulting word to its known vocabulary. If it is in the vocabulary, the agent can just copy it, if it isn't we should see if the agent is arrogant enough to keep it, incorrectly thinking this word exists. In code this would be:

```
def mutate_letter(self, word, alphabet):
    rnd_index = random.randint(0, len(word) - 1)
    letter = random.choice(alphabet)
    new_word = word[:rnd_index] + letter + word[rnd_index + 1:]
    if new_word in self.vocab:
        return new_word
    else:
        return new_word if random.random() < self.arrogance else word
```

We do a similar thing for swapping letters around in a word, since this is one of the other errors that Boter mentions in his list. Another alteration on the letter level is just finding a word that is very close to this word in appearance. We originally did this using Levenshtein distance, closely following the approach we had used in COMPARATIVUS, a project in which comparing and finding matching text was the main focus (De Weerdt, Gelein & Ho, 2017). Unfortunately this approach proved to be too computationally expensive to be usable on this scale, which is why we used a modified Hamming distance function.[35] We go over the entire vocabulary and score

[34] This makes intuitive sense. While you might just as easily skip a letter as you skip a sentence, the latter is much easier to notice for someone who is checking your work.
[35] The large amount of matrix math that was necessary for Levenshtein distance to work was computationally too expensive. Normal Hamming distance only works on equal length strings, but with a slight modification in which we inaccurately account for length difference we can now use its superior speed, since it needs none of the matrix math that made Levenshtein distance so slow.

the words on their edit distance from the original word, which leaves us with a list of the closest words. With all these functions a large part of letter level errors is now covered. Which leads us to the next level words, and where this mysterious `vocab` variable comes from.

While swapping out words can be just as easy as swapping out letters, a little more care should be taken with semantics. While randomly misreading a letter can happen quite easily, and arrogance plays a role in there as well, word level errors are modelled as having to do *only* with arrogance. Sometimes a scribe just feels as if they know best. As mentioned in chapter 2, this would often happen with differences in dialect (Wilson & Reynolds, 1991).[36] The problem with this is that they still have to take the meaning of the word into account. The solution to this problem is also the answer to the question where the agent's vocabulary comes from: word vectors, or word embeddings. Before we continue we should probably briefly discuss them.

Word embeddings are essentially *n-dimensional* vectors that represent the meaning of a word in the same *n-dimensional* space (Liu, Liu, Chua & Sun, 2015). These models are usually trained on large corpora of texts (Mikolov, et al., 2017). This allows us to statistically capture the context of a word and thus the meaning. One algorithm that does this is the so-called *word2vec* approach by Mikolov *et al.* (2017). This algorithm, as implemented by the *gensim* python package is what was used to generate the word embeddings that we will use for our agent's vocabulary. We trained this model on the texts in the Ancient Greek Dependency Treebank 2.1 (2021). This dataset contains a selection of all known ancient greek texts and has proven to generate good embeddings.[37] We created word embeddings with 100 dimensions, since that seems to be a common agreed upon size (Rong, 2014). These word embeddings are also the source of the vocabulary of the agents. Any word not in the embeddings can unfortunately never become part of the vocabulary of the agent.

Now that we have explained the mysterious `vocab` that every agent has we can explain how we do semantic substitution. Thankfully, this is really simple. The *gensim* data object that contains the word vector provides us with a function that gives the most similar *n* words. By taking a pick from the top ten words, we then get some word variety whilst theoretically keeping the semantics as close to identical. The function

[36] As Wilson and Reynolds (1991, p. 48) argue: "They had the effect of instilling the forms and inflections of the Attic dialect so deeply that, when an educated man was transcribing a text, he tended to replace forms drawn from other dialects by the Attic forms which he knew so well."
[37] We are referring to a project that hopefully will be published someday. Together with J. Offerijns we spent the spring of 2020 training a neural network to correctly classify Ancient Greek words.

that handles this substitution is rather short and shown below. The try-except clause was used to prevent crashes when trying to find similar words to words that were not in the original training set.

```
def substitute_word(self, word):
    if random.random() > self.arrogance and word in self.vocab:
        return word
    try:
        top_ten = self.embeddings.most_similar(positive=[word], topn=10)
        return random.choice(top_ten)
    except:
        return word
```

The final level on which scribal errors take place is the line level. Examples of this mostly include the omission of several lines. This can happen for various reasons. For one, the scribe might just not think they are part of the work, or their eye drifts to the same word in the next sentence. To simulate the first we will introduce a small chance, based on an agent's arrogance, to leave out a line. To model the second problem, when your eyes drift from one word to the same word in the next line, we need a more complex solution. For every word in the line we will check if it was also in the previous line. If it is, there is a small chance that all preceding words get cut out between the two occurrences. In code that looks like this:

```
def skip_line(self, sentences):
    sentence, next_sentence = sentences
    if random.random() < self.arrogance:
        return [next_sentence]

    for word in next_sentence:
        if word in sentence:
            start = sentence.index(word)
            end = next_sentence.index(word)
            return sentence[:start] + next_sentence[end:]

    return sentences
```

At this point we have discussed all possible ways of mutations, now we only need to consider how these mutations are applied to have a full understanding of how this model works. It boils down to the following steps: unless spatial separation is disabled, every agent will try to find another agent to read their text. With spatial separation enabled this is only done within the influence radius of an agent. As an agent reads a text the first errors get introduced, this simulates the process of misreading the source manuscript. After that they have a version of that text in their memory, this is the text as they think they have read it. Then to finish one iteration of the model, every agent writes their own canonical version of the text, ready for another agent to read. During this writing process another round of mistakes are introduced, but this time it can also be more serious mistakes such as deliberate word substitutions or line skips.

This entire process of the agent reading other agent's texts and writing their own versions based on it will continue for a specified amount of iterations. After all these iterations are finished the last written text of every agent is saved to the output directory for further analysis. While this process seems rather interactive, especially since we have focussed on its actions in this chapter, this is not necessarily the case. By tuning the parameters we can determine how historically accurate we want the model to be. The amount of interaction between the agents, as well as the number of agents would logically vary for every tradition. An evaluation of the results, and the effect that different parameter settings can have on the output of the model will be the main focus of our next chapter.

6. Evaluation

The time has finally come: we are ready to run our model and see what results it will bring us. In this chapter we will look at various parameter configurations and discuss how they influence the end result. We will systematically go through each of the parameters and discuss its effect on the resulting text. Some of the agent's properties can also be defined as a range, this is the random deviation an individual agent can have from the baseline prescribed in the settings file. An agent's individual value will be samples from a uniform distribution between the two values, including both endpoints.

To measure the changes in a text we could unfortunately not use TF-IDF, which is a standard to compare and classify texts (Ramos, 2003). To properly use the TF-IDF algorithm, which stands for term frequency inverse document frequency, a list of commonly used words is necessary. This improves the accuracy of the comparison vectors that are created (Ramos, 2003). Such a list was unfortunately not at our disposal for ancient greek. As our next option we wanted to compare the texts using edit Levenshtein distance, as we initially also wanted to compare words, but once again performance proved to be a problem.[38]

So, instead of this approach we decided to use only part of the TF-IDF algorithm, namely the term frequency part, and compare the resulting vectors using cosine similarity.[39] To calculate the term frequency vector of a given text we look at all the unique terms in the entire corpus and count how many times they are used. We do this for every term, until we end up with a long, unique, list of counts. This is the term frequency vector, which we then compare to other vectors using cosine similarity. We then can look at the means, minimums, and maximums of the similarity scores to draw some conclusions about the text that was generated. When we study the influence of a single parameter it is safe to assume that the other parameters are set to the recommended settings, which are the default parameters in the settings file.

In this graph, and all others to follow, maximum similarity signifies the highest similarity that was found between two documents. It encodes the chance of creating

[38] Comparing two texts would take upwards of several hours. With five agents we would need ten comparisons, costing us upward of 24 hours of compute time. This would then be necessary for *every* permutation of the settings, which could easily lead to up to a month of constant computing time.

[39] Cosine similarity is a common method of comparing the similarity of two n-dimensional vectors. It is often used in similar scenarios, sometimes even in conjunction with TF-IDF (Li & Han, 2013), (Tata & Patel, 2007), (Rahutomo, Kitasuka & Aritsugi, 2012)

‘tribes’[40] of manuscripts. Minimum similarity shows the biggest difference between two manuscripts, which, combined with the maximum similarity shows us the range of text similarity options. A big difference between maximum and minimum scores, combined with a high maximum score, signifies a very diverse but stable tradition whereas the same difference between maximum and minimum scores combined with a low maximum score is a sign of a divergent tradition with little internal cohesion.[41] The average similarity gives us more information with which to interpret the other lines. The closer the average is to the maximum, the more stable the tradition is, and inversely: when the average is close to the minimum the tradition is more diverse. To summarize: a stable tradition leads to a stemma with less branching, and an unstable tradition leads to a highly branched stemma.

Iterations (n), see figure 2: By studying the graph below, using the criteria discussed above, we can see that there is a clear connection between the amount of iterations and the similarity of texts. We ran seven different configurations, each only differing in the amount of iterations. Every iteration we introduce errors and the chance of correcting these errors with another error is small, which we can see by the slowly declining lines. We very quickly create a more divergent tradition, as can be witnessed by the low average similarity. Thus we decided to stick with 10 iterations for every simulation going forth. This proved to be a good balance between a clearly visible effect on the similarity scores and short runtime.[42]

[40] With tribes of manuscript we mean families that are very similar. In case of a high maximum similarity there are at least two manuscripts that are very similar, forming a family.
[41] A high maximum similarity with low minimum score can be thought of as two different branches in the stemma. Internally these branches are very similar, but the difference between these branches has increased as time goes on.
[42] In total we needed to run 101 different configurations, so the processing time required for one experiment was very important. While the effects of every parameter would have been more pronounced at, for example, 20 iterations, we decided that our default of 10 iterations was enough to draw conclusions.

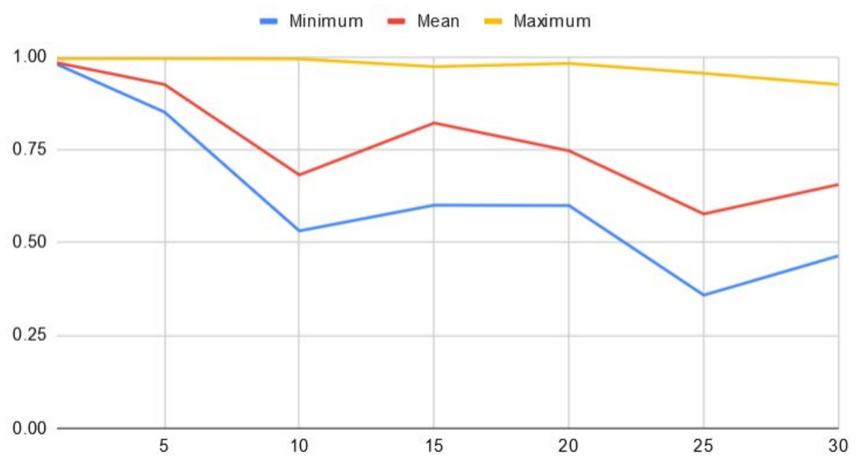

Figure 2. Iterations (x) plotted against document similarity (y).

Population size (n), see figure 3: We ran a total of nine different configurations, populations between size two and ten including both numbers, and our results are shown below. Interestingly a simulation with two agents proved to be very hit or miss, with some simulations becoming very similar, whereas others would be very different. This effect becomes less the greater we make the population, but once again we are limited by computational restrictions. Interestingly, after six agents we see a significant drop in the average and minimum similarity values, likely caused by too many agents inhabiting the same space. Since we kept the simulation space constant in these experiments, more agents would mean more overlap in influence, and thus more sources for every agent to choose from, which causes the tradition to diverge. We chose six agents as a compromise between stability and performance going forward.

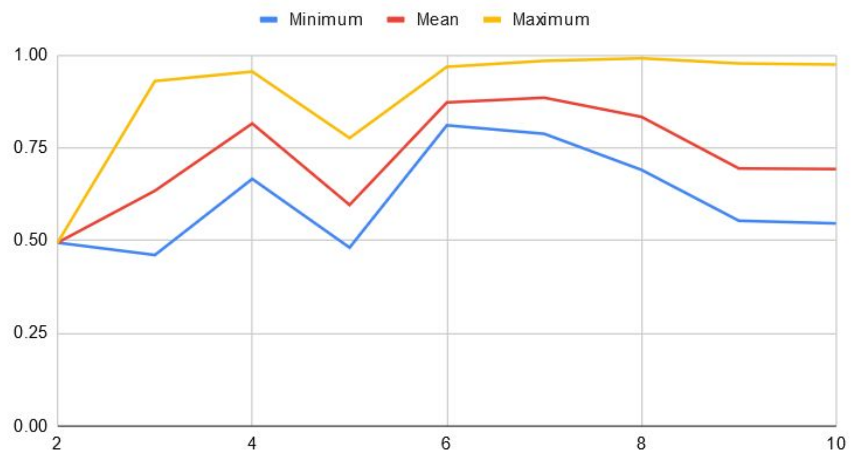

Figure 3. Population size (x) plotted against document similarity (y).

Simulation space (n): as discussed in our previous entry, the simulation space determines how spread out our agents are. This, together with agent influence and population size, determines the amount of interaction between agents. If the number of agents stays the same, we have seen that a larger simulation space will allow more versions of the text to stay alive, creating a more divergent tradition, whereas a smaller simulation space has a more converging effect, up until a point at which the influence of every agent overlaps with too many others. If we set this space to be infinitely small, zero units[43], we effectively disable spatial influence in our simulation. This parameter can basically be seen as another tuning parameter for the population size, therefore it was chosen to tune only one of these two parameters. For any given population size we could find the right simulation space, or vice versa, but given our computational restraints we needed a space for six agents, which made our default space 300 units in size.

Distribution randomness (n[0-1]), see figure 4: this parameter tunes how randomly the agents are distributed across the simulation space. When we set this setting to zero the agents are spaced with an even distance between each agent. Turning this all the way up to one will randomly scatter all agents in the simulation space. Numbers in between these two extremes are possible to make the distribution of agents lean more towards the orderly or chaotic side. Contrary to what we originally expected it turned out, after running 11 simulations with varying distribution randomness values, that we need a little bit of randomness for the most accurate results and a more convergent tradition. A distribution randomness value of 0.4 gave the best results, which we can explain as follows: we hypothesize that a small amount of random distribution provides a balance between the unpredictable chaos of full random distribution and the linearity of complete determinism. When we run a simulation fully deterministic every agent is spaced evenly in the simulation space, which leads to a situation in which no agent has a dominant position. By using a small amount of randomness in the distribution of agents we ensure that some agents have a position that reaches more other agents, thus increasing their influence over the final versions of others. For our default value we therefore chose a distribution randomness of 0.4.

[43] We use 'units' as a measurement for space. If agent A would be at position one and agent B at position four, they would be three units apart. Whether these units represent kilometers, meters or even lightyears is irrelevant for this simulation.

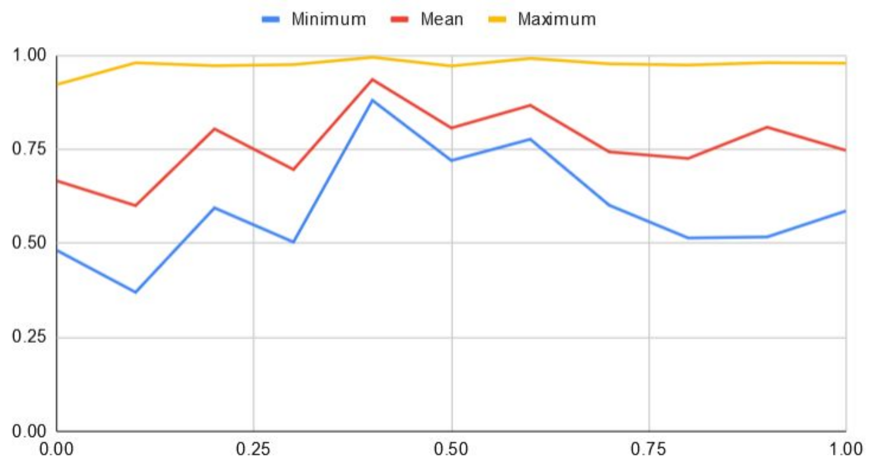

Figure 4. Distribution randomness (x) plotted against document similarity (y).

Agent memory (min [0-1], max [0-1]), see figure 5: Every agent in the simulation will sample its memory from the range prescribed in the settings file. If the minimum is larger than the maximum, randomness in this parameter is effectively turned off. This parameter controls the amount of mistakes an agent makes the most. For the text transmission to become stable we have two requirements to this setting: the minimum agent memory must be relatively high, otherwise we are constantly introducing new errors into the mix. The maximum agent memory must be as close to one as realistically possible. Setting an agent's memory to one will prevent it from ever making mistakes, which is not realistic, whereas a memory of zero will lead to so many mistakes that it becomes unrealistic yet again.

Before we can continue the results we go from our 24 simulations graphed below we need to discuss briefly how we chose to set up our experiments for parameters that are a range. We now not only need to analyse the effect of these two variables, but also the effect of the difference between them. Therefore we decided to create four sets of experiments, each with a larger range between the minimum and maximum value. The difference for this range was chosen depending on the variable. Then we increased our minimum range in six increments from the smallest to the largest possible value. This way we can study the effect of the difference between maximum and minimum values as well as these values separately.

When we look at the results of our simulations for agent memory there are some interesting observations we can make. First of all, there is a very clear correlation between low minimum memory and traditions with very little similarity. Low values for agent memory introduce too many mistakes, even when the maximum agent memory can be near one. As we suspected, a tight range, near the top of the memory spectrum behaves best, which is why we went for a range of 0.8 to 0.99.[44]

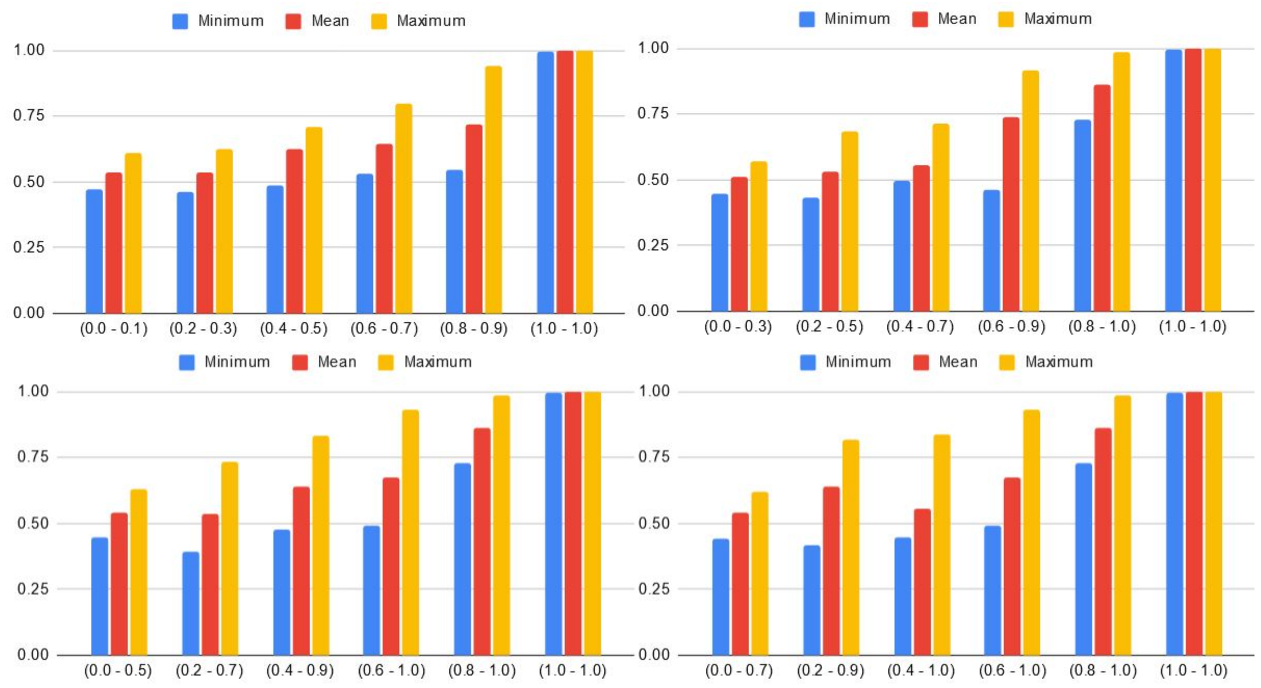

Figure 5: Agent memory with various distributions (x) versus document similarity (y). Top left: range of 0.1. Top right: range of 0.3. Bottom left: range of 0.5. Bottom right: range of 0.7

Agent vocabulary (min [0-1], max [0-1]), see figure 6: by controlling how much of the vocabulary the agent knows we can control how likely an arrogant agent is to make changes to letters or even complete words. In the four graphs, summarizing 24 experiments, we can see that it is hard to draw conclusions, since especially the graphs with larger ranges, the two bottom ones, are nearly identical. This may indicate that when even a few agents have knowledge of vocabulary we immediately get comparatively large changes to the text.

Interestingly, low vocabulary values do not necessarily mean that the resulting transmission will be corrupted. The text may be completely fine as long as the agent has high memory and low arrogance. Even if their memory would be low, their mutated letters would still be caught when they didn't recognize the mutated word. Thus an agent that has no understanding of the language can still make effective copies. Agents who have high vocabulary on the other hand are more likely to make changes to the text because of their deeper understanding. We are much more

[44] The latter, 0.99, because we like the idea of imperfect memory. We are only simulating *on* computers, we aren't simulating them. For the results this has no perceivable difference, making it more of a philosophical point.

dependent on their memory and arrogance values to make sure they don't make errors. These effects can most clearly be seen in the first graph of figure 6 where we can see a descending line in text similarity, and thus a more divergent tradition, with higher vocabulary values.

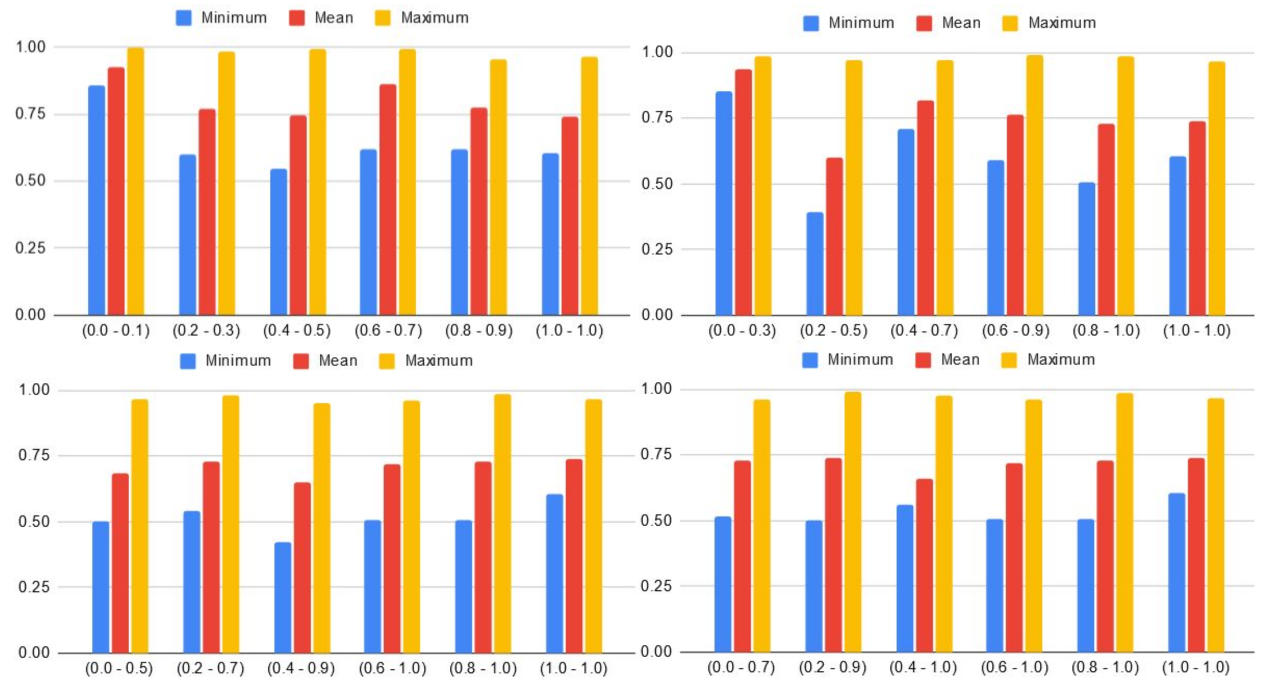

Figure 6: Agent vocabulary with various distributions (x) plotted against document similarity (y). Top left: range of 0.1. Top right: range of 0.3. Bottom left: range of 0.5. Bottom right: range of 0.7

Agent arrogance (min [0-1], max [0-1]), see figure 7: the arrogance value of an agent has a major impact on their behaviour. High values will cause agents to make many assumptions about, and corrections to, the text. The agent firmly 'believes' that they are correct. Low values on the other hand leads to agents that are less likely to accept their own chances. By rejecting their own mutations they effectively 'double-check' their own work. High average arrogance in the population has predictable effects, as can be seen in the graphs below. Many texts will spawn and there is little chance of converging to one version of a text, since each agent is constantly 'improving' the text.

As arrogance increases we can clearly see in the graphs below that the resulting texts diverge dramatically. With large ranges of arrogance we can see that even a small number of arrogant agents can wreak havoc on transparent transmission. Interestingly, we see that our simulation that used only arrogant agents has quite a decent score. While this may be due to chance, there is also a possibility that agents make similar edits or errors in their arrogance, which makes their text editions more similar by accident. It looks like this parameter has the second most control over how many errors a scribe makes, only after its memory.

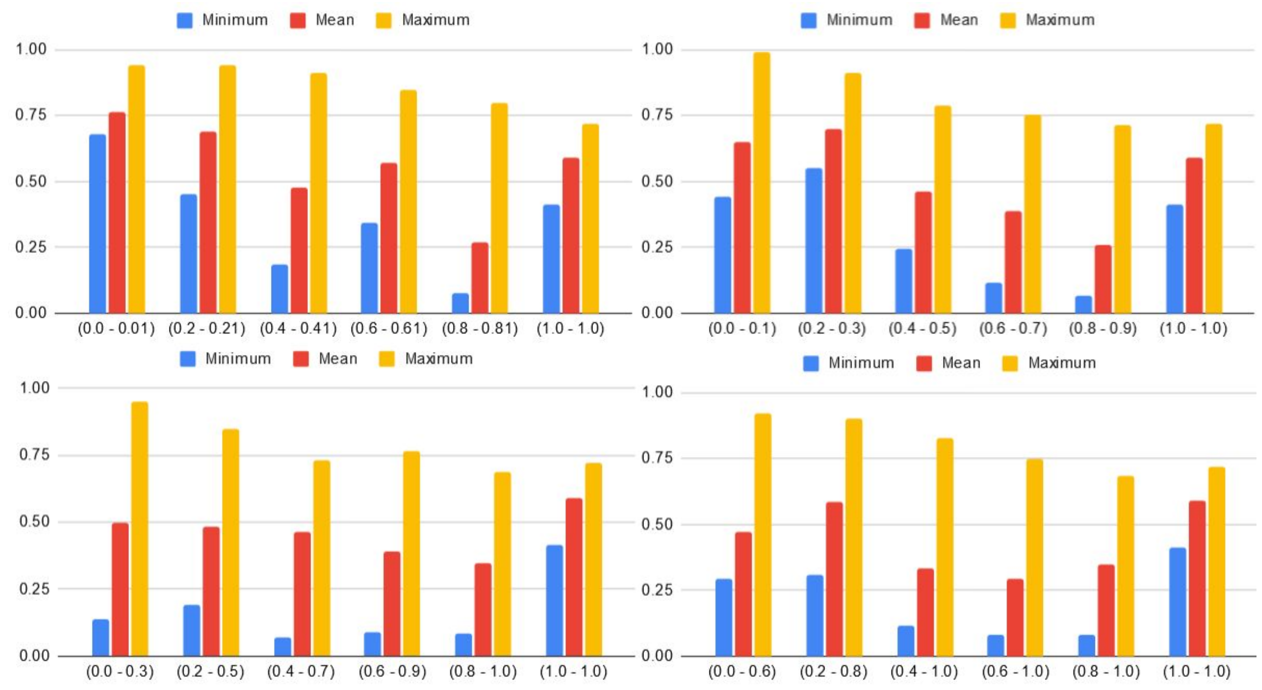

Figure 7: Agent arrogance (x) plotted against document similarity (y). Top left: range 0.01. Top right: range 0.1. Bottom left: range 0.3. Bottom right: range 0.6.

Agent influence (min [0-1], max [0-1]), see figure 8: This range determines the radius as a ratio of the simulation space in which an agent is able to be read by other agents. When this influence is set to one it covers the entire space, while setting it to zero makes the agent invisible to others. Thus high influence improves the chance of other agents picking this agent to read as their source, since our influence covers more agents, whereas low influence all but disqualifies an agent from being read by others. This parameter only works when the simulation space is large enough to fit agents' influence.[45]

From our experiments, graphed in figure 8, it looks like the best settings for convergence require a healthy range of influence, with some agents having little influence, while others have lots of influence. The less influential agents seem to work as some sort of buffer for the agent with the largest influence, making the text far more resilient to changes. While high influence levels are necessary to make sure at least one other agent can read your text, we can also see that the experiments with a narrow range still had trouble converging to a single tradition. The most stable tradition was the one with the largest range of influence. This might hint at the

[45] In extremely small simulation spaces an agent's influence covers the entire simulation space, effectively making its position irrelevant.

existence of a few influential scribes who dictate most of what we know today about for example Homer, with many other editions getting lost in the sands of time.

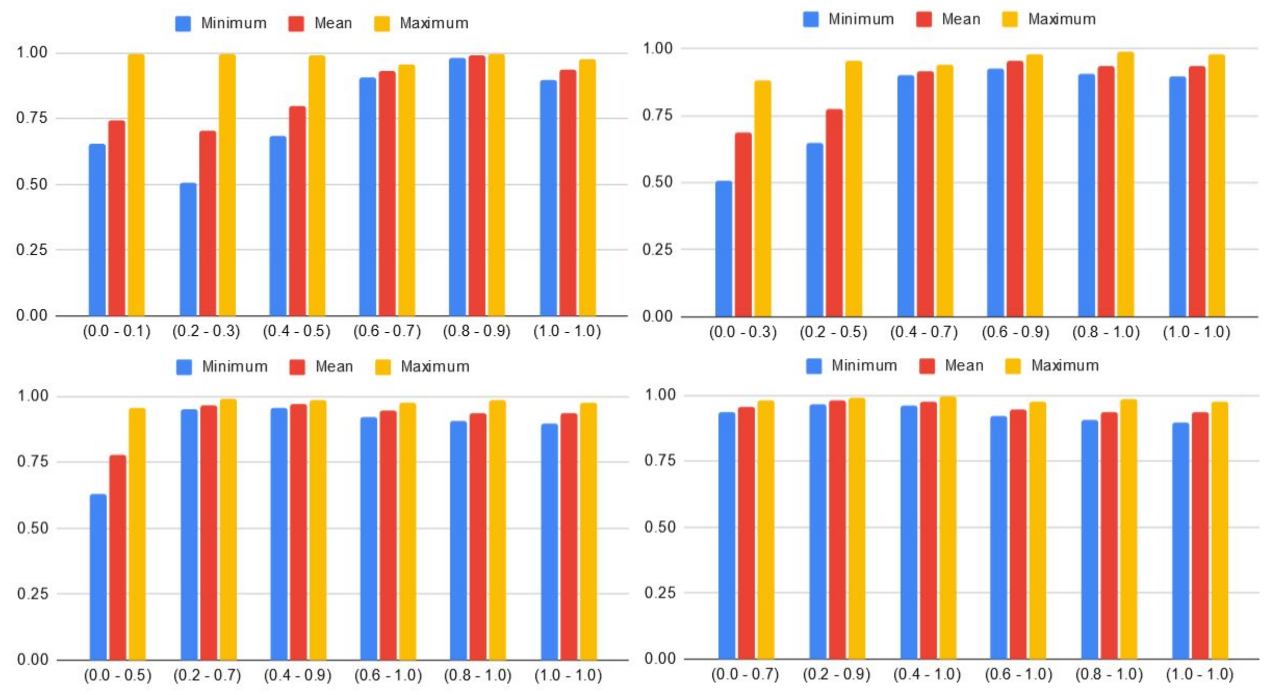

Figure 8: Agent influence (x) plotted against document similarity (y). Top left: range 0.1. Top right: range 0.3. Bottom left: range 0.5. Bottom right: range 0.7.

From all these findings there are two interesting, although tentative, conclusions we can draw about scribal transmission. We should, of course, be very careful with these conclusions since modelling can sometimes tell you more about your own biases in the making of the model than it does about the real world (Ivanovic & Freer, 2009). With these precautions out of the way however, these are the things our model seems to imply about scribal transmission: Firstly, the language proficiency of a scribe is far less important than one would initially guess, and in some cases it can actually hinder transparent transmission due to 'arrogance' of the scribe. Secondly, our model implies that geographic separation could be one of the main reasons for multiple parallel editions of a text existing.

After all this evaluation, it is now time to look back at one of our more improbable goals: using this model to look back into the possible history of a text. On one hand this is highly unlikely. The missing data is not easily replaced by what are essentially guided random manipulations to a text. On the other hand I do believe the model can tell us more about what possibly came before our oldest known manuscript of a given text. In all our examples we see how there are lots of phrases that will not alter. By running this model we can essentially generate a heatmap for a manuscript with locations that are most likely to contain errors. This could serve as a tool for editors, redirecting their attention to passages that are most vulnerable to corruption. True predictions about what ancestor manuscripts might have looked like are surely possible. They are, however, not what this model provides. In the next chapter we will talk about what kind of technology would enable this, as well as about what improvements we could make to our model to make it better suited for its purpose.

Another possible use for the model we have just created is as a tool to gain some insight into the scribes and transmission for a certain tradition. We have unearthed some interesting relations between qualities of the transmission and parameters of an agent. By looking at existing traditions we can therefore draw some conclusions or at least hypotheses about the likely nature of the transmission. For example, a tradition with many different versions could point to the scribes having very little overlapping influence.

7. Future work

There are some obvious improvements that could be made to this model that would greatly enhance its precision in modelling reality. Other alterations that come to mind are not as much clear improvements but more interesting experiments. We also, unexpectedly, were able to draw some hypotheses from this model. After we have discussed all of these things we then take a quick look at what different technological solutions could have brought us.

To start with our new hypotheses: it seems that the importance of language proficiency, as tuned by the ‘vocabulary’ parameter of an agent, is far lower than what we initially would have suspected. Research into the historical data we have concerning the language level of scribes looks to be the next step to turn this tentative hypothesis into more solid scientific knowledge. This hypothesis also opens the way for other research into the importance of language proficiency in written transmission, maybe even in our modern world.

The effect of geographical isolation on text traditions could be another interesting field of research. This could be studied by more closely modelling the dissemination of a particular tradition, based for example on historical data about trade and exchange of scientific knowledge. However, putting this idea into a more modern context, this model could provide insight into the influence of the internet in text transmission. The internet is, in terms of our model, the closest the real world has come to an infinitely small simulation space or agents with infinite influence, since every agent can potentially reach every other agent. Our model could then be used to explain some of the phenomena we are seeing on the internet today.

The first simple improvement to the model would be better word embeddings. Larger word embeddings that are created from a more complete corpus of greek texts will surely aid our semantic switching of words while also increasing the vocabulary of our agents. The manner of creating these embeddings might need some more research as well, since all research that we found was aimed at creating word embeddings for languages without a case system. In ancient greek the grammatical case is signified by the last few letters of a word.[46] It would be interesting to see if this case system has a negative impact on the word embeddings, since it essentially sees one word in two different cases as two different words.[47]

[46] “δοῦλος,” for example, is the subject of a sentence, whereas “δοῦλον” is the object of a sentence. Observe that there is only one letter that differentiates the two.

[47] Further research into the relations between the word vectors of one word in different cases needs to be done before we can draw any conclusions about the accuracy of our current embeddings.

One improvement of the model that would be trivial to implement would be the inclusion of diachronic parameter mutation. Over time, as the simulation progresses, we could change various parameters. This would allow us to simulate very diverse traditions, or traditions in which large abrupt changes happen, such as the invention of the printing press. At this point we could even start thinking about word embeddings that have some notion of time embedded into them, since words and their meaning are not set in stone, but change over time.

The next obvious improvement would be the inclusion of a standardised and proven list of errors and alterations that scribes make. Preferably with statistics about their occurrences included. When we initially started this research we naively assumed such a list existed. The benefits of such a list would be tremendous. It would allow us to be much more precise about our predictions, since we would know the likelihood of mistakes happening in certain places. Since it is unlikely that such a list would be accompanied by a full personal description of the scribe that made the error, this list would also make a good case for switching over our model to a different approach, such as a genetic algorithm. There are ways to create this list of errors and alterations with automated analysis of parallel or otherwise related editions. However, we believe that manual analysis of scholarly editions would yield far more accurate results, since these critical editions often contain much more data about an error other than the fact that it happened. Unfortunately this would be a monstrous undertaking which is unlikely to be done in the near future.

Another method to improve the model would be to run it on more manuscript sources, preferably some that are directly related.[48] This could tell us much about how well the model reflects the reality of what happened, effectively testing the model against reality. Doing this would make for some interesting and doable further research, since most of the required texts, at least those manuscripts of the Iliad, are already available from the Homer Multitext (2020). If we expand the scope of the project somewhat, and are willing to generate new word embeddings, we could even try to model other languages, such as latin, old english, or even languages with a wholly different writing system such as chinese, korean or any of the other southeast asian languages. These languages would however necessitate some minor tweaks to some of the letter manipulation scripts since characters have a completely different meaning in a writing system that is non alphabetical.

[48] For example: by comparing a modern scholarly edition, which represents the scientific consensus of the archetype, the ‘true’ text, to a manuscript edition.

With some more extreme modifications this model could even be used to simulate oral transmission of stories, greatly broadening the scope of its usefulness in the scientific community. The agents do not understand that what they are transmitting is a text. Only the nature of the errors that are introduced, backed up by the handout of Boter (2013), makes this model suited for transmission of written text. If such a list, or better yet a statistical analysis, of errors that storytellers and listeners make were to exist, the implementation would be relatively trivial. Combining this kind of research with the already existing knowledge would allow for a hybrid model, a model that simulates both written and oral text transmission.

One of the more considerable changes we briefly entertained was the idea of creating this model using Evolutionary Algorithms, genetic algorithms to be more exact. By encoding the text as the DNA of a particular individual in the simulation we could then have let 'natural' selection do its job. By controlling the selection parameters and the fitness function you could no doubt create a model with similar possibilities to our current agent based approach. The agent based approach mostly has the advantage of being easier to compare with the real world, as it lets us think of scribes acting on text. A genetic algorithm would necessitate us to abstract away the role of scribes into the selection and fitness criteria. This approach could provide interesting insights into the biases that our agent based model gave us through its design.

Let us then discuss the elephant in the room: machine learning. Neural networks have made huge leaps forwards in the past years. The field is evolving so rapidly that papers with the latest insights are invalidated within a couple of months, only to be replaced by the newest insight. In the opening chapters we discussed them briefly as an option, only to redirect our attention to agent based models, which seemed like a more natural fit. We still stand by that statement, but not for the same reasons. With enough data a neural network could be trained to simulate this model, and possibly go far beyond it by predicting what possible ancestor manuscripts could have been. The problem however lies not with the neural networks, but their hunger for data. There is very little data about what errors scribes have made. So little actually that even making an abstract agent based model proved to be a challenge in that regard. If enough data could be collected the possibilities would be far beyond anything this model can do. The current digital landscape does unfortunately not meet the requirements for such a training task. If in a couple of years more of our manuscripts are digitised, just as the inscriptions had been digitised for Assael et al. (2017), then we will gladly take a new look at the possibilities.

8. Conclusion

To conclude this thesis we will run through all the findings, observations and thoughts we have expressed over the past chapters. To begin we went through all the necessary background knowledge about scribal text transmission as well as some specific background knowledge on the text we are using. Before we continued with our discussion of the actual implementation we also gave some general information about agent based models, and our reasons for picking them. Then we started talking about how we actually implemented this model. Our data was extracted from a `.cex` file, and only references useful to us were saved to separate text files. Word embeddings were made with the *gensim* python library on a large corpus of greek texts. In the rest of this chapter we ran through some of the most important parts of our implementation.[49]

The evaluations gave us some insight into the effect of different parameters, as well as their useful parameter range. Although the model proved to be unable to give us a glimpse in the past, indirectly it could still provide inspiration. We could easily visualize areas in the text where corruption would be probable. In this role it could be useful for an editor of classical texts who wants to know where their attention should be focussed. We also were able to make some interesting tentative hypotheses about the nature of scribal transmission. There even was a clear possibility of studying the historical context of scribes using comparisons with real life data. This sort of experiment would provide further insight into parts of a text transmission tradition that might already be partially lost.

In the next chapter we discussed different solutions to existing problems, such as the size of the word embeddings and the imprecise nature of Boters list (2013). We also looked at various possibilities to open up the scope of the model, to make it more useful for other languages, such as latin, english or even southeast asian languages. We even observed that there were large parallels between this model and a hypothetical oral transmission model, opening up the idea to model oral or even hybrid traditions. With some modification we could even provide insight into the nature of story transmission on for example the internet, due to the abstract nature of the model.

After all that we looked at the options that other architectures, such as neural networks and evolutionary algorithms could provide. While the prospects for both technologies were exciting, it looks like the data is currently not available for neural networks to make a feasible project. Genetic algorithms could be an interesting

[49] The complete code can be found on the GitHub repository:
https://github.com/MGelein/simulating-mistakes

alternative to the agent based models we are using right now, but at the same time there is no good reason to pick them over our existing agent based approach.

We started this paper by wondering if we could solve the Homeric Question using computer models and simulation. The short answer to this is no. The longer answer is: maybe, but not now. Our current model is able to provide scientists some guidance in their decision making, but anything more than that is unreliable due to the abstracted nature of the data sources this model was built upon. We could however already make some interesting hypotheses about the nature of text transmission and the tradition that surrounds it by looking at the characteristics of an existing example contrasted with the results of our model. Further improvements in the field of digital humanities will hopefully make us able to solve the Homeric Question problem.

9. Bibliography

- Almas, B., & Berti, M. (2013, September). Perseids collaborative platform for annotating text re-uses of fragmentary authors. In *Proceedings of the 1st International Workshop on Collaborative Annotations in Shared Environment: metadata, vocabularies and techniques in the Digital Humanities* (pp. 1-4).
- Ancient Greek Dependency Treebank, version 2.1 (2021), downloaded on January 5, 2021 from https://perseusdl.github.io/treebank_data/
- Abuata, B., & Al-Omari, A. (2018). A rule-based algorithm for the detection of Arud meter in Classical Arabic poetry. *International Arab Journal of Information Technology*, *15*(4), 1-5.
- Assael, Y., Sommerschield, T., & Prag, J. (2019). Restoring ancient text using deep learning: a case study on Greek epigraphy. *arXiv preprint arXiv:1910.06262*.
- Bietti, L. M., Tilston, O., & Bangerter, A. (2019). Storytelling as adaptive collective sensemaking. *Topics in cognitive science*, *11*(4), 710-732.
- Blackwell, C. N., & Smith, N. (2019). The cite architecture: a conceptual and practical overview. *Digital Classical Philology*, 88-101.
- de Boer, B. (2006). Computer modelling as a tool for understanding language evolution. In *Evolutionary epistemology, language and culture* (pp. 381-406). Springer, Dordrecht.
- Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the national academy of sciences*, *99*(suppl 3), 7280-7287.
- Cuevas, E. (2020). An agent-based model to evaluate the COVID-19 transmission risks in facilities. *Computers in biology and medicine*, *121*, 103827.
- Dué, C., & Ebbott, M. (2014). An Introduction to the Homer Multitext Edition of the Venetus A Manuscript of the Iliad. *The Homer Multitext (www. homermultitext. org)*.
- Dunbar, R., Gamble, C., & Gowlett, J. (2014). Thinking big: How the evolution of social life shaped the human mind. Thames & Hudson.
- Fisher, N. R. (2015). hubris. In *Oxford Research Encyclopedia of Classics*.
- Fowler, R. (2004). The Homeric Question. *The Cambridge Companion to Homer*, 220-32.
- Gomez, J., Prieto, J., Leon, E., & Rodriguez, A. (2020). INFEKTA: a general agent-based model for transmission of infectious diseases: studying the COVID-19 propagation in Bogotá-Colombia. *medRxiv*.
- Goody, J. (2006). From oral to written: An anthropological breakthrough in storytelling. *The novel*, *1*, 3-36.
- Helbing, D. (2012). Agent-based modeling. In *Social self-organization* (pp. 25-70). Springer, Berlin, Heidelberg.

- Homer Multitext, (2020) hmt-2020i. From: https://github.com/homermultitext/hmt-archive/tree/master/releases-cex
- Ivanovic, R. F., & Freer, J. E. *(2009). Science versus politics: truth and uncertainty in predictive modelling. Hydrological Processes, 23(17), 2549.*
- Kaše, V. (2019). *Tracing the Origins of Eucharistic Magic: On the Role of Cognitive Attraction in the Cultural Transmission of Collective Rituals.*
- Li, B., & Han, L. (2013, October). Distance weighted cosine similarity measure for text classification. In *International conference on intelligent data engineering and automated learning* (pp. 611-618). Springer, Berlin, Heidelberg.
- Liu, Y., Liu, Z., Chua, T. S., & Sun, M. (2015, February). Topical word embeddings*. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 29, No. 1).*
- Mathijsen-Verkooijen, M. T. C. (1995). Naar de letter. Handboek editiewetenschap. KNAW Press.
- Mathijsen, M. (2010). Vijftien jaar Naar de letter. *Naar de letter: handboek editiewetenschap.-4e ongew. opl.[met nawoord]*, i-vi.
- Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., & Joulin, A. (2017). Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405.*
- Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.
- Mueller, M. (2013). *The Iliad*. A&C Black.
- Nagy, G. (2009). *Homeric questions*. University of Texas Press.
- Parry, M. (1930). Studies in the epic technique of oral verse-making. I. Homer and Homeric style. *Harvard Studies in Classical Philology*, *41*, 73-147.
- Rahutomo, F., Kitasuka, T., & Aritsugi, M. (2012, October). Semantic cosine similarity. In *The 7th International Student Conference on Advanced Science and Technology ICAST* (Vol. 4, No. 1, p. 1).
- Ramos, J. (2003, December). Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning* (Vol. 242, No. 1, pp. 29-48).
- Rong, X. (2014). word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738.*
- Royse, J. R. (2013). Scribal Tendencies in the Transmission of the Text of the New Testament. In *The Text of the New Testament in Contemporary Research* (pp. 461-478). Brill.
- Tata, S., & Patel, J. M. (2007). Estimating the selectivity of tf-idf based cosine similarity predicates. *ACM Sigmod Record*, *36*(2), 7-12.
- Tyers, B. (2016). INI Files. In *Practical GameMaker: Studio* (pp. 155-160). Apress, Berkeley, CA.

- Van Rossum, G. (2007, June). Python programming language. In *USENIX annual technical conference* (Vol. 41, p. 36).
- De Weerdt, H., Gelein, M., & Ho, B. COMPARATIVUS: A Text Comparison Platform. 2017.
- De Weerdt, H., van Beijeren, G., and Gelein, M. (2019) “Reading The Essentials of Governance from Taizong’s Reign Digitally.” https://dh.chinese-empires.eu/zgzy/ (publication in progress)
- West, M. L. (1973). *Textual criticism and editorial technique applicable to Greek and Latin texts*. Walter de Gruyter.
- West, M. (2011). The homeric question today. *Proceedings of the American Philosophical Society*, *155*(4), 383-393.
- Wilson, N. G., & Reynolds, L. D. (1991). *Scribes and scholars: a guide to the transmission of Greek and Latin literature*. Clarendon Press.

Appendix A: Excerpt from handout by Gerard Boter (2013)

Tot ca. 1450 werden alle teksten steeds opnieuw met de hand gekopiëerd. Ieder exemplaar was uniek. Bij het kopiëren werden fouten gemaakt.

Voorbeelden van fouten

- dicteerfouten: "hij koude op zijn pen"; “uitwijden”; "er valt geen pijl op te trekken." N.B. dicteren kan ook "inwendig", dus zonder hardop spreken, gebeuren. In het Grieks o.a. iotacisme (de uitspraak van ι , ε ι , η , ῃ, ο ι , υ als /i/:
  αἰπύ > ἐπ ε ί; ἤ δ η > ἴ δ ο ι ; β als /v/ uitgesproken: ε ὗ ρ ο ν > ἕ β ρ ο ν .
- transpositie = omzetten van letters: lopen > polen; tobben > botten; ἔ β α λ ε ν >
  ἔ λ α β ε ν .
- associatie: je denkt ergens aan, en daardoor schrijf je het ook op. Bijvoorbeeld: "Na een kwartier in de parfumerie allerlei luchtjes te hebben opgesnoven deed hij opgelucht de geur open." πύ λ α ι > θ ύ ρ α ι .
- anticipatie = vooruitlopen: "Met een geweldige trap schoot hij de man naar de voorste man". Plato, R. 611a1 δ ῆ λ ο ν ὅ τ ι ἀ ν ά γ κ η α ὐ τ ὸ ἀ ε ὶ (ε ἶ ν α ι D)
ὂ ν ε ἶ ν α ι .
- perseveratie = doorwerken: "De keeper trapte de bal naar de voorste bal". Op een Griekse vaas aan de ene kant de handtekening: Χ Σ Ε Ν Ο Κ Λ Ε Σ Ε Π Ο Ι Ε Σ Ε Ν; op de andere kant Χ Σ Ε Ν Ο Κ Λ Ε Σ Ε Π Ο Κ Λ Ε Σ Ε Ν.
- haplografie = iets één keer schrijven: De Boelelaan > De Boelaan. ἂ ν ἀ ν ά γ κ η >
  ἀ ν ά γ κ η .
- dittografie = iets twee keer schrijven: "U kunt dit nalezen op de de website".
  ἀ ν ά γ κ η > ἂ ν ἀ ν ά γ κ η .
- omissie = weglating, vaak ten gevolge van le saut du même au même = de sprong van hetzelfde naar hetzelfde. "De schoolleiding heeft ten eerste besloten dat leerlingen uit de lagere klassen geen alcohol mogen drinken; verder is besloten dat brugklassers om elf uur naar huis moeten." Dit wordt dan, doordat het oog verspringt van het eerste naar het tweede “besloten”: "De schoolleiding heeft ten eerste besloten dat brugklassers om elf uur naar huis moeten." Plato, R. 328d1-2 (...) ο ὐ δ ὲ ν ἄ ν σ ε ἔ δ ε ι δ ε ῦ ρ ο ἰέ ν α ι ,
  ἀ λ λ ' ἡμ ε ῖς ἂ ν

π α ρ ὰ σ ὲ ἦ μ ε ν · ν ῦ ν δ έ σ ε χ ρ ὴ π υ κ ν ό τ ε ρ ο ν δ ε ῦ ρ ο ἰ έ ν α ι :
ἀ λ λ '-ἰ έ ν α ι om. D.

- glossen = toelichtingen. Bijvoorbeeld: in een tekst staat "wie een kuil graaft voor een ander, valt er zelf in". Iemand zet als toelichting in de kantlijn: "dit spreekwoord zegt dat mensen met kwade plannen daar zelf de dupe van worden". Een volgende kopiïst denkt dat dit in de tekst hoort te staan, en hij schrijft: "wie een kuil graaft voor een ander, dit spreekwoord zegt dat mensen met slechte plannen daar zelf de dupe van worden, valt er zelf in." Plato, R. 575c4 ο ὐ δ ' ἴ κ τ α ρ β ά λ λ ε ι] ο ὐ δ ' ἴ κ τ α ρ ἐ γ γ ὺ ς ἐ σ τ ὶ δ ὲ +++ ο ι μ ί α ὥ σ π ε ρ κ α ὶ τ ὸ
  ο ὐ δ ' ἴ κ τ α ρ ἥ κ ε ι ς β ά λ λ ε ι F.
- leesfouten: in handschrift worden letters vaak met elkaar verward: nn/m (bomen/bonnen); c/e/l (belt/beet). In Grieks majuskelschrift: Α Δ Λ ; Γ Τ ;
  Ε Θ Ο ;
  Λ Λ Μ ; Η Ν .
- woordscheiding: in de oudheid schreef men zonder spaties en leestekens; bovendien had men alleen hoofdletters. Welke drie mogelijke betekenissen zie je in GEEFJANTIENPATAT? In het Grieks: Γ Ε Τ Ι = γ ' ἔ τ ι of γ έ τ ι ;
  Δ Ε Ι Δ Ε = δ ε ῖ
  δ έ of δ ὲ ἰ δ έ.