Word Embeddings for the Morphosyntactic Analysis of the Voynich Manuscript

Adrien Schurger-Foy
Department of Computer Science, EPFL, Switzerland
Faustine Rossi
Department of Life Sciences Engineering, EPFL, Switzerland
Alexandre Tkatch
Department of Microengineering, EPFL, Switzerland

I. Introduction

The Voynich manuscript is an undeciphered piece of handwritten work illustrated with numerous drawings, which has been dated to the early 15th century. In this project, we aim to use word embeddings to capture morphosyntactic as well as lexical-semantic patterns in the manuscript. This project is an extension of a previous work that focused on suffixes [1]. This earlier work has found clusters of certain suffixes, and the nearest neighbours of many words only differ with regards to the input word by one or two characters. We will explore the spatial distribution of words with common prefixes by experimenting with different word embeddings. The idea is that words with a common morpheme tend to be grouped together if the morpheme is a prefix.

We will first determine the performance of the models on a Latin text of approximately the same size, after which we will apply the best model to the Voynich. We will take a step back from the previous work and try to enrich the knowledge about Voynich.

II. Data

We selected a reference text in Latin, named "Secreta Secretorum" to evaluate the different embeddings obtained. This medieval document is a treatise, divided into sections[2]. The Voynich manuscript is also assumed to be separated into different sections and thought to be an instruction manual[3]. It also contains around the same number of words as the Voynich manuscript: $|\Omega|_{ref} = 39400$ against $|\Omega|_{Voynich} = 28893$. The vocabulary size is also comparable: $|\nu|_{ref} = 1237$ against $|\nu|_{Voynich} = 991$.

III. Methods

A. Model used

The proposed word embeddings are created using the fastText model [4], tested with the skip-gram or cbow architectures. fastText is an extension of the word2vec algorithm, in which a word is split into n-grams (a subword with n letters). Then the architecture attributes a vectorized representation of this n-grams, and finally sums them to obtain the complete representation of the word. Furthermore, the cbow architecture tries to predict a word from a context, while skip-gram predicts the context from a word.

We chose to leave some parameters with their default value that you can find in appendix A.

We also fixed the loss function to the **softmax function** (which is not the default function for the Windows version of fastText) and the minimal frequency of occurence of a word in the text to **5**. To create such a model, please refer to the function *Create_model.py* from our GitHub and its description. A simple tokenization is done before constructing the model, but when it was necessary (for example with the Vonich manuscript), we used a more complex pre-tokenizer upstream from this function and used the tokenized file to run *Create_model.py*. We need this additional step since the transcriptions have information fasttext can't process, as explained below.

B. Pre-processing

The tokenizer used by the Python module fastText is inappropriate for the Voynich manuscript. But Secreta Secretorum only contains spaces, so we can use it directly to create a model, as indicated in the section III-A. We created then one tokenizer on the latest transcription of the Voynich [5] as followed:

- Special ASCII code characters (e.g. @192) were simply replaced with '@', they are rare and thus would not be taken into consideration by fastText anyway.
- Ligatures (denoted by '{...}') were kept.
- When multiple possible transcriptions are available (denoted by [x:y:z]), the most likely is taken (the first).
- New lines, certain and uncertain word separators (denoted by '.' and ',' respectively) were all considered separators.

C. Hyperparameters

fastText models are built from a lot of parameters. But the parameters that we consider as meaningful for our purpose are the minimum **(minn)** and maximum **(maxn)** length of a n-gram, the learning rate **(lr)** and the embedding dimension **(dim)**. We created then different models, for different combinations of these parameters and evaluated

which one is the most effective in terms of clustering by words with a common prefix. We denote for the next sections $(\chi_i)_{1\leq i\leq n}$, the n models we created by playing on the hyperparameters we defined above.

D. General protocol

As the Voynich manuscript is undeciphered yet, we need a reference text to evaluate if, for a given model (χ_i), a morphosyntactic clustering (more precisely words with a common substring that is morphologically meaningful) happens or not. But to reinforce our confidence in the embedding model, we also decided to work on suffixes, to compare our results with the voynich2vec embedding proposed by [1].

We followed the protocol described below:

Protocol:

1) **Random corruption** of Secreta Secretorum with **15 different fake words** with the morpheme brl- (not a prefix) or the morpheme -prl (-prl nor -rl nor -l are prefixes in latin). Each word is added **at least 5 times**
2) **Creation of a word embedding** on the tokenized and corrupted version of Secreta Secretorum with a set of parameters χ_i
3) From the model, selection of **the most abundant potential prefixes and suffixes** of length in range [1:4]
4) Calculation of the **set's 'distinctness'**: the mean inter-set distance divided by the mean intra-set distance
5) t-SNE representations of the embedding space serve as a sanity check and will show strange things if something is wrong with the model (this allowed us to find a decent learning rate for example)
6) For a given architecture (either skipgram or cbow), **intrinsic evaluation** for a given set of parameters {minn,maxn}: the corrupted words should not cluster because they don't contain prefixes. Furthermore, prefixes present in the text should cluster together
7) Comparison of the different results obtained by the two architecture: the model that rejects brl- and -prl clusterings, as well as considers that prefixes (found in the literature) cluster together is the best one
8) **Reduction of the dimension** and observation of the results until a significant change: selection of the dimension without damages

E. Qualitative tools

This tool refers to the step 7) of the protocol. From the step 3), we selected the most abundant potential prefixes and suffixes in the text. We decided to study the 15 most common morphemes of length 1, 2, 3, and 4 resulting in a total of 60 potential prefixes in Secreta Secretorum. We reason that the embeddings will be most reliable for frequently occurring morphemes, as the model will have trained on many instances of such morphemes. We then did a qualitative analysis on the morphemes found, thanks to online ressources and determined whether a morpheme is a prefix/suffix or not. To then attest that embeddings did in fact show clustering by words with a common morphosyntactic prefix, we implemented a t-SNE visualization thanks to the sklearn and pandas librairies, to reduce the dimension space from dim $\geq$ 50 to 2, so that we can better understand the word embedding space. But unfortunately, its use was limited to the approximate evaluation of the learning rate (step 5), because we remained skeptical about the clusters formed by the t-SNE. Indeed, we loose a lot of information by reducing the dimension from 100 to 2 and the results are for us unreliable (for more details about the t-SNE results, see B).

F. Quantitative tools

To numerically quantify how well words with the same prefixes are clustered together, we calculated what we call 'distinctness': the interclass distance divided by intraclass distance.

$$\frac{\frac{1}{|U|}\sum_{e_1\in C}\sum_{e_2\in U\setminus C} D_{cos}(e_1,e_2)}{\frac{1}{|C|}\sum_{e_1,e_2\in C} D_{cos}(e_1,e_2)}$$

Where C is a class (set of embeddings of words with a given morpheme), U is the set of all embeddings, and D_{cos} is the cosine distance between embeddings. To understand which distinctness scores are meaningful, we calculated the distinctness of 100 random sets of embeddings, for different sized sets as shown in figure 1.

IV. Evaluation

We will detail the results on the prefixes with a length of 3, but the analysis was done similarly on length = 1,2,4, and on the suffixes. You can find these results in appendix C.

A. Qualitative analysis

From the literature, we determined the nature of the 14 most abundant morphemes in Secreta Secretorum (we excluded the corrupted one). The table I shows as an example 10 of them whose length is 3.

Prefixes List		
Prefix	Pourcentage in the text (%)	Meaning
qua	1.3	Relative pronoun, ”which way”/”as”
con	2.7	Derived from ”co-”
ver	0.8	”true”
ven	0.8	”come”
ali	2.0	”other”/”different”
pro	1.3	“in favour of”/”on behalf of”
fac	1.2	”do”/”make”
qui	1.0	Relative pronoun ”who”/”that”
reg	1.0	”rule”/”control”
bon	1.0	”good”

Table I: *10 morphemes with 3 letters, their abundance in the vocabulary of 1247 words (including the corrupted words), and their meaning, if it exists. The grey rows indicate morphemes that are in fact not prefixes. The green indicate prefixes. White indicate root.*

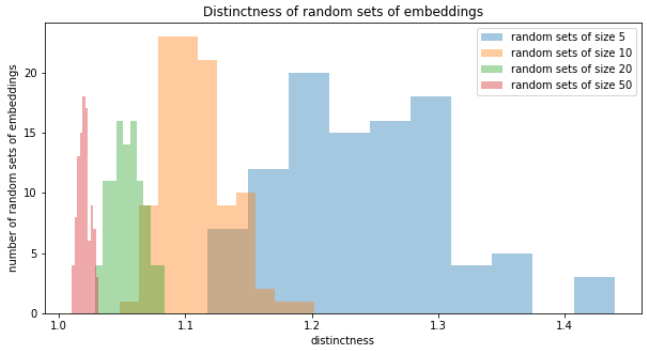

Figure 1: *Distributions obtained by calculating 100 times, the cluster distinctness on different subset sizes*

B. Quantitative analysis

We found that **lr=0.005** was good in terms of t-SNE shape we obtained, and as we continued to decrease it, the average loss stagnated. Given the parameter dimension d=100 we played with the parameters minn and maxn and the architectures (either skipgram or cbow). For skipgram, the only model that lead to a non-clustering of the corrupted prefixe was with minn=3, maxn=3. This is problematic because we don't want that the embedding depends on the prefix's length. So we reject the skipgram structure, which is surprising because [1] provides an embedding with this structure. However, they did not fully explain the results obtained on their intrinsic evaluation. We can afford to doubt the choice of the model. We tested then on the cbow architecture.

For minn=3 and maxn=3 or 6, we found that the words

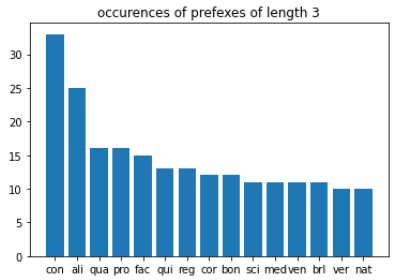

Figure 2: *15 most occurent prefixes of length 3 in the Secreta Secretorum vocabulary, after keeping the words with at least 5 occurences in the entire text, and adding words with the prefix brl-*

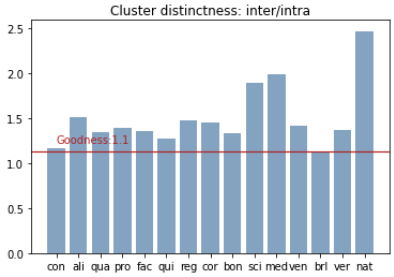

Figure 3: *Cluster distinctness of the 15 most occurent prefixes of length 3 in the corrupted Secreta Secretorum vocabulary. The cluster distinctness of brl- is represented with the red horizontal line*

with the prefix brl- did not significantly cluster together. Indeed, the measured cluster distinctness is 1.1 (figure 3). As the prefixe brl- is present around 10 times in the vocabulary as seen in the figure 2, the clustering could be purely random as suggested in the figure 1. With the model {minn=3,maxn=6}, independant from the morphem's length selected, the morpheme con- is not rejected. This is understandable since co- is the original morpheme. With the same logic as before, qui- does not cluster significantly, which could set limits to the embedding, **more sensitive to roots than to prefixes.**

We did then the same analysis on the suffixes. For the same model, the morpheme -prl is rejected, which is consistent. The other morphemes are found in the litterature to be mostly **suffixes**: they indicate cases/declinations or conjugations. The distinctnesses obtained are then relatively low, compared to what we obtained for the prefixes, but stay significant. This observation may strengthen the fact that the embedding is more sensitive to roots than prefixes. Finally, minn=3 and maxn=6 fixed, we decreased the dimension until important information is lost. Indeed, we have to be carefull, because **we work on a small dataset**, which increases the chances **to overfit.** The main criterium was that the barplot, as depicted in figure 3, became really flat (no prefixes found), under a certain dimension. That lead us to dim =

50, whithout damaging the model.

C. Final word embedding

The final word embedding is then chosen with the parameters from the section III-A, and with the followed hyperparameters:

dim	minn	maxn	architecture
50	3	6	cbow

These hyperparameters affect the cosine distance between embeddings, but we found that reasonable hyperparameter combinations lead to similar relative distinctnesses in the voynich text. In other words, prefixes with high distinctness remain higher than others even when hyperparameters are adjusted.

V. Voynich

A. Prefix selection

For the reasons described in III-E, we selected the 10 most common morphemes of length 1 through 4, giving a total of 40 morphemes.

B. Morphological analysis

In order to determine whether a set of embeddings form a good cluster, we look at the set's 'distinctness'.

Theoretically, if a set of embeddings do not form a good cluster, the distinctness will be close to 1. Conversely, if the ratio is significantly above 1, the set forms a distinct cluster. But what exactly counts as 'significantly' above 1? To answer this question, we calculated the distinctness of 1000 random sets of embeddings. We used random sets of size 5, 10, 20, and 50 as all notable sets of potential prefix words fall around this range.

As you would expect, it becomes increasingly unlikely to get high distinctness as the size of the random set increases. Even for a small set, we can see that achieving a distinctness of over 10 by 'accident' is unlikely, although not impossible (a few random sets out of 1000 reach 20). For a larger set, anyone would agree that 5 is a lower bound on what is not possible by 'accident'.

cho- stands out in the figure 7, as many words in our model start with it in addition to having the highest distinctness. We take this as evidence that it is a morpheme, although whether or not it is a prefix is uncertain. Many other things can be found from this analysis, but a few that stood out to us were *l-*, *a-*, and the longer (3-4 character) prefixes with high distinctness. The longer prefixes with high distinctness seem to be root words rather than prefixes: they always take up most of the word they are a part of. There are 34 and 49 words starting with *a-* and *l-* respectively, and their distinctness lies at around 2. As such, while these may not be true prefixes, it seems that words starting with these are in some way related.

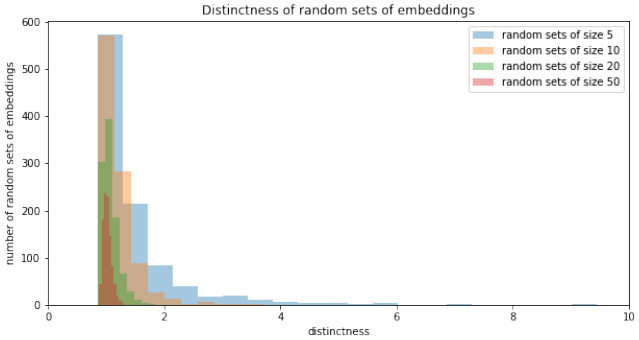

Figure 4: *Distinctness of random sets of embeddings*

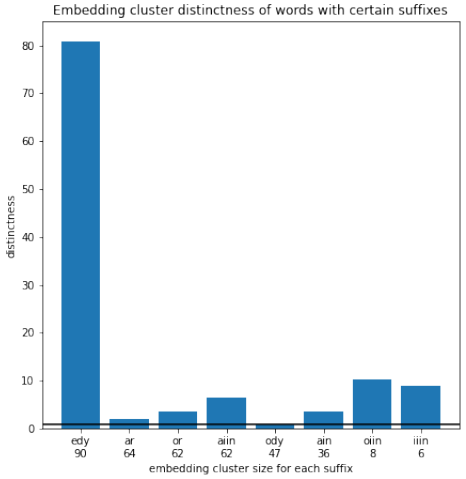

Figure 5: *Embedding cluster distinctness of words with certain suffixes*

C. Coherence with previous work

We ran the previous analysis but on suffixes instead of prefixes, and compared our results with those of the paper this project extends. In that paper, the authors focus on the suffixes in figure 5, and conclude that *-or* and *-ar* are not suffixes while *-aiin*, *-oiin*, *-iiin*, *-edy* and *-ody* are. Our method doesn't lead to the exact same conclusion. While there are no doubts regarding *-edy*, *-ody* actually has a distinctness less than 1, meaning words ending as such do not form a cluster at all in the embedding space. Furthermore, we cannot dismiss *-or* and *-ar* as they show sufficiently high distinctness in the context of figure 4 to be morphologicaly meaningful.

D. Spelling variant detection

The previous work also looks into whether *-aiin* and *-oiin* are in fact the same morpheme, we try to explore this with a different method. To try to explore this a stemmer

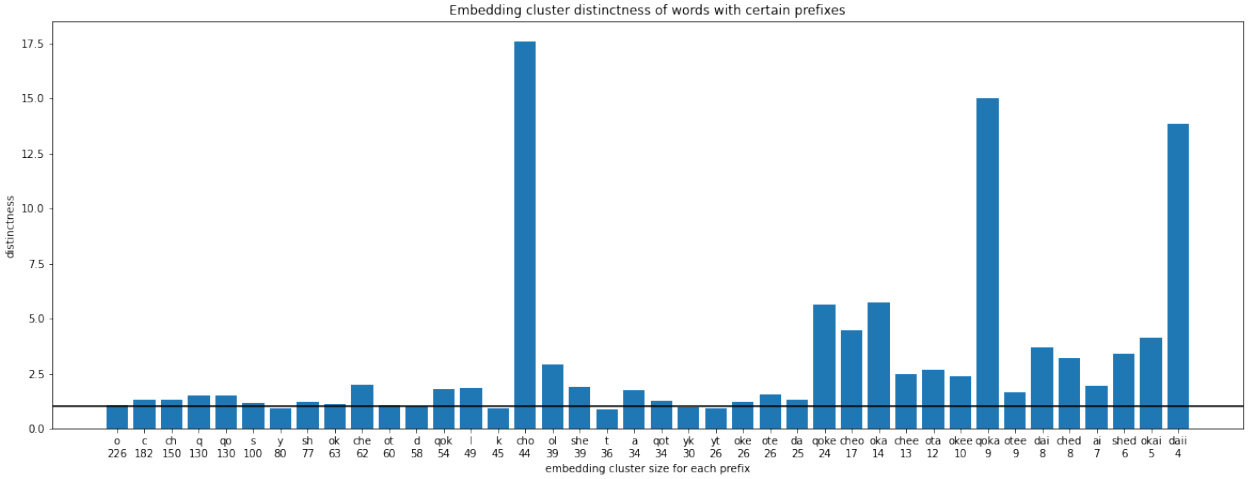

Figure 6: *Embedding cluster distinctness of words with certain prefixes*

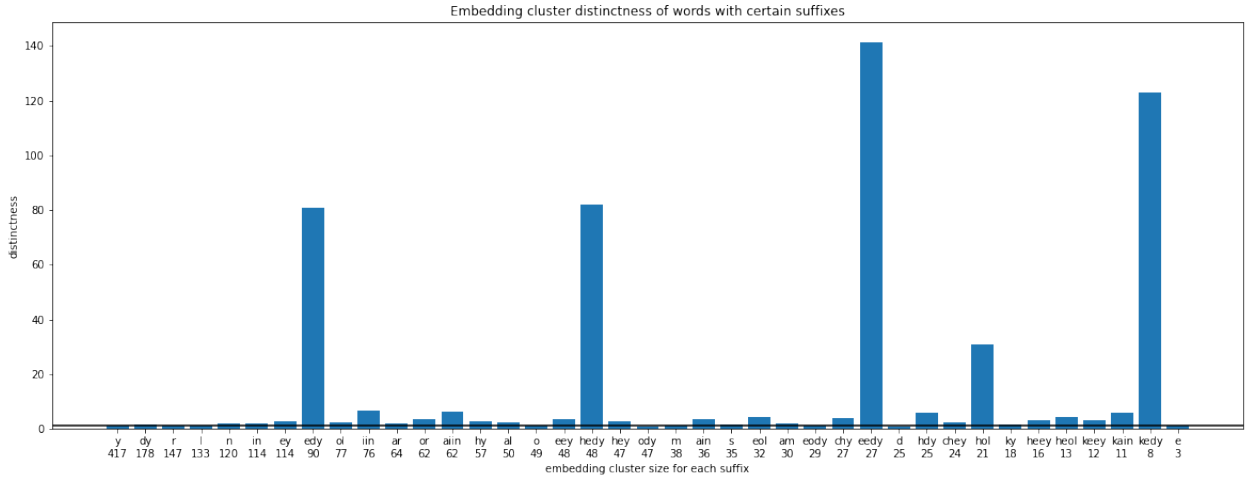

Figure 7: *Embedding cluster distinctness of words with certain suffixes*

was implemented to see if these endings could be grouped together. Unlike fastText, this stemmer is a clustering based on various distance metrics on the words themselves. The idea was to extend this analysis to prefixes, other suffixes and words in general, and incorporate embedding distinctness information, but we did not have time to complete everything planned. For more details on the implementation of this stemmer please refer to the appendix.

VI. Conclusion

In conclusion, this project builds on and attempts to extend the work that attempts to identify suffixes in the Voynich manuscript using fastText [1]. Although fasttext is also used in this project, the architecture chosen is different (cbow instead of skipgram). Another aspect that was examined is the attempt to use other metrics for the prefix detection. Indeed the t-SNE visualization can be misleading because it does not directly show clusters based on distance. Given this news metric (distinctness), our results differ for some morphemes. In particular, words ending with -ody do not seem to form clusters (distinctness lower than 1). Moreover, the endings -or and -ar showing a rather high distinctness could still be considered as potential prefixes.

For future works it could be interesting to reinforce qualitative tools with more linguistic knowledge. There also parameters to tune such as the occurence frequency and the context window (ws). Other architectures such as GloVe could also be tried to see if the clustering can be improved.

APPENDIX

A. Default parameters used

ws	epoch	min Count Label	neg	word Ngrams	bucket	lr Update Rate	t	verbose
5	5	0	5	1	2e6	100	1e−4	2

Table II: *Default parameters kept for the fastText algorithm, additional description in [6]*

B. t-SNE results

First, we thought we could exploit the strength of such a representation to see clusters appearing. According to the important role of the "perplexity" hyperparameter described in [7], we played on it to obtain a good visualisation. As shown in the figure 10 but more precisely in the figure 8, a too high (respectively a too low) perplexity produces small "aggregates" in background that we don't want. 9 tends to create an "homogeneous" shape and seems to indicate little clusterings. However, this unreliability has led us not to rely on these visual results. But interestingly, we found that the t-SNE created some weird cosine-shapes when the learning rate was too high 11. We exploited then the t-SNE algorithm to have an idea of how to parametrize the learning rate.

C. Detailed distinctness for different lengths of morpheme

D. Analysis on another reference text

As the German language possesses more prefixes and suffixes than the Latin language, we also decided to extend the analysis on a German text, to see if it reinforces the latin results. We used a small text extracted from the Europarl corpus that we truncated to have 30 000 words, which is around the same number of words as the Voynich. We tokenized it before using it (function *tokenizer_german_text.py* on the GitHub). Each line that began with the character '<' was suppressed. The punctuation and some special characters are retrieved, but not the accentuation such as umlaut, which are important for the German language. At the end, the tokenized file contains one word per line, each word in lower case. We applied the final model we chose in section IV-C on this text.

We found some common German prefixes such as "ein-", "un-" (more precisely, an analysis is needed for the morphemes with length = 5, to see if the prefix is rather "unter-" than "un-", "unt-" and "unte-" that have a significant distinctness). That is also the case for "durch-". "werd-" wird also as prefix detected, which is normally not the case. Indeed, the parsing did not cut well the German text, so that unwanted words are formed like "werdenwir" instead of "werden","wir". We will improve the parsing, but most importantly, we always have to check, even if a distinctness is significantly high, which words are concerned.

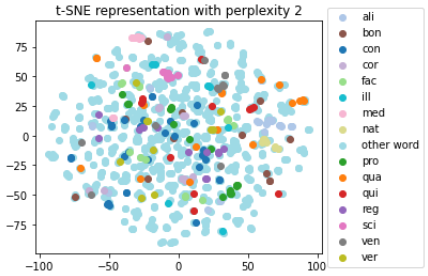

Figure 8: *t-SNE of the final model chosen (IV-C), for perplexity = 2*

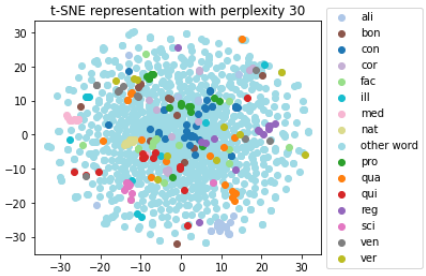

Figure 9: *t-SNE of the final model chosen (IV-C), for perplexity = 30*

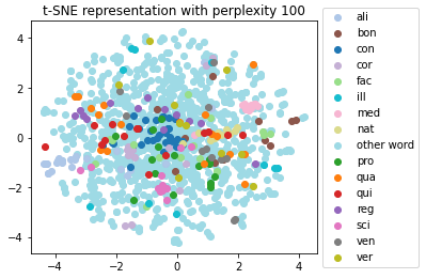

Figure 10: *t-SNE of the final model chosen (IV-C), for perplexity = 100*

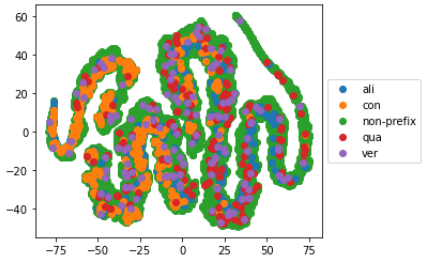

Figure 11: *t-SNE of a cbow model with minn = 3, maxn = 6, with lr = 0.1, for a perplexity = 30*

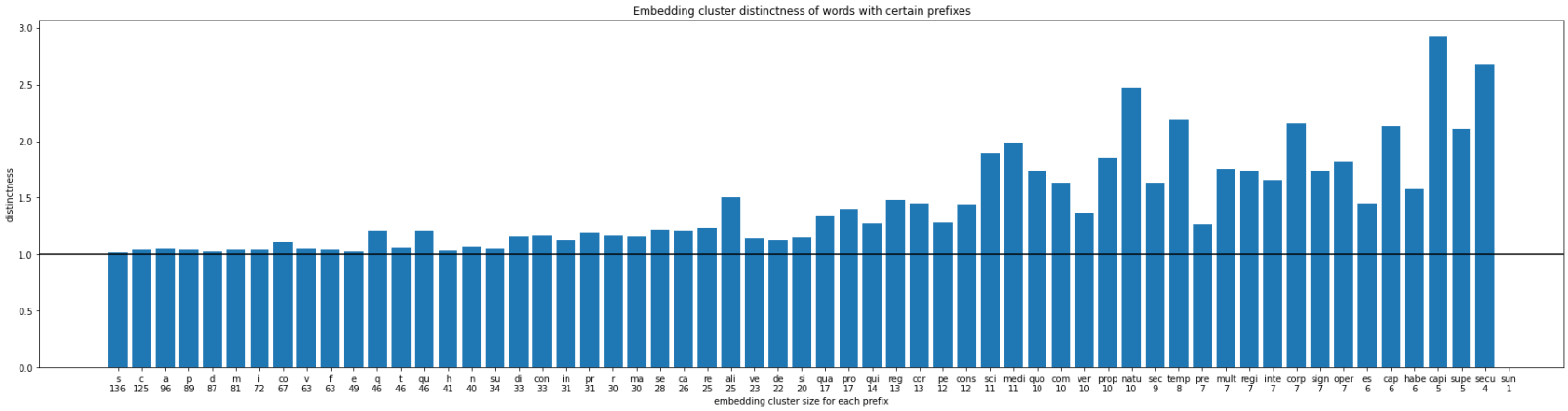

Figure 12: *Cluster distinctness of the most occurent potential prefixes of lengths between 1 and 4 in Secreta Secretorum*

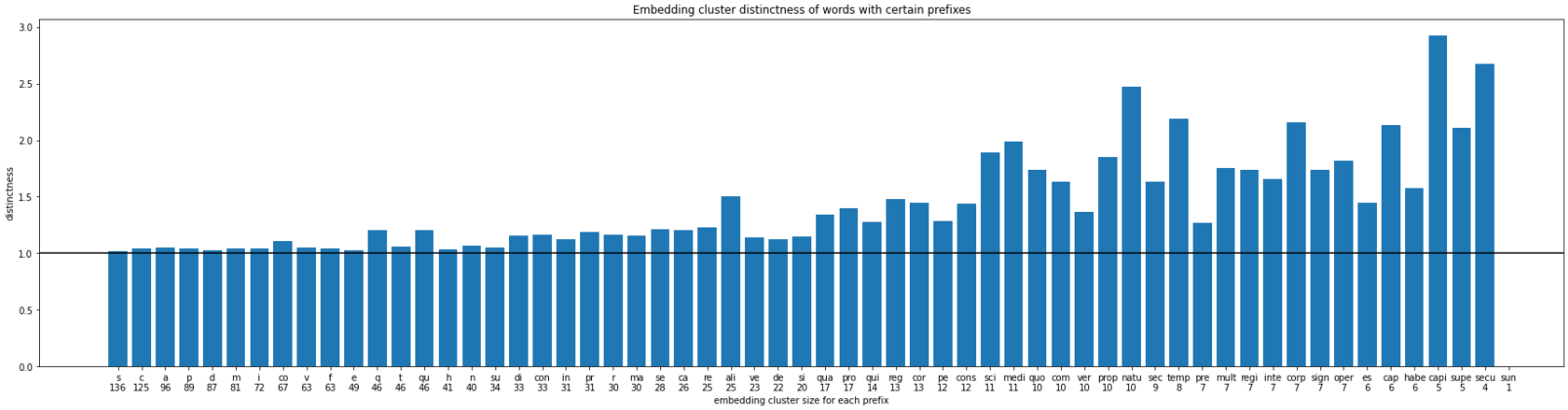

Figure 13: *Cluster distinctness of the most occurent potential prefixes of lengths between 1 and 4 in Secreta Secretorum*

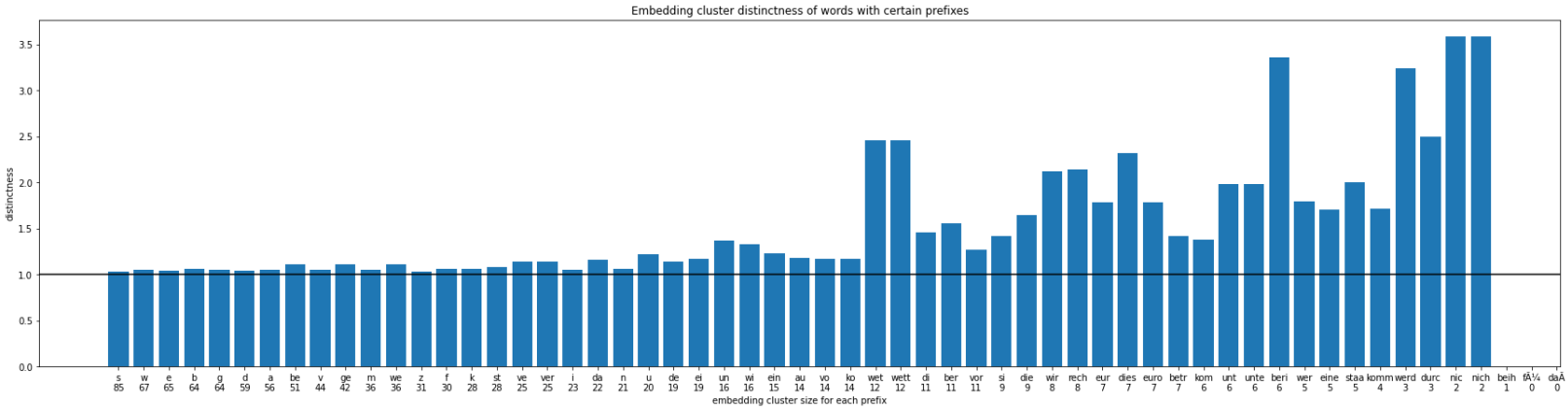

Figure 14: *Cluster distinctness of the most current potential prefixes of lengths between 1 and 4 in the German text*

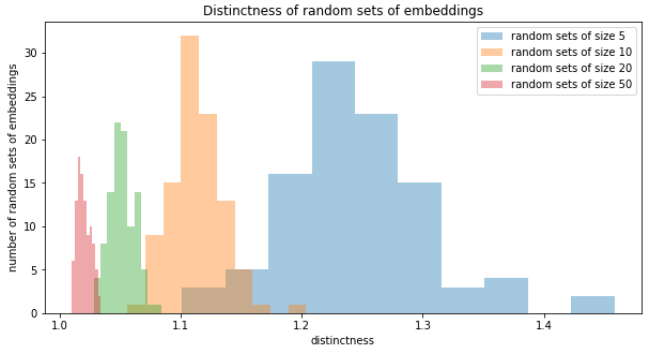

Figure 15: *Distributions obtained by calculating 100 times, the cluster distinctness on different subset sizes*

For other morphemes such as "wet-" or "wett-", we need a deeper linguistic analysis, because we did not found any signification for it. However, globally, the model seems also to work properly on the German text.

E. Spelling variant detection

The spelling of ancient languages is not as formalized as those of today. This may be a problem for the Voynich manuscript where the same word could have been written in different ways. For this reason an attempt has been made to implement a variant spelling detector based on different distance notions that have been proposed in the literature. There are three advantages of this approach. The detector is language-agnostic and doesn't require a large amount of data. We can also use words instead of word vectors [8]. These reasons make this method useful for the pre-processing step before subsequent analyses of the Voynich manuscript.

1) Pre-processing: A labeled dataset was created from the Appendix Probi source text which is a list of 227 pairs of Latin terms. In particular, a pair includes a word and a common spelling mistake of the same word. After the pre-processing step each word is associated with a number indicating the identifier of a cluster. For example, the words "speculum" (correct classical form) and "speclum" (spelling variant considered as less correct) have the same number indicating two spelling variations of the same word and that they should be cluster together. The final dataset contains all the words of the Appendix Probi source associated with the correct cluster label. The dataset is then split into a training dataset (80 %) and a testing dataset (20 %).

2) Method and model selection: In this part we used the training dataset. Hierarchical clustering was used to group the spelling variants of the same word. This clustering uses different distance metrics which will be defined later in the text. We then form flat clusters from this hierarchical clustering by choosing a threshold. In particular the words in each flat cluster have a cophenetic distance smaller than

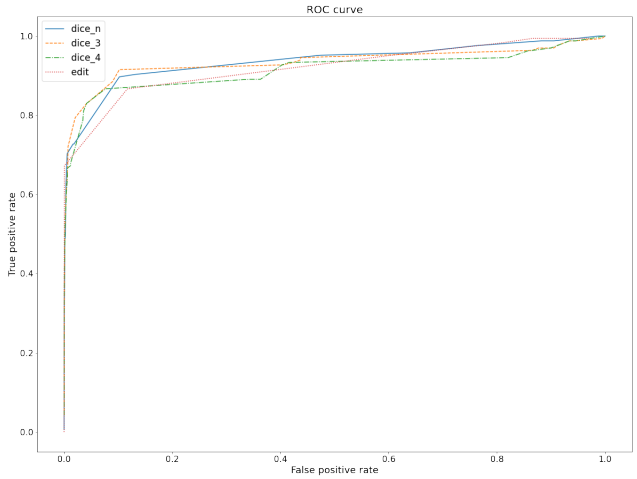

Figure 16: ROC curve for the four different models.

this threshold t. The different notions of distance used are the following.

$$Dice_n(X,Y) = 1 - \frac{2c}{x+y} \tag{1}$$

where x and y are the total number of n-gram tokens for the words X and Y and c is the number of n-gram tokens common to X and Y.

$$D_3(X,Y) = \frac{n-m+1}{m} \sum_{i=m}^{n} \frac{1}{2^{i-m}} \tag{2}$$

$$D_4(X,Y) = \frac{n-m+1}{n+1} \sum_{i=m}^{n} \frac{1}{2^{i-m}} \tag{3}$$

where m is the position of left-most character mismatch, and $n+1$ is the length of the longest string.

The Levenshtein distance between two strings is the minimal number of insertion, deletion and substitution to transform one string into another. This last distance is not very morphologically sensitive compared to the three first metrics [8].

We used a simple grid search to choose the optimal threshold that maximizes the true positive rate and minimizes the false positive rate for the four metrics. In our context the true positive rate is the proportion of pairs with both clusterings (predicted clusterings and ground truth clusterings) having the samples clustered together. The false positive rate is the proportion of pairs with the true label clustering not having the samples clustered together but the predicted clustering having the samples clustered together.

As seen in the figure 16 the model using the D_3 distance is the best. As expected the Levenshtein distance is the

worst performing model for this specific task. The following table shows the best threshold for each model with the corresponding true and false positive rate.

	$Dice_n$	$Dice_3$	$Dice_4$	Levenshtein
Threshold	0.51	4.93	1.28	2.46
False positive rate	0.10	0.10	0.08	0.12
True positive rate	0.90	0.91	0.87	0.87

We selected the D_3 distance with the threshold 4.93.

3) Result: We use the testing part of our dataset (remaining 20% of the dataset) to test the selected model. We used the D_3 metric with a threshold of 4.93. The true positive rate is 88.37% and the false positive rate is 6.06%.

REFERENCES

[1] E. Baum and W. Merrill, "Voynich2Vec: Using FastText Word Embeddings for Voynich Decipherment," Tech. Rep., 2018.

[2] *Secretum Secretorum*, 08 2021. [Online]. Available: https://en.wikipedia.org/wiki/Secretum_Secretorum

[3] R. Sravana and K. Kevin, "What We Know About The Voynich Manuscript," Tech. Rep., 2020. [Online]. Available: https://aclanthology.org/W11-1511.pdf

[4] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," Tech. Rep., 2017.

[5] "Voynich MS - Text Analysis - Transliteration of the Text," 06 2021. [Online]. Available: http://www.voynich.nu/transcr.html

[6] Meta, "List of options · fastText," 2020. [Online]. Available: https://fasttext.cc/docs/en/options.html

[7] M. Wattenberg, "How to Use t-SNE Effectively," 10 2016. [Online]. Available: https://distill.pub/2016/misread-tsne/

[8] J. Snajder and B. Basic, "String Distance-Based Stemming of the Highly Inflected Croatian Language," Tech. Rep., 2009.