Cross-Language Substitution Cipher : An approach of the Voynich Manuscript

Elie Duthoo
elie.duthoo@protonmail.com

Abstract

The Voynich Manuscript (VMS) is an illustrated hand-written document carbon-dated in the early 15th century. This paper aims at providing a statistically robust method for translating voynichese, the language used in the VMS. We will first provide a set of statistical properties that can be applied to any tokenizable language with sub-token elements, apply it to Universal Dependencies (UD)[1] dataset plus VMS (V101 transliteration[2]) to see how it compares to the 157 corpora written in 90 different languages from UD. In a second phase we will provide an algorithm to map characters from one language to characters from another language, and we will apply it to the 158 corpora we have in our possession to measure its quality. We managed to attack more than 60% of UD corpora with this method though results for VMS don't appear to be usable.

1 Introduction

The VMS is one of the rare data-rich document that remain not understood thus far. A first question we must ask ourselves before trying to translate voynichese is to know if the text contains any sense (?), (?). We can separate two situations here: in one case the text was designed to not contain any meaning, (?); in the other, the raw document was designed to hide a meaning (?). This paper won't elaborate too much on these two points for the following reasons: In a first approach, it seems implausible that this manuscript was designed to imitate a language that well in the 15th century while still respecting basic statisical properties of languages (?), though it still could be extremely elaborated which would be non-trivial to prove or disprove. On the second possibility, any text could hide any meaning, if I tell you to run when I say "Blueberry", I change the meaning of "Blueberry" to "Run". It means that, as anything could mean anything, we will never be sure of the meaning of anything, including the VMS. This is just to illustrate how solid theories proning that the VMS is hiding a message should be.
One last thing is to understand the vocabulary we used when studying the VMS. "Deciphering the VMS" isn't necessary used to mean that it was intentionally ciphered to hide a message. However, the methods used for deciphering intentionally encoded messages can still in part be applied on unintentionnaly non-understandable messages.

In this paper, we aim at providing non-refutable and measurable statistical tools for translating one unknown characters language text to a language with known characters. We will mostly provide:

- A set of properties and an analysis to compare languages of the world
- An algorithm that takes in inputs a source corpus with unknown characters and a support corpus, in two different languages, and that associate characters from the source corpus to characters from the support corpus

As we will mostly be working with alphabetical languages, we will use the word "character" instead of "sub-token elements", and "words"

[1] https://universaldependencies.org/
[2] http://www.voynich.nu/transcr.html

Figure 1: Start of the second line of f20v folio

instead of "tokens" for the sake of understanding.
The algorithm provided will be measured on multiple combinations of UD languages (?) to estimate its capabilities, and then applied on voynichese. This way we claim to provide a measurable way to try to translate voynichese. This analysis will mostly be based on existing transliterations of voynichese, we consider using the images only as a confirmation for found results.

2 Handmade statistical language properties

2.1 Preprocessing before the analysis

Our first goal was to identify which languages seemed to be the closest of voynichese. A lot of work has already been done on this subject but not with the large amount of corpora, languages and idioms that UD dataset contains. Different assumptions will need to be made to do this preliminary work. We only possess visual informations, folios, on voynichese, so we first need to go from these visual informations to a set of characters. This work has already been done but different results outcome depending on if you take the V101 or the EVA transliterations. For example, on folio f20v, second line, EVA starts with "sos ykaiin" while V101 starts with "sos 9hom". The 'm' from V101 ends up being 'iin' in EVA which will lead to different results depending on the algorithm you use on it (see Figure 1).

Now, instead of images, we have a list of characters and words. We already have made the assumption that the document was meaningful, we now make the assumption that these characters are part of an alphabet, and that when combined they form words that contain an individual meaning.
Our statistical properties will be computed on V101.
As a preprocessing, we ignore all punctuations in UD dataset. As voynichese doesn't seem to be using punctuations, we drop them to artificially make all of our corpora closer to voynichese. Our database is composed of 90+1 languages which contain known characters and words, except for one: voynichese.

2.2 Unsupervised character-based analysis

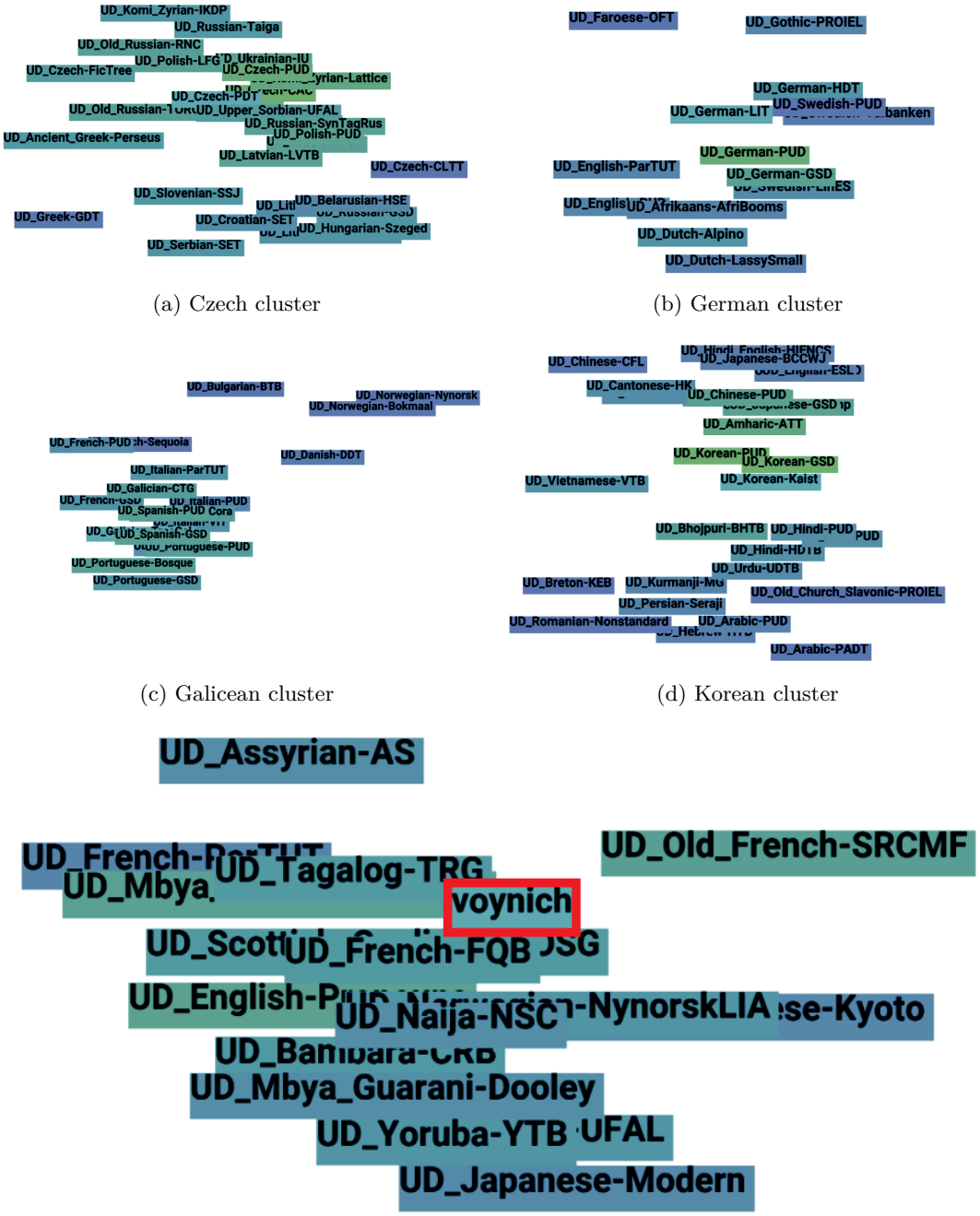
We have a lot of informations for all 90 languages in UD, but in order to compare them with voynichese, we're only able to use informations that are shared with voynichese. Only the sub-token element classification and tokenization are provided for the 91 languages.
We build a set of properties that are shared by all languages:

- Quantiles and means of words lengths
- Variance of words lengths
- Quantiles, mean and variance on proportions of unique characters per word
- Proportions of the N most frequent M-gram as a percentage of all M-gram
- Mean and variance on the proportion of unique words for a set of M words on N iterations
- Based on a word size, average size of the next and precedent words in absolute or relative value compared on current word size

These properties should avoid being too much dependent on the size of the corpus as UD corpora lengths wildly vary.

We are able to build a 68-long embedding for each language. We use the T-SNE algorithm (?) provided by Google Tensorflow Projector[3] to analyse our results. Main language groups seem to be approximately respected (see Figure ??). In our analysis, we tried giving different weights for each statistic to have a better representation, with the quality of the representation being measured with how close in ranks two corpuses from the same language are, but this approach seemed to overlearn same language closeness while degrading different but same-group language proximity, so this approach was abandoned. We can see balto-slavic languages in a cluster

[3] https://projector.tensorflow.org/



(a) Czech cluster

(b) German cluster

(c) Galician cluster

(d) Korean cluster

(e) Voynichese cluster

Figure 2: Clusters of languages

(??), germanic languages in an other (??), pure roman languages (??), asian languages (??) which seem to share some features and a last cluster containing voynichese (??).

This last cluster is quite interesting because it contains a lot of poorly used languages. In the original space, V101 VMS is close in order of proximity to English-Pronouns, MbyaGuarani, OldFrench, Scottish-Gaelic, French FQB, Tagalog, French Spoken, Bambara, Marathi, Norwegian NynorskLIA, Yoruba and English LinES. First we'll provide some insights on these corpora.

- English-Pronouns is a corpus based on artificially made redundant sentences in which only pronouns change

- Mbya Guarani is a language from South America tribes with around 15.000 native speakers
- Old French is a corpus based on 10 french texts written from 9th to 13th century
- ScottishGaelic-ARCOSG is a corpus composed of transcripts of interviews, oral narratives, news scripts, fictions etc. Scottish-Gaelic is a language that derive from Old and Middle Irish.
- French FQB is a corpus composed only of questions written in french
- Tagalog is a language spoken around the Philippines
- French Spoken is a corpus composed only of retranscripted french, with most aspect of a real conversation retranscripted (space fillers, onomatopoeia etc.)
- Bambara is a lingua franca used in Mali
- Norwegian NynorskLIA is based on transcribed spoken Norwegian dialects
- Yoruba is a pluricentric language spoken in West Africa
- English LinES is a quite standard modern english corpus

These findings support multiple hypotheses that were made in the past about voynichese. These hypotheses include:

- Voynichese cannot be trivially linked to any existing language, as the closest languages we found cover a lot of unlinked different origins
- VMS may be a transcript of an oral language, as are French Spoken, Norwegian NynorskLIA, ScottishGaelic-ARCOSG but at some extents almost all closest corpora are or could be transcripts of a spoken language
- VMS may have been artificially forged to illustrate voynichese as a language, as is English-Pronouns (and French-FQB could count as artificially forged)
- Voynichese may be a medieval european language, as is Old French
- Voynichese may be an european/latin adaptation of a foreign spoken language, as are Mbya Guarani, Tagalog, Bambara and Yoruba

After this analysis, none of these previous hypotheses seem to be refutable. A large point that would include all closest found relations is that voynichese may be a particular form of european literature (like a poem, an oral transcript, both...). Based on this extremely large possibility, our knowledge as data-scientists, and mostly on the data we have, we think that finding a character mapping from voynichese to another existing european language seems to be the best way to tackle the problem.

3 Attacking voynichese

3.1 Problem formulation

Our last hypothesis is that voynichese is a kind of european language on which whatever kind of transformation could have been applied. We have access, thanks to UD, to a large amount of different tokenized european corpora. We will describe two methodologies to compare two european languages. Most specifically, our goal is to find a character mapping between an european language with unknown characters to another european language with known characters. This way, we are able to measure the quality of our algorithm as we can measure the quality of our mapping on all the known character languages we have. As an example, if our algorithm can find the correct mapping (a=a, b=b, etc..) from anonymized english characters to french characters, despite these languages being quite far from each others, we can consider having a great algorithm. In fact, for each language, we only need to have one other language able to uncover its anonymized characters. Then we can apply this algorithm with voynichese as the anonymous language and find the best support language between all others.
However, even if we can measure how great we did on UD, we will also need a metric to measure how great we did on voynichese. Our main metric will be based on the proportion of unique words we successfully translated. A word is considered as successfully translated if, after a character mapping applied on its characters, the result can be found in the known-character language. The overall proportion of word translated was a less precise metric in our experiments.
Moreover, the final text must make some kind of senses. This metric is quite hard to formalize. If the text corresponds to the images without using the images, it seems to be a great

clue.
A final constraint is to have a computationnaly efficient algorithm. As we have more than 100 corpora with often more than 50 unique characters (uppercase included as characters are anonymized), even if testing one character combination took one second, testing all combinations would take billions of years. For two languages composed of 50 characters, there could be $50! = 3.04 * 10^{64}$ combinations, and we would ideally like to test all languages with each other which would mean $158^{157} * 50! = 4.70 * 10^{409}$ combinations to test. This impossibility is the main reason for the two next proposed algorithms:

3.2 Character embedding proximity

Our first approach was to create an embedding for each character in the source language and the target language, and randomly map the closest characters between each others as an heuristic. This approach did work at some points for same-language corpora, but quickly became inefficient as languages would differ.
Characters embeddings were built with:

- Quantiles on relative positions (in percentage of length) by words : each character has a list of positions based on all words it's in, we took some quantiles from this list
- Average position in a word
- Word average size: each word a character is in has a size, we took quantiles from this list
- Quantiles on relative positions by unique words
- Frequency

Here are the simplified steps for building this algorithm:

1. Given a source language A, and a support language B
2. Make embeddings for each characters from A and B
3. Measure distances between A and B character embeddings
4. Assign each character from A its closest non-already-assigned character from B
5. Read the corpus A and count how much word from B you found

For all languages but voynichese, you can compare the end result with the true mapping. The following results are always presented for different corpora.

4 Source=French, Support=French

For same language but different corpora, we achieve understandable results. After running the algorithm, we are able to find back 64% of words in the support corpus, and 32% of unique words. The best achievable results with the right mapping are 91% and 79% of unique words. An example of result: "Alors que la plus grande partie de la transition numbrique est sans prbcbdent auw à les la transition sereine du de le poufoir elle ne est pas". This text is slightly understandable for a french speaker and it does make some sense (tokens including punctuations were deleted).

5 Source=French, Support=Portuguese

French and portuguese are languages which are close to each other. Our results after some run is 32% of non-unique words found and 2% of unique words found. The best achievable results are 38% and 11%. An example of result: "dgE yïery e Vrytib ceoOuitgeu 8 salybkyt 2 euret 1 prygy a 5 hreoéat". The original sentence is "Até agora o Brasil conquistou 8 medalhas 2 ouros 1 prata e 5 bronzes". We can see that numbers do end up close enough in the embedding space, but the results are quite disappointing for characters. The fact that the non-unique word proportion of found word is so high reflects the need for considering unique words. Small words are quite easy to map with characters, and are extremely frequent, even if you mistake an "a" for an "o" and an "o" for an "e", you'll end up matching two letters as "o" and "e" exist as individual words in portuguese. This also means that it's probably better to have a metric based only on longer words. Making a character mapping that match the word "a" is quite easy, but making a character mapping that match all unique long names is harder and thus a better metric.

6 Source=Voynichese, Support=Old French

For voynichese, if we aim at having the maximum of non-unique word translated, it's not hard to reach up to 15-20% of non-unique

words translated. Here is an example of result "giusé s bie in avil fme Jmns öéüé". It's supposed to be understandable in Old French, but it's gibberish. The average length of the 154 found words is 2.8 characters, with only 2 found words being at a maximum of 5 characters.
If we aim at having the best score on unique word translated, we reach around 2% of unique words translated. Result:" bivst s yie in dciz qoe Sons Rtüt s yon qoers tons". Both results don't seem to be intelligible. Even if we learned how to build a better metric with this approach, the fact that it's not able to perform on different languages doesn't make it a great candidate for finding a great character mapping between two languages.
What we mainly learned:

- Don't trust non-unique word metrics
- Don't trust metrics on small words
- Compare metrics between languages. The fact that we could at most only find 11% of words from French to Portuguese doesn't make the 2% such a bad score.

All of these will be used for our next algorithm.

6.1 FEFCMA: Fast Efficient Frequency-based Character Mapping Algorithm

6.1.1 Method

As the embedding based algorithm seemed to be a dead end because of the computational complexity to test enough combinations on accurate enough embeddings, we built another algorithm based on dynamic programming. As you remember the computational complexity of testing all character combinations, this was of course an example of the worst use case where we would have to try all combinations, but would we really need to do that?
We designed an algorithm that will be referred as FEFCMA[4] that can measure the quality of a partial character mapping to efficiently map characters from one language to another. See algorithm ??, written in pythonic pseudo-code.

(*)The rank matrix is a matrix of size N*M, with N the number of words and M a maximum word length. At (x,y) in the matrix, we have:

- The occurence rank in the total corpus of the y-th character of the x-th word
- Or -1 if the character is excluded or unmapped
- Or -2 if it's an end-word padding

(∩∗) The ∩∗ is here to represent a partial intersection that count as valid any point containing a -1.
Example: We have the word AAC in language A, and the word BBC in language B. In the rank matrices mA and mB, it respectively gives [0,0,4,-2,-2] and [1,1,0,-2,-2], with 5 the maximum word length. The first iteration try a mapping from rank 0 in A to rank 0 in B, from char A to char C. In mAB we get AAC becoming CC?, so it gives us [0,0,-1,-2,-2]. If we have a word containing CC? in mB (like CCV), the line AAC from language A count as at least partially valid. The algorithm counts the proportion of at least partially valid lines above a certain word length, and keeps the best at each iteration. Then the algorithm test a mapping from rank 0 in A to rank 1 in B, etc.. After testing all possible ranks in B for rank 0 in A, it keeps the best

```
Data: Source corpus from language A,
      support corpus from language B
Result: M = Character mapping from
        A chars to B chars
init: Rank matrix (*) mA of A words;
init: Rank matrix mB of B words;
init: mAB a -1 matrix of shape mA;
init: Sort chars by frequency in A; in B;
init: M an empty mapping;
for each char rank cAr do
    let cA char with rank cAr;
    let scores a list;
    for each char rank cBr do
        let cB char with rank cBr;
        m = M;
        let m a mapping with cA = cB;
        for each cr1=cr2 in m do
            Where cr1 in mA, put cr2 in
             mAB
        S = len(mAB ∩∗ mB) / len(mAB);
        add (cB,S) to scores;
    M[cA] = argmax_cB(scores)
M;
```

Algorithm 1: FEFCMA

[4] https://github.com/Whiax/FEFCMA

mAB for rank 0 in A and continues to rank 1 in A with possible ranks in B, and so on.
An intuition on why this algorithm works is as following: we start with all characters unknown between A(source) and B(support). Then, we map the most frequent character from A to a certain number of character from B in order of frequency to see witch one is the best. When we only look at long-enough words, the proportion of partially found words creates a peak for the correct mapping. When we map 'a' to 'b', we find less partially valid words than when we map 'a' to 'e', and less than when we map 'a' to 'a', as it has a perfect correspondence. The 'a' of A will be more often placed at a position considered as valid for language B when it's placed where the 'a' of B is placed, provided A and B are close enough. Words that are present both in A and B will create a peak on the perfect correspondence, as will all other close words. This algorithm is based on similar words between languages. As all languages are a way of communication, exchanges between languages are frequent, so are matching words or derivatives.
This algorithm is vulnerable to biases between languages where char X from lang A would frequently be in place of char Y from lang B. After usage it turns out that these biases are not that frequent and having a bad result for only one character doesn't make the whole text unintelligible. After trying all required combinations for a character, we take the best mapping and preserve it for the next character from A. This algorithm can be repeated multiple times, if the algorithm did a mistake on one character, launching the algorithm again with the next character solved could help it find the right character on the second try as the peak could move on a more probable mapping, knowing we would have less partially mapped characters.
The algorithm can be used as a preprocessing before a more brute-force attack on closest characters. If our 'a' from French is in order of best score mapped to 'e', 'i', and 'a' in English, then we still can try all of these possibilities provided the combinations to test aren't too numerous. This would be applicable after the FEFCMA by saving all combinations scores. We consider this applicable for a maximum distance of 3 for each character, knowing the bruteforce method never try surjections.

6.1.2 Results

We apply FEFCMA on the 10 first characters of the source language to the support language. We apply it twice by enabling or disabling multiple-source for one target character (a=>a, b=>a, c=>e..), aka surjections.
The raw results are provided for all languages of UD + voynichese (except for some ill-formed UD corpora). The results are presented as a matrix where the y-th row, x-th column represents how well we attacked the y-th corpus/language as a source with it's x-th closest corpus/language as a support. This analysis is highly pluri-dimensional as the following characteristics shall be reflected:

- What is the distance to the perfect measurable mapping? Red to orange to yellow to green background
- Are the two languages using the same set of characters? Blank background if no (not enough characters in common), as the quality of the character mapping can't be measured relatively to a perfect mapping.
- Are the two languages the same (or almost, like old variants)? Underline if yes.
- What is the proportion of perfectly found words? This will be the number in the cell

Results took one week on a 4cores*3.8GHz CPU (on a laptop) to compute. Tables can be found in appendix.

Analysis: We consider an attack as a perfect success if all first 10 most frequent anonymized characters from the source corpus were mapped to their true value in the support corpus. Based on our results, 30% of source corpora were perfectly attacked with another language. 50% were partially attacked with 7 or more perfectly found characters, and 57% could have been understood after a transcript with 5 or more perfectly found characters and 8 or more partially found characters (characters that are close enough to the true mapping to be found by bruteforce on closest characters).
When taking into considerations the situation where the source language can be the same as the support language, we reach up to 54% of perfectly attacked source corpora (all characters were anonymized and then rightfully

mapped back to what they originally were by FEFCMA). Up to 64% of source corpora could be understood after the mapping as they were partially attacked (7 or more perfectly found characters) and 67% of source corpora could probably be found by a bruteforce attack.
Tables in appendix show that recent european languages are the easiest ones to attack which is not surprising as they provide the more data and were all close to other european languages. Results for old european languages are of a great interest. Between AncientGreek-Perseus and Greek-GDT, we found 7 characters out of 10 which seems to be enough to understand and refine results. Old French was also successfully attacked.
For languages that were close to voynichese, we got the following results:

- English-Pronouns is the only english corpus with bad results
- We failed to have great results on Mbya Guarani, Tagalog, Yoruba or Bambara, all these languages are based on alphabets.
- Old French was successfully attacked with but not only by other modern french corpora, so was FrenchFQB
- Scottish-Gaelic was perfectly attacked with Irish as a support
- We got 7 great characters on Norwegian-NynorskLIA with its closest corpora being in French or English, but the corpus can probably be perfectly attacked with another Norwegian corpus

To summarize these results, if voynichese is close to an old european language, it seems very plausible that we could be able to uncipher it with this algorithm. As we are competent in french language, we tried to agressively optimize FEFCMA parameters to attack it with Old French as a support. Despite reaching up to 15% of unique v101 words giving an existing old french word only by a character mapping, the resulting text truly lacked of meaning (cf appendix). These high results show how important a careful analysis on the produced transliteration is, percentages of translated words are not robust enough metrics to prove a VMS translation correct. We propose some tracks to continue our work:

- We first recommend, for anyone trying quantitative analysis, to have a scientifically measurable approach by testing it on UD or other corpora. If one approach theoretically works on VMS/voynichese but fails on all other known languages, we recommend going into more depth.
- We recommend, for everyone, to continue seeking for languages which could be close to voynichese. This could be done by an enhanced visual analysis on VMS, on characters, and on other metrics like the one we can find in the World Atlas of Language Structures[5]. One can also use machine learning to train algorithm to identify which language does an anonymized corpus belong to, measuring its quality with a Leave-One-Out strategy and using it on VMS.
- If one consider that voynichese is or derive from an old european language, we recommend to use FEFCMA[6], optimizing it with hyperparameters, reapplying it multiple times and to test multiple VMS transliterations with multiple european languages. This approach seems to be working for european languages.
- The possibility that VMS was built in the same way as Tagalog, Mbya Guarani, Yoruba, Bambara, or other languages of this type could make us ask ourselves how we would translate these languages if we only had a 30.000 words corpus available. This could show how important qualitative analysis could be furthermore required for dealing with the VMS.
- We wouldn't deter seeking clues that would indicate that VMS was an artificially meaningless forged document, though these efforts would probably be better directed by following previous recommendations on having a robust analysis with multiple metrics based on other corpora.

We can't pursue this analysis in more depth because of our lack of precise knowledge in all the languages involved in this work. For example we can't pursue the analysis of a character mapping between V101 and Scottish-Gaelic because we wouldn't be able to estimate the quality of the results.
To be furthermore validated, the algorithm

[5] https://wals.info/
[6] https://github.com/Whiax/FEFCMA

should also be evaluated between languages that don't share characters but where a character mapping does exist. This is the case between Ancient-Greek and Latin but our algorithm didn't seem to find the original mapping (α ->a, β ->b, λ ->l...) when we tried to use it with Ancient-Greek as a source and Latin as a suport.

7 Conclusion

It doesn't seem like our algorithm was able to attack voynichese, though we did continue old lines of research. Research on the VMS seems to still need to go into more depth. Despite our results based on massive data analysis, we don't think that quantitative analysis is a dead end, however we probably still need big chunks of qualitative analysis to deepen the focus of quantitative works.

Appendix

A FEFCMA scores

Afrikaans-AfriBooms	1.5	1.7	0.0	0.0	0.1	0.1	0.1	0.0	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.2	0.0	0.1	0.1
Amharic-ATT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AncientGreek-PROIEL	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AncientGreek-Perseu	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Arabic-PADT	0.3	0.1	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1
Arabic-PUD	2.6	0.2	0.0	0.0	0.2	0.1	0.0	0.0	0.0	0.1	0.2	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.2
Armenian-ArmTDP	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0
Assyrian-AS	0.0	0.0	0.0	0.0	0.5	0.0	0.0	0.9	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.5	0.0	0.0	0.9	0.5
Bambara-CRB	0.0	0.0	0.9	0.0	0.0	0.7	0.0	0.0	0.0	0.0	0.1	0.1	0.1	0.1	0.2	0.0	0.1	0.0	0.1	0.4
Basque-BDT	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.1	0.1	0.0
Belarusian-HSE	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0
Bhojpuri-BHTB	0.3	0.1	0.0	0.1	0.2	0.0	0.2	0.2	0.6	0.0	0.1	0.0	0.4	0.1	0.2	0.0	0.1	0.0	0.0	0.0
Breton-KEB	0.0	0.1	0.0	0.1	0.0	0.1	0.3	0.2	0.2	0.3	0.3	0.3	0.0	0.0	0.2	0.2	0.0	0.0	0.1	0.2
Bulgarian-BTB	0.0	0.0	0.1	0.2	0.1	0.1	0.1	0.2	0.0	0.1	0.2	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.1	0.1
Buryat-BDT	0.2	0.0	0.2	0.2	0.1	0.0	0.4	0.0	0.2	0.1	0.2	0.0	0.0	0.0	0.2	0.3	0.2	0.0	0.0	0.1
Cantonese-HK	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Catalan-AnCora	0.6	0.6	0.4	0.1	0.4	0.6	0.1	0.1	0.1	0.8	0.1	0.8	0.3	0.2	0.2	0.2	0.1	0.8	0.3	0.1
Chinese-CFL	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		0.0	0.0	0.0	0.0	0.0	
Chinese-GSD	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Chinese-GSDSimp	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Chinese-HK	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Chinese-PUD	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Coptic-Scriptorium	0.1	0.2	0.4	0.1	0.1	0.2	0.0	0.1	0.2	0.0	0.1	0.2	0.0	0.1	0.0	0.1	0.0	0.2	0.2	0.2
Croatian-SET	0.0	0.4	0.0	0.8	0.0	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Czech-CAC	0.5	0.0	0.2	0.2	0.0	0.0	0.0	0.1	0.1	0.1	0.1	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0
Czech-CLTT	0.0	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.0	0.0	0.0	0.1	0.0	0.0
Czech-FicTree	0.0	0.0	0.1	0.5	0.0	0.2	0.5	0.0	0.0	0.0	0.0	0.1	0.2	0.0	0.0	0.0	0.0	0.0	0.1	0.1
Czech-PDT	0.5	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.2	0.0	0.1	0.0	0.1	0.0	0.1	0.4	0.0
Czech-PUD	0.3	0.6	0.6	0.0	0.0	0.1	0.2	0.0	0.0	0.0	0.3	0.2	0.2	0.0	0.0	0.0	0.1	0.1	0.0	0.2
Danish-DDT	1.6	1.0	0.1	0.0	0.1	0.2	0.0	0.3	0.0	0.0	0.1	0.1	0.1	0.3	0.1	0.0	0.0	0.1	0.1	0.0
Dutch-Alpino	1.7	0.2	0.1	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.2	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0
Dutch-LassySmall	2.2	0.2	0.1	0.2	0.0	0.0	0.1	0.0	0.0	0.1	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0
English-EWT	1.9	0.0	1.1	1.7	0.1	1.2	0.0	0.1	0.0	0.0	0.1	0.2	0.2	0.2	0.0	0.0	0.4	0.0	0.0	0.0
English-GUM	1.4	2.8	0.0	1.5	2.2	0.0	0.1	0.1	0.2	0.1	0.1	0.4	0.0	0.3	0.0	0.2	0.5	0.0	0.0	0.2
English-LinES	3.4	3.1	1.7	0.0	0.0	0.0	0.3	1.8	0.0	0.2	0.1	0.0	0.0	0.5	0.0	0.1	0.0	0.2	0.2	0.0
English-PUD	3.3	2.6	0.0	3.7	2.9	0.1	0.2	0.1	0.2	0.2	0.5	0.0	0.0	0.6	0.3	0.3	0.2	0.0	0.3	0.1
English-ParTUT	2.0	0.0	2.7	0.1	0.0	3.2	0.4	0.0	0.0	0.6	0.0	0.1	0.2	0.1	0.2	0.2	2.4	0.1	0.3	0.1
English-Pronouns	0.0	3.8	0.0	0.0	1.9	0.0	0.0	1.9	0.0	3.8	0.0	0.0	1.9	0.0	0.0	0.0	1.9	0.0	0.0	0.0
Erzya-JR	0.0	0.1	0.0	0.1	0.2	0.2	0.0	0.0	0.0	0.1	0.2	0.0	0.0	0.0	0.1	0.1	0.2	0.1	0.2	0.0
Estonian-EDT	0.1	0.1	0.2	2.7	0.0	0.4	0.0	0.1	0.4	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.4
Estonian-EWT	0.0	0.1	0.0	6.4	0.2	0.0	0.3	0.0	0.1	0.4	0.0	0.0	0.1	0.0	0.3	0.0	0.0	0.1	0.1	0.4
Faroese-OFT	0.0	0.0	0.0	0.0	0.1	0.0	0.1	0.1	0.3	0.0	0.0	0.0	0.1	0.1	0.0	0.1	0.0	0.0	0.0	0.0
Finnish-FTB	4.9	1.6	0.0	0.0	0.2	0.3	0.1	0.0	0.0	0.0	0.1	0.0	0.1	0.1	0.0	0.0	0.2	0.0	0.0	0.0
Finnish-PUD	6.5	5.8	0.0	0.0	0.0	0.1	0.4	0.0	0.1	0.1	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.1	0.2	0.1
Finnish-TDT	2.4	6.7	0.0	0.0	0.0	0.0	0.3	0.0	0.1	0.1	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.1	0.0	0.1
French-FQB	0.1	2.2	2.5	0.2	2.9	4.0	0.0	0.2	0.1	0.0	0.3	0.1	0.0	2.2	0.0	0.1	0.0	0.2	0.9	0.1
French-GSD	1.4	0.0	1.8	0.0	0.1	0.0	0.0	0.0	0.1	0.0	0.0	0.1	0.3	0.0	0.2	0.0	1.2	0.0	0.5	0.1
French-PUD	4.5	3.4	2.5	0.0	0.1	0.1	0.0	0.2	0.0	0.7	0.6	0.1	0.2	0.1	0.3	0.8	0.7	0.1	0.1	0.0
French-ParTUT	3.6	5.5	4.6	0.0	0.1	2.2	0.9	0.0	0.1	0.0	0.0	0.0	0.7	0.0	0.0	1.0	0.2	0.1	0.1	0.1
French-Sequoia	2.2	3.8	0.1	0.1	0.1	0.1	0.2	0.1	0.0	0.1	0.0	2.0	0.1	0.2	0.0	0.6	0.1	0.1	0.0	0.0
French-Spoken	1.5	0.1	0.7	2.1	0.0	4.2	0.1	0.0	0.0	0.0	3.3	5.4	0.1	0.3	0.0	0.0	0.8	0.4	0.2	0.5
Galician-CTG	2.2	1.3	0.2	0.0	0.1	0.1	0.4	3.2	0.5	3.4	0.2	1.4	0.1	2.8	0.1	2.6	0.1	0.1	0.1	0.0
Galician-TreeGal	4.3	2.3	2.3	4.5	0.4	4.3	0.9	6.4	0.4	4.1	0.1	0.2	0.2	0.1	0.2	0.2	0.6	0.3	0.2	0.3
German-GSD	0.6	1.1	0.0	0.6	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
German-HDT	1.3	0.7	0.0	0.0	0.1	0.0	0.0	0.2	0.8	0.1	0.0	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0
German-LIT	1.2	1.9	0.3	0.2	0.1	0.1	0.2	0.1	0.1	2.1	0.0	0.0	0.0	0.0	0.0	0.3	0.0	0.0	0.0	0.1
German-PUD	1.9	1.2	1.9	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0
Gothic-PROIEL	0.1	0.1	0.1	0.0	0.1	0.2	0.1	0.0	0.0	0.0	0.1	0.0	0.1	0.1	0.1	0.2	0.0	0.0	0.0	0.1
Greek-GDT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Hebrew-HTB	0.3	0.1	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.0	0.4	0.1	0.3	0.0	0.1	0.0	0.0	0.3	0.0
Hindi-HDTB	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0
Hindi-PUD	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.0
Hungarian-Szeged	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Indonesian-GSD	1.2	0.0	0.1	0.0	0.1	0.0	0.0	0.2	0.0	0.1	0.1	0.1	0.1	0.0	0.0	0.0	0.1	0.0	0.1	0.0
Indonesian-PUD	5.4	0.2	0.0	0.2	0.1	0.2	0.0	0.3	0.2	0.0	0.0	0.3	0.2	0.2	0.0	0.2	0.2	0.1	0.2	0.0
Irish-IDT	0.2	0.1	0.0	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0

Table 1: Scoring charts - green means 10 characters out of 10 were found, yellow is for 8 or more, orange 5 or more, red 4 or less, blank means we can't measure the mapping automatically as the two languages most frequent characters sets don't intersect enough (less than 70% of the 20 most frequent characters in common)

Italian-ISDT	2.9	0.1	4.2	0.0	1.9	0.3	0.5	0.1	0.1	0.1	0.0	0.1	0.1	0.0	0.2	0.1	0.2	0.1	1.6	2.9
Italian-PUD	5.6	4.7	5.9	0.3	0.5	0.2	0.5	0.4	0.2	0.6	0.0	0.2	0.1	0.3	0.1	0.4	0.0	0.5	0.2	0.3
Italian-ParTUT	6.6	5.7	3.3	0.2	0.2	0.2	0.3	0.3	0.4	4.2	0.3	0.4	2.4	0.2	0.0	0.0	0.0	0.6	0.1	0.4
Italian-PoSTWITA	3.1	3.1	0.1	5.1	4.9	2.3	0.2	0.3	0.1	0.0	0.3	0.2	0.0	0.3	0.0	0.0	0.0	0.3	0.1	0.0
Italian-TWITTIRO	8.1	0.4	0.0	0.8	3.5	0.7	0.4	0.0	0.2	0.8	0.0	0.6	0.0	0.9	0.5	0.4	0.3	0.5	0.5	0.7
Italian-VIT	1.8	2.6	4.2	0.1	0.1	0.3	0.2	0.2	0.0	0.2	0.0	0.4	0.1	0.3	0.1	0.4	0.4	0.0	0.2	0.4
Japanese-GSD	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Japanese-Modern	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.9	2.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.9	2.9
Japanese-PUD	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Karelian-KKPP	0.4	0.4	0.3	0.1	0.0	0.1	0.5	0.0	0.2	0.1	0.3	0.2	0.0	0.4	0.0	0.0	0.0	0.1	0.1	0.0
Kazakh-KTB	0.1	0.3	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.1	0.1	0.0	0.1
KomiPermyak-UH	0.6	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.0	0.0	0.0	0.6
KomiZyrian-IKDP	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
KomiZyrian-Lattice	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Korean-GSD	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Korean-Kaist	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Korean-PUD	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Kurmanji-MG	0.3	0.0	0.6	0.0	1.0	0.0	0.6	0.8	0.7	0.0	0.2	0.8	0.0	0.2	0.3	1.4	0.0	0.1	0.1	0.0
Latin-ITTB	0.0	0.0	0.0	0.0	2.6	0.1	0.1	0.1	0.0	0.1	0.0	0.1	0.0	0.6	0.2	0.1	0.4	0.1	0.1	0.2
Latin-PROIEL	0.3	0.1	3.0	3.2	0.0	0.0	0.2	0.2	0.1	0.1	0.0	0.3	0.0	0.2	0.0	0.0	0.1	0.2	0.1	0.1
Latin-Perseus	0.2	0.1	0.0	0.1	0.3	0.3	0.1	0.0	0.0	0.1	0.1	0.1	4.0	0.0	0.0	0.2	0.1	0.0	0.5	0.0
Latvian-LVTB	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0
Lithuanian-ALKSNIS	0.5	0.0	0.2	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Lithuanian-HSE	4.0	0.0	0.6	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.3	0.2	0.0	0.0	0.0	0.0	0.0	0.0
Livvi-KKPP	0.1	0.0	0.0	0.1	0.3	0.3	0.0	0.0	0.6	0.1	0.0	0.7	0.1	0.0	0.0	0.0	0.0	0.1	0.1	0.0
Maltese-MUDT	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.2	0.0	0.1	0.1	0.4	0.3	0.0
Marathi-UFAL	0.0	0.2	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.1	0.0	0.1	0.0	0.3	0.1	0.1	0.2	0.0	0.0	0.1
MbyaGuarani-Thomas	0.4	0.9	0.9	0.0	0.4	0.0	0.0	1.3	0.9	0.9	0.9	0.4	1.3	1.3	0.0	0.0	2.6	0.9	1.7	1.7
Moksha-JR	0.0	0.0	0.7	0.7	0.7	0.0	0.0	1.0	0.0	0.7	0.0	0.0	0.3	0.0	0.0	1.3	0.0	0.3	0.3	0.3
Naija-NSC	0.1	1.3	0.3	0.0	0.3	0.0	4.5	1.0	0.0	1.2	0.0	0.6	0.0	0.0	0.3	6.1	0.6	0.1	0.6	0.1
NorthSami-Giella	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.2	0.2	0.0	0.0	0.1	0.1	0.1	0.0	0.0	0.0	0.0	0.0
Norwegian-Bokmaal	1.4	1.8	0.1	0.1	0.1	0.0	0.2	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Norwegian-Nynorsk	2.5	0.8	0.0	0.1	0.1	0.1	0.0	0.0	0.1	0.1	0.0	0.1	0.1	0.1	0.0	0.0	0.1	0.1	0.0	0.5
Norwegian-NynorskLIA	0.0	0.2	0.0	0.1	0.0	0.8	0.0	0.2	0.3	0.0	0.0	0.0	0.1	0.1	0.3	0.4	0.0	0.1	1.3	0.4
OldChurchSlavonic-	0.0	0.0	0.0	0.1	0.0	0.1	0.0	0.4	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.3	0.0	0.0
OldFrench-SRCMF	0.2	0.2	0.2	0.1	0.1	0.2	0.0	0.5	0.1	0.2	0.1	0.3	0.1	0.1	0.0	0.1	0.1	0.1	0.2	0.2
OldRussian-RNC	0.0	0.2	0.0	0.0	0.1	0.1	0.6	0.0	0.1	0.4	0.2	0.0	0.1	0.0	0.6	0.7	0.0	0.1	0.0	0.0
OldRussian-TOROT	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.1	0.0	0.0	0.0	0.0	0.1	0.2	0.0	0.0
Persian-Seraji	0.0	0.0	0.3	0.2	0.3	0.0	0.1	0.2	0.0	0.4	0.2	0.1	0.0	0.0	0.2	0.1	0.0	0.1	0.1	0.1
Polish-LFG	0.0	0.0	0.4	0.0	0.0	0.0	0.0	0.0	0.1	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Polish-PDB	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.4	0.1	0.0	0.0	0.0	0.0	0.0	0.0
Polish-PUD	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.0
Portuguese-Bosque	3.7	1.8	1.1	1.7	0.2	1.8	0.7	0.1	0.2	0.1	0.1	2.0	0.0	0.3	0.1	0.1	0.1	0.0	0.1	0.0
Portuguese-GSD	3.8	1.8	1.8	1.1	0.2	0.1	1.8	0.0	0.7	0.0	0.2	0.1	2.0	0.2	0.1	0.3	0.0	0.1	0.0	0.0
Portuguese-PUD	2.6	0.3	5.7	2.0	5.7	2.5	0.9	0.3	0.0	3.4	2.2	0.3	0.4	0.4	0.3	0.0	0.2	0.1	0.0	0.0
Romanian-Nonstandard	0.0	0.0	0.2	0.0	0.2	0.1	0.0	0.1	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.2	0.1	0.0	0.1	0.0
Romanian-RRT	0.0	0.1	0.1	0.2	0.0	0.1	0.1	0.0	0.0	0.0	0.1	0.0	0.2	0.0	0.0	0.0	0.0	0.1	0.6	0.0
Romanian-SiMoNERo	0.0	2.9	0.1	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.2	0.4	0.0	0.0	0.0	0.0
Russian-GSD	0.1	0.4	0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.1	0.4	0.1	0.0	0.0	0.0	0.0	0.0	0.1
Russian-PUD	1.3	0.1	0.0	1.1	0.0	0.0	0.0	0.1	0.2	0.1	0.2	0.0	0.0	0.0	0.0	0.2	0.1	0.0	0.8	0.0
Russian-SynTagRus	0.5	0.2	0.1	0.1	0.0	0.1	0.9	0.7	0.1	0.0	0.0	0.0	0.0	0.0	0.1	0.2	0.0	0.0	0.1	0.1
Russian-Taiga	0.3	1.1	0.4	0.0	0.5	0.1	0.7	0.0	0.0	0.0	0.1	0.0	0.0	0.1	0.1	0.1	0.2	0.0	0.1	0.2
Sanskrit-UFAL	0.0	0.1	0.0	0.4	0.4	0.1	0.0	0.2	0.0	0.0	0.0	0.2	0.3	0.0	0.2	0.1	0.0	0.0	0.0	0.2
ScottishGaelic-ARCO	0.0	0.0	0.0	0.2	1.5	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.2	0.1	0.2	0.0	0.0	0.1	0.0	0.0
Serbian-SET	0.3	0.9	0.1	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
SkoltSami-Giellagas	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Slovak-SNK	0.1	0.2	0.2	0.0	0.0	0.0	0.3	0.0	0.0	0.2	0.1	0.0	0.1	0.2	0.1	0.1	0.0	0.2	0.0	0.0
Slovenian-SSJ	0.3	0.4	0.0	0.1	0.1	0.0	0.1	0.0	0.1	0.0	0.1	0.2	0.0	0.0	0.2	0.0	0.0	0.0	0.2	0.0
Slovenian-SST	1.4	0.2	1.1	0.0	0.4	0.5	0.0	0.4	0.2	0.0	0.6	0.0	0.1	0.0	0.1	0.0	0.5	0.6	0.2	0.0
Spanish-AnCora	1.3	3.2	0.2	0.8	0.6	0.2	0.3	0.0	0.1	1.3	0.0	0.1	0.1	1.5	1.3	0.1	0.0	0.3	0.1	0.2
Spanish-GSD	3.2	0.2	1.2	0.6	0.8	1.3	0.3	0.3	1.4	0.1	0.0	0.1	0.3	0.1	0.1	1.6	0.2	0.0	0.0	0.0
Spanish-PUD	4.2	4.0	1.3	0.2	0.4	0.2	0.1	1.7	0.0	1.7	0.0	0.0	0.2	2.0	1.6	0.6	0.5	0.1	0.2	0.2
Swedish-LinES	1.7	0.3	0.0	0.0	1.0	0.2	0.1	0.7	0.1	0.0	0.2	1.2	0.1	0.1	0.1	0.1	0.1	0.0	0.1	0.0
Swedish-PUD	2.8	1.3	0.2	2.9	1.3	0.0	0.1	0.1	0.0	0.2	0.9	0.2	0.1	0.1	0.2	0.2	0.3	0.1	0.3	0.0
Swedish-Talbanken	1.6	1.1	0.0	0.7	0.0	0.1	0.8	0.1	0.1	0.1	0.2	0.1	0.0	0.5	0.1	0.0	0.0	0.0	0.0	0.0
SwedishSignLanguag	0.4	0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.9	0.0	1.3	1.3	0.0	0.0	0.9	0.0	0.0	0.4	0.4
SwissGerman-UZH	0.4	1.2	0.0	0.0	0.4	1.6	0.2	0.2	0.2	0.2	0.2	0.2	0.4	0.0	0.0	0.2	0.2	0.2	0.0	0.4
Tagalog-TRG	0.0	0.0	2.2	1.1	1.1	1.1	0.0	0.0	0.0	2.2	0.0	0.0	0.0	2.2	3.3	1.1	1.1	1.1	0.0	0.0
Tamil-TTB	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Telugu-MTG	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.0	0.0	0.0	0.3	0.2	0.0	0.3	0.0	0.1	0.2	0.0
Thai-PUD	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.2	0.1	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0
Turkish-GB	0.2	2.1	0.0	0.0	0.1	0.2	0.1	0.0	0.0	0.1	0.0	0.0	0.2	0.0	0.0	0.0	0.1	0.0	0.1	0.1
Turkish-IMST	0.6	0.1	0.1	0.0	0.1	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.7	0.0	0.0	0.0
Turkish-PUD	0.1	0.2	0.0	2.2	0.2	0.0	0.9	0.1	0.0	0.2	0.0	0.1	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0
Ukrainian-IU	0.2	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0
UpperSorbian-UFAL	0.1	0.0	0.2	0.2	0.1	0.1	0.0	0.0	0.0	0.0	0.1	0.1	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.1
Urdu-UDTB	0.2	0.1	0.0	0.1	0.8	0.1	0.0	0.2	0.1	0.4	0.3	0.3	0.3	0.1	0.0	0.1	0.3	0.3	0.0	0.2
Uyghur-UDT	0.1	0.1	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0
Vietnamese-VTB	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Warlpiri-UFAL	0.0	0.8	0.0	0.0	5.1	3.4	0.8	0.8	3.4	0.8	0.0	2.6	0.8	3.4	2.6	3.4	1.7	0.0	1.7	2.6
Welsh-CCG	0.0	0.0	0.1	0.1	0.2	0.2	0.0	0.0	0.0	0.1	0.1	0.0	0.3	0.0	0.0	0.1	0.0	0.2	0.1	0.0
Wolof-WTB	0.0	0.1	0.3	0.2	0.0	0.1	0.1	0.0	0.1	0.0	0.0	0.0	0.3	0.1	0.0	0.0	0.3	0.1	0.2	0.7
Yoruba-YTB	0.4	0.0	0.4	0.4	0.0	0.0	0.0	0.0	0.4	0.0	0.4	0.0	0.0	0.0	0.4	0.0	0.0	0.0	0.0	0.4
voynich (v101)	0.1	1.5	0.5	0.1	0.1	0.2	0.2	0.3	0.1	0.1	0.1	0.7	0.0	0.7	0.3	0.0	0.4	1.2	0.0	0.2

Table 2: Scoring charts - green means 10 characters out of 10 were found, yellow is for 8 or more, orange 5 or more, red 4 or less, blank means we can't measure the mapping automatically as the two languages most frequent characters sets don't intersect enough (less than 70% of the 20 most frequent characters in common)

B VMS attacked with Old French

errr ere* erre* errer* trere* err_ rires _ es er_ r e_ s ez _ re perreres errerre erresre erresirre ieres es* eres* ires* r es* es* irere_ e re_ re eres* irres esrres _ sres es* eres* e_ es erre* emes_ errrez esres errrese irere e_ ire errrre eses erreses eres* ertrre ierrre errr es* es* es* irresez trerese _ rerre errese rese ereses* erre* erirre errer* eses eres* e_ s imese errere e_ irres erres estre* esrese _ irre errer* eses resr esies ereses* eres* ieres eres* ere* tere* erre* e_ irrese trese r es* errre er_ se_ _ rre e_ trerre e_ e_ ese ie_ ere ere_ irre ereses* irrre errere efe errese errese e_ r er_ te err errese eres* eser erere errer* _ rrer e_ rrere er_ s erese errrer erre_ e irerme errre eres* es* eres* re mes* erres es* e_ er es* er_ rre e_ eses estre* imes es* erese eres* eese e_ ires_ e emes irerrre eres*

Figure 3: FEFCMA results | Source=v101 | Support=OldFrench: star indicates the word exists in Old French, underscore indicates the letter wasn't mapped

As you can see, the results are gibberish. Without limitations, the algorithm attributes the letter 'r' and 'e' quite a few times. Here is the mapping: o=>e | 9=>e | a=>e | c=>r | 1=>i | e=>s | 8=>r | h=>r | y=>s | k=>r | 4=>p | m=>s | 2=>t | C=>r | 7=>r | s=>r | n=>z | p=>z | g=>m | K=>f | H=>_ | A=>_ | j=>_ | 3=>_ | z=>_ | 5=>_ | (=>_ | d=>_ | f=>_ | i=>_ | u=>_ | *=>_ | M=>_ | Z=>_ | %=>_ | N=>_ | J=>_ | 6=>_ | +=>_ | G=>_ | ?=>_ | W=>_ | x=>_ | I=>_ | E=>_ | Y=>_ | !=>_ | t=>_ | S=>_ | F=>_ ' Results don't look better with more limitations.

References

Andronik Arutyunov, Leonid Borisov, Sergey Fedorov, Anastasiya Ivchenko, Elizabeth Kirina-Lilinskaya, Yurii Orlov, Konstantin Osminin, Sergey Shilin, and Dmitriy Zeniuk. 2016. Statistical properties of european languages and voynich manuscript analysis.

Henrique F. de Arruda, Vanessa Q. Marinho, Luciano da F. Costa, and Diego R. Amancio. 2019. Paragraph-based representation of texts: A complex networks approach. Information Processing Management 56(3):479–494. https://doi.org/10.1016/j.ipm.2018.12.008.

Sravana Reddy and Kevin Knight. 2011. What we know about the voynich manuscript. In Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities. Association for Computational Linguistics, Portland, OR, USA, pages 78–86. https://www.aclweb.org/anthology/W11-1511.

Torsten Timm. 2014. How the voynich manuscript was created.

Alexander Ulyanenkov. 2016. Voynich manuscript or book of dunstan coding and decoding methods.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. Journal of Machine Learning Research 9:2579–2605.

Daniel Zeman, Joakim Nivre, Mitchell Abrams, Noëmi Aepli, Željko Agić, Lars Ahrenberg, Gabrielė Aleksandravičiūtė, Lene Antonsen, Katya Aplonova, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, Victoria Basmov, Colin Batchelor, John Bauer, Sandra Bellato, Kepa Bengoetxea, Yevgeni Berzak, Irshad Ahmad Bhat, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Agnė Bielinskienė, Rogier Blokland, Victoria Bobicev, Loïc Boizou, Emanuel Borges Völker, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Kristina Brokaitė, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Tatiana Cavalcanti, Gülşen Cebiroğlu Eryiğit, Flavio Massimiliano Cecchini, Giuseppe G. A. Celano, Slavomír Čéplö, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Jayeol Chun, Alessandra T. Cignarella, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Elvis de Souza, Arantza Diaz de Ilarraza, Carly Dickerson, Bamba Dione, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Hanne Eckhoff, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Olga Erina, Tomaž Erjavec, Aline Etienne, Wograine Evelyn, Richárd Farkas, Hector Fernandez Alcalde,

Jennifer Foster, Cláudia Freitas, Kazunori Fujita, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta González Saavedra, Bernadeta Griciūtė, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Nizar Habash, Jan Hajič, Jan Hajič jr., Mika Hämäläinen, Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Johannes Heinecke, Felix Hennig, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Takumi Ikeda, Radu Ion, Elena Irimia, Ọlájídé Ishola, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Markus Juutinen, Hüner Kaşıkara, Andre Kaasen, Nadezhda Kabaeva, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Tolga Kayadelen, Jessica Kenney, Václava Kettnerová, Jesse Kirchner, Elena Klementieva, Arne Köhn, Kamil Kopacewicz, Natalia Kotsyba, Jolanta Kovalevskaitė, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Lucia Lam, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phương Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Maria Liovina, Yuan Li, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Sarah McGuinness, Gustavo Mendonça, Niko Miekka, Margarita Misirpashayeva, Anna Missilä, Cătălin Mititelu, Maria Mitrofan, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Keiko Sophie Mori, Tomohiko Morioka, Shinsuke Mori, Shigeki Moro, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Robert Munro, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lương Nguyễn Thị, Huyền Nguyễn Thị Minh, Yoshihiro Nikaido, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Atul Kr. Ojha, Adédayọ̀ Olúòkun, Mai Omura, Petya Osenova, Robert Östling, Lilja Øvrelid, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Guilherme Paulino-Passos, Angelika Peljak-Łapińska, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Daria Petrova, Slav Petrov, Jason Phelan, Jussi Piitulainen, Tommi A Pirinen, Emily Pitler, Barbara Plank, Thierry Poibeau, Larisa Ponomareva, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Peng Qi, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Ivan Riabov, Michael Rießler, Erika Rimkutė, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Valentin Roșca, Olga Rudina, Jack Rueter, Shoval Sadde, Benoît Sagot, Shadi Saleh, Alessio Salomoni, Tanja . 2019. Universal dependencies 2.5. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University. http://hdl.handle.net/11234/1-3105.