Unsupervised Analysis of the Voynich Manuscript

Peter Bloem, 0168491

3 Juli 2006

abstract

The aim of this project is to research the possibilities of applying unsupervised learning techniques for natural language and other sequential data to undeciphered texts and manuscripts. The undeciphered text used is the Voynich Manuscript, a mysterious book from the 15th or 16th century that is written in an unknown script. Some methods that could be applied to manuscripts such as these will be discussed. Furthermore, the results of applying some of these techniques to the text of the manuscript will be discussed.

Introduction

The field of computational natural language learning can be roughly divided into two areas; supervised learning and unsupervised learning. The former requires some kind of meta-data to be supplied with the natural language data that it learns from, which it will then learn to produce on its own for unseen natural language data. For instance, from a large corpus of words that have been tagged with their linguistic class (noun, verb, etc), a POS tagging algorithm can learn to tag new text in the same way. Similarly from a large set of sentences with parse trees based on some grammar, an algorithm can learn to construct parse trees or even grammars for sentences that do not occur in the dataset. The main drawback to this approach, of course, is that the original data to train the algorithm will have to be produced by hand. Tagging all word of a corpus, or parsing all sentences by hand is an arduous task that is not always feasible (especially for small languages). A second, more idealistic concern, is that this method of language learning falls somewhat short of the original goals of Artificial Intelligence. If humans are able to learn language by example, that means that most of the structures present in language can be retrieved from it without any meta data and algorithms should be able to detect it. Methods that find structure in language without any other data than the raw language are known as unsupervised methods. An added benefit of unsupervised systems is that they can often detect structure in other sequential data as well. Some of these systems have been successfully applied to music, DNA and protein sequences.

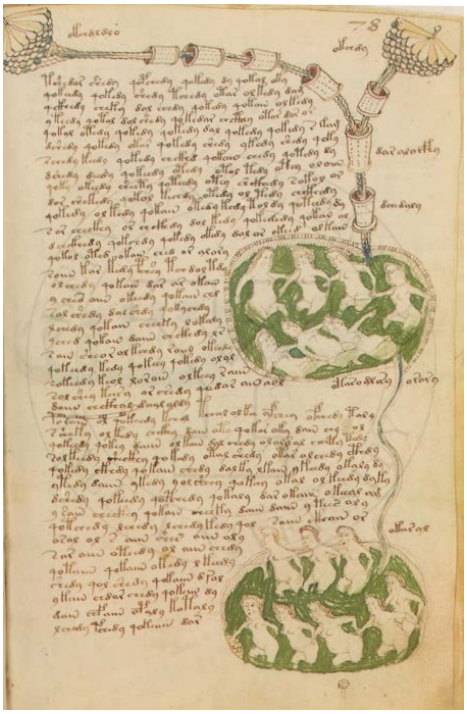

Figure 1: A page from the Voynich Manuscript depicting tiny naked women bathing in bizarre balneological constructions.

Most research into unsupervised learning of sequential data focuses on datasets that have some known structure, so that the performance of the algorithm can be easily evaluated by comparing detected structures, to already known ones. One of the benefits of systems like these, however, is that they can be applied to data with truly unknown structure, to gain insight into the nature of the data (as opposed to using the systems to simply parse new data automatically). One of the more extreme examples of this kind of data, would be undeciphered manuscripts of an unknown nature. There are several manuscripts (often written hundreds of years ago) that are composed of some language-like sequence of

characters, but have so far defied any attempt at translation. Examples are the Rohonczi codex, thought to have been created in the early 16th century, or the Codex Seraphinianus, an artistic creation from the 1970's.

The most famous of these, however, is undoubtedly the Voynich Manuscript. Named after Wilfrid M. Voynich, who discovered it in 1912, the manuscript has resisted attempts at decipherment by cyptographers and linguists of many periods, including a group of US codebreakers during World War II. The manuscript contains roughly 200 000 characters, in an unknown script. It is written from left to right, without punctuation marks and richly illustrated with crude drawings of unknown plants, the signs of the zodiac, astronomical diagrams, and naked bathing women, in complex 'plumbing' structures.

Unsupervised systems for learning natural language would be great tools for the analysis of texts like the Voynich manuscript, since systems like these can discover great hierarchical structure in texts, using just the raw data. This project aims to research some of these possibilities. The main procedures that will actually be used on the manuscript are MEX and Adios.

MEX is an algorithm that can discover notable subsequences in sequential data. It can for instance, with reasonable accuracy discover words in a text in which the spaces have been omitted. Since it has been questioned whether the spaces that separate the words in the Voynich Manuscript can be trusted as the true delimiters of the text, MEX can help in identifying other meaningful ways of tokenizing the Voynich Manuscript.

Adios (Automatic Distillation of Structure) is a system that uses MEX to discover hierarchical structure in a text, similar to parse trees based on rewrite rules. The structures that Adios comes up with in the Voynich text can be used to compare the text to other languages and sequential structures (by preforming Adios on them as well) in hopes of gaining further insight into the nature of the text. One of the advantages of Adios for this kind of approach, is that Adios can generate a histogram of how sequences are broken up into various types of patterns. Using these histograms, a dendrogram can be calculated for texts in various languages showing how closely they are related. With this approach, we can calculate the 'distance' between the text we are analyzing and several known languages, which can be a helpful clue to the nature of the text.

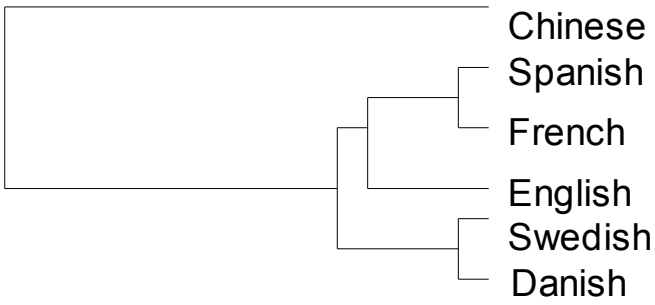

Figure 2: This dendrogram was computed by Solan et al. [2005] by computing the euclidean difference between pattern spectra for several corpora. It clearly show the expected linguistic classification for the languages. The corpora use where six translations of the Bible.

A short overview of the Voynich Manuscript

The manuscript was discovered by book collector Wilfrid M. Voynich in 1912. Since that time, the history of the manuscript has been researched extensively. It can be traced back as far as Emperor Rudolf II of the first German Empire, who apparently acquired it for 600 ducats.

In the 17th century the manuscript was in the possession of Georg Baresch, an obscure alchemist from Prague. Baresch enlisted the help of Athanasius Kircher, a famous scholar from that time, to decipher the manuscript. Kircher was especially famous for deciphering the Egyptian hieroglyphics (though his translations have now mostly been refuted). Kirchers response is unknown, however it is known that Kircher attempted to acquire the manuscript. Baresch wouldn't part with it, but when Baresch died, the manuscript passed to his friend Johannes Marcus Marci, who promptly sent it to Kircher. The letter that Marci sent with it is still attached to the manuscript.

For the 200 years following this exchange the manuscript was (probably) kept with the rest of Kircher's personal correspondence at the library of the Collegio Romano University. In 1870, when the forces of Victor Emmanuel II of Italy annexed the papal state the book was moved to the private collection of then rector magnificus Petrus Beckx to keep it safe.

In 1912, when the Collegio Romano sold part of its collection, Voynich acquired the manuscript along with 30 others.

Figure 3: Rudolf II, (1552-1612), Emperor of the Holy Roman Empire and earliest known owner of the Voynich Manuscript.

Proposed Solutions & Hypotheses

Since its discovery by Voynich there have been many theories about the nature of the Manuscript, and some (refuted) claims of translation.

Encryption

One popular hypothesis is that the text is an encrypted form of some western language. However, any proposed encryption method would have to account for the odd statistical properties of the manuscript. Similar theories have been proposed that suggest steganography, a technique that creates a mainly meaningless text, with the true message hidden in seemingly insignificant details.

Artificial Language

Another way to explain the odd characteristics of the manuscript is the theory that it was written in some constructed language. The earliest known examples of such languages fall in the 17th century, but it is always possible that they were in use before that period. A category

based philosophical language, where each word is a path down a classification tree, with each suffix a subcategory, would account for the strange structure of the words and the high repetition.

Hoax

All students of the Voynich Manuscript have had to consider the possibility that there is no deeper meaning behind the text, that the manuscript is a simple fabrication, used perhaps to con Rudolf II out of 600 ducats. Most serious students of the manuscript have reached the same conclusions, namely that there are structures in the text that are not necessary for a fabrication. However, considering the lack of success in deciphering the text and the odd nature of the illustrations, this option cannot be easily dismissed.

Glossolalia

The repetitious text, and the seeming lack of any word level structure are quite similar to the text that is produced by people 'speaking in tongues'. The possibility of the manuscript being transcribed glossolalia (or perhaps even a form of written glossolalia) has been suggested more than once, but since large sets of reference material of transcribed glossolalia are scarce, this theory is difficult to follow up on.

Unsupervised Natural Language Processing

Before resorting to the more complicated unsupervised learning algorithms, there are several, more simple techniques, that are used often in both linguistics and cryptanalysis, that can show a great deal about the nature of a text.

Simple Statistics

Some basic statistics are often enough to crack the simplest of codes. For instance, a substitution cipher (a code where every character is simply replaced with another, such as a=b b=c, c=d, etc) can be easily cracked by counting the frequency of each character. This can then be matched to the frequencies of characters in the original language, and the message can be decoded. If the original language is not known, simple statistical methods can also help greatly to determine it. A distribution of word lengths in the text can be matched to word length distributions from known languages to find the original language (provided that the encryption method didn't alter the distribution).

N-grams and Entropy

Like determining character frequencies, word frequencies can help to match a text to some known language. The most frequent words, their lengths and the way they occur in the text can be compared to known other texts in hopes of finding notable similarities or differences. If, instead of single words (or characters), the frequencies of all sequences of tokens with some length n (n grams) are counted, this gives the cryptanalyst some information on word

order. For instance, if a trigram is created (all frequencies of all 3-token combinations are counted) it can be used to estimate p(word3| word1 word2), that is, the probability that word3 follows word1 and word2 (in other words, the probability of encountering word3, if word1 and word2 have just been encountered). These probabilities can, as always, be matched to those of known languages, but they can also be used to calculate the entropy of the text.

Entropy can be seen as a measure of the randomness of a sequence. Natural language is a 'random' sequence, since we can not perfectly predict next word or character in some sequence, but it is not perfectly random, since we do know that some words or characters are more likely to occur. This 'degree of randomness' can be measured by calculating the entropy of a sequence. The basic formula for calculating the entropy H is as follows:

$$H = -\sum_i p(i) \log_2 p(i)$$

Where p(i) is the probability of encountering token i of all n possible tokens (this can be estimated by dividing the number of times token i occurs by the total number of tokens in a corpus). This value can also be seen as a measure for how much information each new token gives the reader (or how many bits would be needed minimally to encode the token).

However, once we've already encountered tokens, the probability of what token can come next changes. (for instance, if we've encountered the word “the”, then “table” is a more likely choice for the next word than “a”, whereas without any knowledge of preceding tokens, “a” would be the more likely choice).

To calculate the entropy of a text based on already encountered tokens we apply the same principle, but instead of summing over all possible values of p(i), we sum over all possible values of p(i| j), to calculate the first order entropy (based on one previous token):

$$H = -\sum_j p(j) \sum_i p(i|j) \log_2 p(i|j)$$

A general formula for the m-th order entropy then becomes:

$$H = -\sum_{i_1} p(i_1) \sum_{i_2} p(i_2|i1) \ldots \sum_{i_m} p(i_m|i_{1,} i_{2,} \ldots, i_m) \sum_{i_{m+1}} p(i_{m+1}|i_{1,} \ldots, i_m) \log_2 p(i_{m+1}|i_{1,} \ldots, i_m)$$

The first or second order entropy of a text can give an analyst a great deal of information on the structure of the underlying text.

Automatic induction of Parse Trees

When all these methods fail to produce results that are useful enough to decipher the text, more sophisticated unsupervised techniques can be used, for instance to determine the grammatical structure of the sequence. Since the work of Noam Chomsky, linguists have modeled sentences and grammar as parse trees derived from rewrite rules. Consider, for instance, the following grammar:

$S \rightarrow NP\ VP$
$NP \rightarrow DET\ N$
$VP \rightarrow V$
$DET \rightarrow the$
$N \rightarrow man$
$N \rightarrow woman$
$V \rightarrow walks$
$V \rightarrow dances$

By 'rewriting' the sentence symbol *S* with *NP VP*, *NP* with *ADJ N* and so on until all classes have been replaced with words, new sentences can be generated or existing sentences can be 'parsed' in to a tree structure.

Several sentences can be parsed by this grammar, each with it's own tree. In this manner linguists have constructed large and complex grammars that are meant to be able to parse as much of a language while dismissing as many linguistically incorrect sentences as possible.

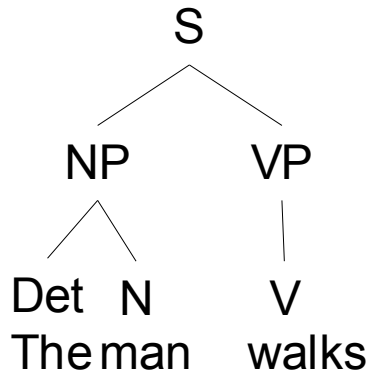

Figure 4: A simple parse tree.

Unsupervised algorithms for learning grammars can automate this process of distillating grammars from large samples of natural languages. Two such methods, ABL and Adios, will be discussed.

ABL

Alignment-Based Learning (ABL) was developed by van Zaanen in 2000. The algorithm works by comparing every sentence in a corpus to every other sentence. Comparing two sentences to one another it tries to find constituents (nodes in the parse tree), based on the principle that a constituent can be replaced by another constituent of the same kind, without rendering the sentence grammatically incorrect. For instance in the sentence "The man walks" the noun-phrase *The man* can be exchanged for *A beautiful woman*, and the sentence would still be correct. Reversing this principle, if a group of subsequences (parts of sentences), can be replaced by each other, they are constituents of the same kind. For instance, if the algorithm compares the sentence *What is a family fare?* to the sentence *What is the payload of an African swallow?*, it determine that the phrases *a family fare* and *the payload of an African swallow* can be interchanged, and are therefore constituents of the same kind (noun phrases).

Of course not all two sentences can be aligned this cleanly, as figure 5 shows. In cases such as these, where the alignment is ambiguous, the edit distance algorithm by Wagner And Fischer (1974) is used. This algorithm

from ()$_1$ San Fransisco (to Dallas)$_2$
from (Dallas to)$_1$ San Fransisco ()$_2$

from (San Fransisco to)$_1$ Dallas ()$_2$
from ()$_1$ Dallas (to San Fransisco)$_2$

from (San Fransisco)$_1$ to (Dallas)$_2$
from (Dallas)$_1$ to (San Fransisco)$_2$

Figure 5: Three different ways to align three sentences, with option three as the preferred alignment.

finds the minimum number of operation needed to get from one sentence to the other, where allowed edit operations are the insertion, deletion or substitution of a word. Those places in the sentence where no operation was applied, contain the identical words and the rest are constituents.

While results for ABL are certainly impressive, one major problem in using ABL to analyze manuscripts such as the Voynich Manuscript is that it relies very heavily on the sentence as a unit. Because it starts with the sentence constituent and works its way down the parse tree, the sentences need to be correctly delimited, or the algorithm is useless. Of course, the Voynich Manuscript and most manuscripts like it, lack any kind of clear punctuation marks, which means that sentence ends aren't easily detectable. One solution would be to delimit the text with the paragraph ends, which could combine several sentences into one, but it would ensure that no sentences are cut of. The problem with this approach is that a very large corpus of sentences (or rather, paragraphs) would be needed to be able to match constituents, and the Voynich Manuscript is a small corpus as it is.

Another approach altogether would be to apply ABL at the character level, using word ends for sentence delimiters. Applied to English, such an approach would yield at least some structure. Consider for instance the words “walking” and “dancing”. Applying ABL to these would determine 'walk' and 'danc' to be constituents. Since the words of the Voynich Manuscript seems to be much more structured than those of English (eg. most characters only occur in specific places in a word), ABL should be able to derive a very distinct structure from the Voynich Manuscript at the character level.

Considering the problems described above, the application of ABL to the Voynich Manuscript is left as future research.

Adios

Adios was developed by Solan, Horn, Ruppin and Edelman and was first presented in 2002. The algorithm consists of two parts. The first part is MEX, which detects sequences of tokens that are notable (like words, or parts of words at the character level, or common phrases like “in spite of”, at the word level). The second part, Adios, makes use of MEX to detect words that often occur in the same context (using the same principle of interchangeable constituents as ABL) and creates a new 'token' for those tokens, which takes their place. Applying this principle multiple times creates a tree of tokens which, contrary to ABL, grows bottom up.

MEX

MEX and Adios both operate on the same datastructure, which is constructed as follows. A graph is constructed, with at each node (or vertex), a token from the corpus. Each sentence in the corpus is then drawn as a path along the graph. This creates a graph with directional edges, that can be cyclical, and in which edges from one node to itself are allowed. Two special nodes are also created to mark the beginning and

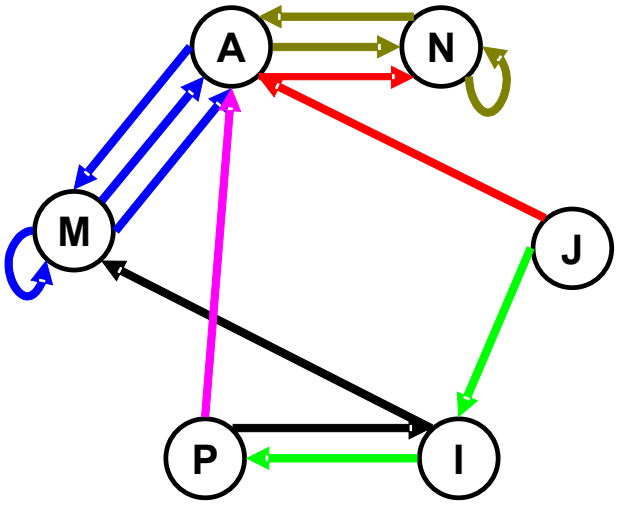

Figure 6: A simple example of the graph that MEX and Adios operate on. In this case, the letters are used as tokens and several words are marked out as paths on the graph.

end of each node (this is not strictly necessary. but it can be helpful).

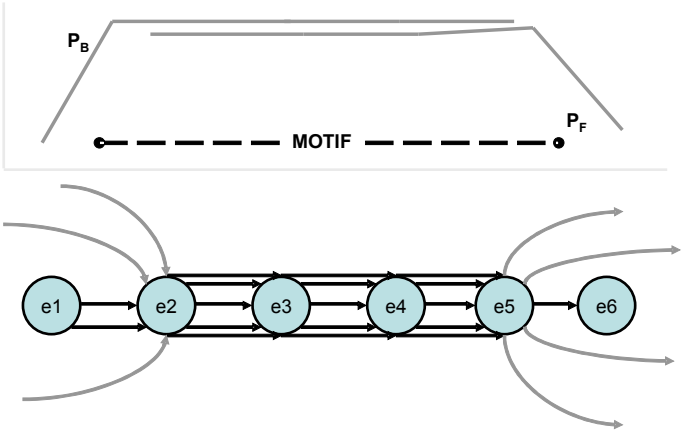

Figure 7: A schematic representation of how motifs are detected along a path (e1 to e6 in this case). At e6, a lot of paths diverge from the path we're following, causing the forward probability to drop. between e2 and e1 the backward probability drops in the same way. Between these two drops a motif is defined.

This graph is then used to find notable sequences of tokens, which we call motifs or patterns. Informally, we can describe a motif as a sequence of tokens that has a lot paths following it, with a sudden drop in the number of paths at both ends of the sequence. Mathematically speaking, however we will not base the definition directly on the number of paths, but on a Markov model of probability. Normally a Markov probability like p(n|joh) (the probability of encountering n, given that j, o, and h have been encountered in that order already) would be calculated with a 4-gram. However, using our graph we can easily estimate this probability counting the number of paths for j to n, through o and h and dividing it by the number of paths from j to h, through h. Thus for any sequence (e_1; e_n)(which is shorthand for e_1, e_2, e_3, ..., e_n) we can define the forward moving probability:[1]

$$P_F(e_1;e_n)=p(e_n|e_{1,}...e_{n-1})=\frac{l(e_1;en_n)}{l(e_1;e_{n-1})}$$

and the backward moving probability:

$$P_B(e_1;e_n)=p(e_1|e_n,...e_2)=\frac{l(e_1;en_n)}{l(e_2;e_n)}$$

Where l(s) stands for the number of paths that follow sequence s. Thus, the forward probability defines the probability of encountering some token *after* encountering some sequence of tokens, and the backward probability of encountering some token *before* a sequence of tokens (moving backwards through the text). For a sequence of one token, the probability is:

$$P_F(e_1)=P_B(e_1)=p(e_1)=\frac{l(e_1)}{\sum_{e_i} l(e_i)}$$

We want to define the beginning and end of a motif as sudden drops in the forward and backward probability (a drop in backward probability defines the beginning, a drop in forward probability the end). We define the drop between the second-to-last token in a sequence and the last token, based on the full sequence, as:

1 The creators of Adios called this right and left moving probability. Since not all sequences move from left to right, I found it more intuitive to adopt this terminology.

$$D_F(e_1;e_n)=\frac{P_F(e_1;e_n)}{P_F(e_1;e_{n-1})} \quad D_B(e_1;e_n)=\frac{P_B(e_1;e_n)}{P_B(e_1;e_{n-1})}$$

We require these drop strengths to be below a certain threshold n (between 1 and 0, usually around 0.65), to be considered as start and end points for the motifs.

Since we are at times dealing with very small samples (only several paths crossing a node), we need to make sure that th data we have could not represent a drop incidentally, that is, against the odds. To determine this we calculate the significance of the drop. We assume that

$$\frac{P_F(e_1;e_n)}{P_F(e_1;e_{n-1})} \geq n \;\rightarrow\; P_F(e_1;e_{n-1}) \geq n\,P_F(e_1;e_n)$$

That is, our drop is *not* stronger than our threshold and we calculate the probability of getting the numbers paths that we got. We require this probability to be smaller than some value a (usually around 0.01). The method for determining the significance is described by Solan et al. (2004).

With these definitions we can begin our search for motifs. For every path that we have, we find all forward and backward drops along the path. In order to do this we check all the sequences along the path from all possible positions i to all possible positions j. (Note that this describes all forward sequences, all backward sequences and all single token sequences). For all the sequences s we calculate the probabilities $P_{F/D}(s)$.

Since all these sequences start at some point along the path and end at some point along the path, we can plot the probabilities in a square matrix, where the column number j determines the start point and the row number i determines the end point:

$$M_{ij}=\begin{cases} P_F(e_i;e_j) & if\ i>j \\ P_B(e_j;e_i) & if\ i<j \\ P(e_i) & if\ i=j \end{cases}$$

Written out explicitly, M looks like this:

$$M=\begin{Bmatrix} p(e_1) & p(e_1|e_2) & p(e_1|e_1e_3) & \cdots & p(e_2|e_1e_3\ldots e_k) \\ p(e_2|e_1) & p(e_2) & p(e_2|e_3) & \cdots & p(e_2|e_3e_4\ldots e_k) \\ p(e_2|e_1) & p(e_2|e_1) & p(e_3) & \cdots & p(e_3|e_4e_5\ldots e_k) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ p(e_k|e_1e_2\ldots e_{k-1}) & p(e_k|e_2e_3\ldots e_{k-1}) & p(e_k|e_4e_5\ldots e_{k-1}) & \cdots & p(e_k) \end{Bmatrix}$$

In this matrix we can now easily mark the significant drops between vertically adjacent cells. What we are looking for to create a motif is a $D_F(S_f)$ and a $D_B(S_b)$, such that the sequences S_f and S_b used to calculate the drop overlap and the backward drop comes before the forward drop. Out of all possible motifs that satisfy these requirements, the one for which the average of the significances of both drops is lowest is returned.

In applying this procedure to the Voynich manuscript, one change can be made to the algorithm. Since we are not using Adios (yet) to apply the procedure continuously to a

changing graph and just want the motif, and since we have a relatively small corpus (and thus, will want all the information the procedure can retrieve from it), instead of using just the leading motif for each path, we extract all motifs that are significant. In fact, extracting just one motif per path is rather arbitrary in general, as one path may contain many interesting motifs, whereas others may only contain one barely significant motif. By returning all motifs we can be sure we're not overlooking anything.

Distillation of Structure

After the initialization of the graph, Adios goes through three stages to distill the grammar from the corpus.

1. **Pattern Distillation**

For all paths, MEX is performed to find the leading motif in the path, called P. A new Vertex is then created for P, and the graph rewired, so that paths that crossed P will now cross the new vertex instead.

2. **Generalization, first step**

A context window is defined as a sequence of L nodes along a path (L being a parameter of the algorithm, usually set at around 3 to 5). A slot is then defined as one of the nodes in the context window. A *generalized search path* is then defined as all paths through the nodes of the context window, with the exception of the slot (note: the generalized path is thus as long as the context window and branches into several different nodes at the slot). MEX is then performed on the generalized search path, to determine the leading motif along it.

From all possible slots in all possible context windows on all paths in the corpus, we determine a generalized search path and a leading motif. The leading motif P from all searches is selected, and an equivalence class containing those vertices that appeared at the slot in the generalized search path. A new vertex is now created, like in the first step, that replaces the nodes in P on all those paths that fully crossed P.

3. **Generalization, bootstrap**

We once again define a context window on a path and a slot j in the context window, as in the previous step. This time, however, we compare the vertices encountered at j to all existing equivalence classes finding the one that has the greatest overlap E(j) (returning none if the overlap is below a threshold w, usually 0.65).

For this slot and its equivalence class we go through all the vertices in the context window except the slot, defining the as a new slot, k. We check all the paths going through the context window, k and the equivalence class E(j) (or the vertex j, if no equivalence class was returned), creating a *reduced generalized search path*.

MEX is performed on this search path to extract its leading pattern. For the leading P of all searches, an equivalence class is constructed for its slot k (unless it overlaps perfectly with E(j)), a new vertex is again created for P, and the graph is rewired.

Step 3 is then repeated until no further significant patterns can be found.

Application to the Voynich Manuscript

Both MEX on its own and Adios can be useful in analyzing the structure of texts like the Voynich Manuscript. The motifs that MEX returns can be analyzed in various way to compare them to those that are extracted from corpora in other languages. A second possibility is to re-tokenize the corpus with the motifs (rather than the original word based tokenization). In the case of the Voynich Manuscript, doubt has risen whether the words can be trusted as tokens (especially with regards to the high second order entropy). A way to circumvent this would be to disregard the spaces and retokenize the corpus with MEX motifs. If the same method is performed on a corpus in a second language, the language can be compared to that of the manuscript (in terms of entropy, token length, etc.) without having to rely on the Voynich words.

The relevance of the Adios algorithm needs little explanation. Revealing the inner structure of texts such as these can be considered the holy grail of their analysis. Of course the distilled structure still needs to be compared to the results over other corpora in order to give them meaning, but a deeper general structure than a grammar is difficult to imagine.

Results

MEX

french	english	bible	vms A	vms B	vms full
(10661 / 393 kb)	(12025 / 404 kb)	(6259 / 239 kb)	(2646 / 67 kb)	(8644 / 141 kb)	(14325 / 227 kb)
ssepar	th	andhe	ai	ka	dyc
eparto	andthe	th	ii	cheody	ota
heures	hic	andof	daii	edyqokedy	or
ileas	tio	andbe	cheol	ote	oloka
ssepa	re	he	okch	keedyote	edyqokedy
separ	which	osephara	okcho	kai	ino
ou	no	harand	cth	olsh	okedych
le	thatio	hara	rcho	heolch	ka
an	edfrom	un	iinshe	inch	ii
hile	ounder	ofth	sho	she	dyq
leme	compass	hear	chyqo	airo	keeol
de	from	eland	dai	eeych	hck
nsle	thatthe	halso	otcho	ched	okch
asfo	ngthe	thou	che	yqo	tch
rtou	place	hold	aiincho	ckh	otar
endan	anda	herand	ctho	ai	eolshedy
eu	ofthis	years	cph	aiinqo	eyq
elle	andby	shall	otch	dylsh	oteodych
ati	enight	osep	olcho	aiinokeedyq	yqo

Table 1: This table shows the top 20 motifs (sorted by weight) for various languages. The numbers below the language names represent the number of motifs for that particular corpus, and the length of the corpus in kilobytes. The French corpus used is the original version of Around the world in 80 days *by Jules Verne, The English corpus is* Frankenstein *by Mary Shelly. The column labeled* bible *contains the results for the King James bible, cut off to make it roughly the same length as the Voynich Manuscript. The columns labeled vms A and vms B show the motifs for a certain way of dividing the manuscript in two parts according to strong statistical differences in the text. The last column shows the results for the entire manuscript*

At the character level, MEX performed quite well, returning around ten thousand motifs for corpora the length of a 200 page paperback. As wonderful as such large amounts of data are, they also mean large amounts of time in analyzing. On the surface differences between the sets of motifs and their attributes are hard to find. Most collections have roughly the same averages and the same distributions in terms of motif length, significance and weight (the strength of the drops in probability). Because the work required to thoroughly analyze the data falls outside the scope of the project, presented here are only a quick overview of the data and some immediately apparent differences.

The tables for Voynich A and Voynich B clearly show very different motifs for both languages. In fact the overlap for the top 1000 motifs (by weight) for both texts is only 15%. However, from the results on the French corpus, it can be seen that the subject matter can have quite an influence on the resulting motifs; parts of the name 'Passepartout' (one of the characters from “Around the world in 80 days”) occur several times in the top 20. Motifs which would obviously not be found in other French corpora. So from these results, we can only conclude that vms A and vms B are different, to find out whether this is caused by some structural difference or simply a difference in subject matter would require some further analysis.

At the character level, the Voynich language produces more motifs than an English corpus of similar length. This seems to suggest more structure at the character level than is seen in most western languages. This idea is backed up by analysis of the character level entropy and several proposed character level grammars and finite state machines, that were successful in describing very large percentages of the manuscript.

To test the validity of these finds and the MEX implementation itself, it was also tested on the decimals of pi (roughly 400 kb's worth), which, as would be expected, returned no motifs.

At the word level MEX fails to find any motifs. The finds in other languages range from 44 on French to 1 on Latin. Since (like Latin) the Voynich Manuscript has a very high first and second order word entropy, these results are not surprising. As First or higher order entropy can be interpreted as a measure for how well a text can be modeled as a set of Markov probabilities (the lower the entropy the better) and MEX basically works by modeling the text as a set of (variable order) Markov probabilities, it's clear that MEX won't do well on texts the a high second order entropy. One additional conclusion that we can draw from this result is that entropy orders higher than second would be very unlikely to be any lower. MEX calculates Markov probabilities of very high orders (the length of the path), and failed to find any significant pair of overlapping drops.

Adios

Since an actual implementation and in depth study of Adios and its behavior lies somewhat outside the scope of this project, we limit ourselves to a very basic analysis with the default values (n = 0.65, a = 0.01 and L = 4). The following results were generated with Adios-lite, a demo implementation that is freely available from the Adios website. It is limited to 100 patterns, but serves to illustrate the effects of Adios on the Manuscript.

At word level, MEX achieved no results on the manuscript. No structure at all was found by Adios. Parallel runs on corpora of similar length showed at least some results, for languages like English, French or Chinese. Even in Latin, a language with a very high second order entropy, much like the Voynich language, five patterns were found, in a corpus a bout 75% the size of the Voynich manuscript. The the results of MEX at word level, these results can be attributed to the manuscript's high second order entropy. The context windows that Adios uses to search for words appearing in the same context is relatively small (the default setting is 4 words) since the second order entropy is so high, very few sequences of words will occur often enough to provide such contexts. It seems that Adios is not well suited to detect the word level structure of the Voynich Manuscript, if there is any.

Considering how well MEX performed at the character level, it should be very interesting to see Adios perform at the character level. Unfortunately, because of the low number of nodes (one per character) and (consequently) the high number of paths, Adios becomes very slow at the character level. It spends on average 3 hours on one path per run over the manuscript. Considering that there are 1804 paths (paragraphs) in the manuscript and it needs to make several runs over over all of them, this test is far too time intensive for this kind of project. (Let alone running it on other corpora as well for reference) The output of the running program did however suggest that Adios discovered patterns and was rewiring several nodes of the graph.

Conclusions

The MEX and Adios algorithms were applied to the Voynich Manuscript, in hopes of analyzing its structure. The efforts using MEX at the character level resulted in a lot of data that will certainly be of use in the further analysis of the text.

At the word level both MEX and Adios returned no results at all, which seems to point towards a low level of structure, or at least, no structure in terms of Markov models.

The character level analysis using Adios had too long a running time to finish before the deadline of the project, but the output of the program looked hopeful.

On the whole, unsupervised learning algorithms can certainly be a useful tool in the analysis of texts like the Voynich Manuscript. However, results are not guaranteed and a solid understanding of their workings and behavior is necessary to interpret the results.

Further Research

Some possible areas of future research:

- Various other unsupervised learning approaches are available. Most, however, will have to be adapted somehow to be able too deal with small datasets and a lack of sentence delimiters.

- The motifs found by MEX can be used to tokenize the corpus (ignoring spaces, or simply treating them as delimiters) using these corpora, the Voynich Manuscript can be compared to other corpora at a higher level than character level without depending on the word delimiter. Some method would have to be designed to find the best way too 'cover' the text using the motifs. (ie. choosing those motifs that allow as many of the characters of the corpus to be part of a motif-token as possible). Some other way of defining the best tokenization (such as giving motifs of greater strength or significance precedence) may be preferable.

- Some students of the manuscript have suggested that the words of the Voynich Manuscript should be split into two character syllables prior to analysis. Both MEX and Adios may have more luck with these tokens than with word based tokens.

- The theory that the manuscript contains transcribed glossolalia is very difficult to verify without a sufficiently large corpus of transcribed glossolalia (preferably from the same person). I researched the possibility of obtaining such a corpus, but the three people that would be able to help in this respect are all currently unreachable for various reasons. Perhaps in the future, such corpora will become available, so that tests like these and others can be performed on it.

References

- Zach Solan David Horn, Eytan Ruppin and Shimon Edelman (2005) Unsupervised learning of natural languages.
- Zach Solan David Horn, Eytan Ruppin and Shimon Edelman (2005). Motif extraction and Protein Classification.
- Zach Solan David Horn, Eytan Ruppin and Shimon Edelman (2004) Supporting online material.
- Menno van Zaanen (2000) ABL: Alignment-Based Learning.
- Rene Zandbergen(2000) From digraph entropy to word entropy in the Voynich MS