# An alternative Hidden Markov Model for vowel-consonant identification,

## with an application to the Voynich MS text

*René Zandbergen (European Space Agency, Darmstadt)*

**Introduction**

Hidden Markov Modelling (HMM) is a statistical technique that was developed in the 1960's, but only found significant application considerably later. One of these applications is analysis of written text on a character basis, in particular for vowel-consonant detection. A good description of this application is given in Ref.1.

HMM analysis of a text assumes that a sequence of characters that we observe can be modelled by another hidden sequence of states. The number of different states is always much smaller than the number of different characters. In the case of vowel and consonant detection we are dealing with two states. This involves two processes. The first process defines how one hidden states causes the next hidden state, while the second process defines which character is generated or 'emitted' by each state. The following figure from Ref.1 may clarify this.

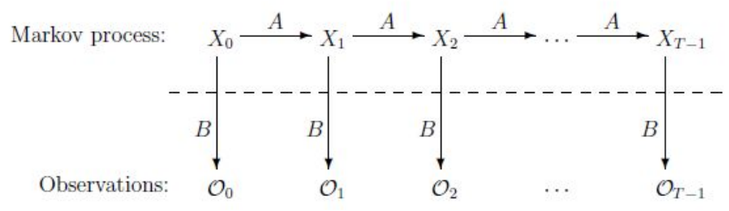


*Figure 1: schematic representation of a Hidden Markov Model, from Ref.1*

The Markov process A describes the transition from one hidden state X to the next hidden state. Process B describes the transition from a hidden state X to an observed character O. Note that the analyst only sees the observations O, while the hidden states X (above the dotted line) are not known, but can be estimated. The sequence of observations O is nothing more than the string of characters that make up the text.

Both processes A and B are stochastic, which means that they consist of a list of transition probabilities. In mathematical terms, both A and B are represented by a matrix. $A$ is a square matrix with dimension $N$ x $N$ where $N$ is the number of states. $B$ is a rectangular matrix with dimension $N$ x $M$ where $M$ is the number of different characters.

HMM analysis now consists of computing the probabilities related to both processes, i.e. the numbers in the matrices $A$ and $B$, based on an input text. The standard algorithm for doing this is the Baum-Welch algorithm (see e.g. Ref.1). This algorithm is indeed capable of detecting vowels and consonants in strings of meaningful text in known languages, when it is being applied with two states.

**An alternative Hidden Markov formulation**

A disadvantage of the Baum-Welch algorithm is that it requires several hundred iterations in order to converge, and in each iteration the entire text has to be processed. This paper presents an alternative algorithm that requires processing the text only once, in order to create a character transition frequency matrix. It then uses this matrix to estimate the best matching $A$ and $B$ matrices. The processing time becomes essentially independent of the length of the text. For lack of space, the method can only be summarised briefly here. More details are provided in Ref.2.

A key quantity in the alternative method is the vector $p$ with dimension $N$ of stationary probabilities. It represents the probability that each hidden state occurs in the text. In addition, the method introduces the $E$ and $D$ matrices as normalised versions of the $A$ and $B$ matrices, as follows:

$$e_{i,j} = p_i \cdot a_{i,j} \quad \text{and} \quad d_{j,k} = p_j \cdot b_{j,k} \qquad \text{(Eq.1a,b)}$$

such that:

$$\sum_j e_{i,j} = p_i \quad \text{and} \quad \sum_k d_{j,k} = p_j \qquad \text{(Eq.2a,b)}$$

(By convention, for all square matrices used here, the first index refers to the left character or state of a pair and the second index to the right character or state.)

Furthermore, the method introduces the matrix $R$, as the matrix of probabilities that the state is $j$, given that the emitted character is $k$. For each column in the $R$ matrix (i.e. for each character), the sum of the column elements is one. The relation between the $R$ and $D$ matrices is as follows (where $c_k$ is the character frequency of character $k$):

$$d_{j,k} = c_k \cdot r_{j,k} \qquad \text{(Eq.3)}$$

For any given text we may estimate the vector $C$ of character probabilities and the matrix $T$ (of dimensions $M$ x $M$) of character transition probabilities simply by counting occurrences in the text (frequencies). In the following, we will tacitly equate probability with frequency.

In addition, the relation between the $R$ and $E$ matrices is as follows:

$$E = R \cdot T \cdot R^T \qquad \text{(Eq.4)}$$

Once an HMM using $N$ states has been applied to a text, we have obtained the best matching (according to some criterion) $A$ and $B$ matrices. We may now imagine that we let this Hidden Markov Model run for a while, in order to generate some 'simulated text'. We can then predict what will be the transition frequencies of this simulated text, without actually generating this text. This is the matrix $S$:

$$S = B^T \cdot E \cdot B \qquad \text{(Eq.5)}$$

Note that the character frequencies of the simulated text will be identical to those of the input text.

In the approach presented here, the $A$ and $B$ matrices are optimised such that the resulting simulated character transition frequencies $S$ are a best match for the actual transition frequencies $T$ of the text that is being analysed.

<u>This gives rise to the following iterative procedure or algorithm</u>:

Given an a priori $R$ matrix, or an updated $R$ matrix in the iterative procedure:

1. Compute $D$ using Eq.(3)
2. Compute $p$ using Eq.(2b)
3. Compute $B$ using Eq.(1b)
4. Compute $E$ using Eq. (4)
5. Compute $A$ using Eq.(1a) (note: this matrix is not strictly needed in the process)
6. Compute $S$ using Eq.(5)
7. Compute a 'score' from the difference between $T$ and $S$

For the score function, the best results have been obtained by using the Bhattacharyya distance (see Ref.3).

If another iteration is needed, the $R$ matrix is adjusted in some way, and the process is repeated until convergence is reached. The difficult part in this process is the method of adjusting the $R$ matrix, which is critical in order to achieve that the optimal solution is found. It is a 'hill climbing' procedure that is still undergoing further study. It becomes more complicated for higher numbers of states. At present, working algorithms have been defined for 2 and 3 states.

*The problem of the space character*

In HMM analysis, the space character is usually treated as a normal character, that can be emitted by any of the states. Treating word spaces in this way is not entirely satisfactory. Clearly, spaces behave differently from normal characters. This can be solved elegantly with the present method, because it is possible to define one additional state, which is forced to emit only the space character. At the same time, none of the other states are allowed to emit a space character. Effectively, the stationary probability of this state is known from the start and fixed. All entries in the $R$ matrix for this character and its corresponding state are also known from the start and fixed. The computational overhead for adding this special state is minimal, and experimentation shows that results with this approach are better. In the following, this approach will be referred to as an "$N$+1" analysis.

**Application to known plaintexts**

The alternative HMM described before has been applied to the English sample text: "The fall of the house of Usher" by Edgar Allan Poe. After cleaning the text (removing punctuation, consecutive whitespace and converting all characters to lower case) this text consists of 40,094 characters. To show the effect of the number of states, the text has been processed for the cases: 2 states, 2+1 states, 3 states and 3+1 states. The relative success can be measured by looking at the Bhattacharyya distances between the $S$ and $T$ matrices. The absolute values are not particularly meaningful, but the differences highlight the better performance of the $N$+1 models. Due to the correct mapping of the space character, already the a priori correspondence is better.

| Case: | 2 states | 2+1 states | 3 states | 3+1 states |
|---|---|---|---|---|
| **A priori** | 0.190 | 0.168 | 0.190 | 0.168 |
| **Final** | 0.145 | 0.128 | 0.134 | 0.114 |

To visualise a $T$ matrix of character pair frequencies, all values are normalised by dividing the pair frequency by the single character frequency of each of the two single characters. If the result is >1 (preferential combinations of characters), its hue will be red, and if it is <1 its hue will be blue. This is shown in Figure 2 below for two plain texts, one in Latin and one in German. In all square matrices the first character of each pair is on the vertical scale and the second character on the horizontal scale. The processing used a 3+1 state model and characters are sorted by state. Within each state, the characters are sorted by decreasing frequency.

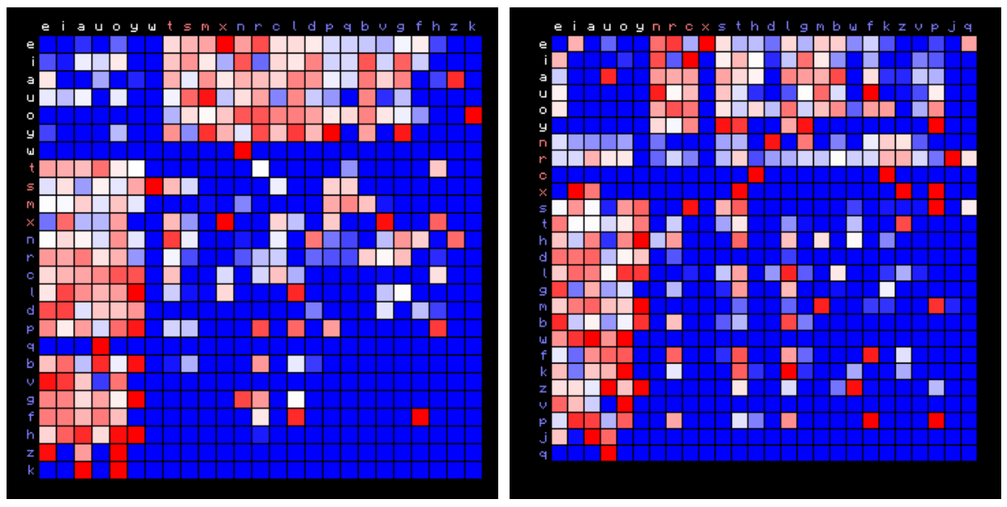


*Figure 2: Left: T matrix for a Latin text. Right: T matrix for a German text. The colour of the characters in the legends indicates the three different states. Both languages have one vowel state and two consonant states.*

This result for the two known plain texts clearly shows how vowels prefer to combine with consonants and vice versa, and also, that the majority of combinations exists. The figures are largely symmetric, at least in a qualitative sense. (See Ref.5 for more examples.)

**Application to the Voynich MS text**

The Voynich MS is a late medieval MS that includes a mysterious, unreadable text (see Figure 3 and Ref. 4).

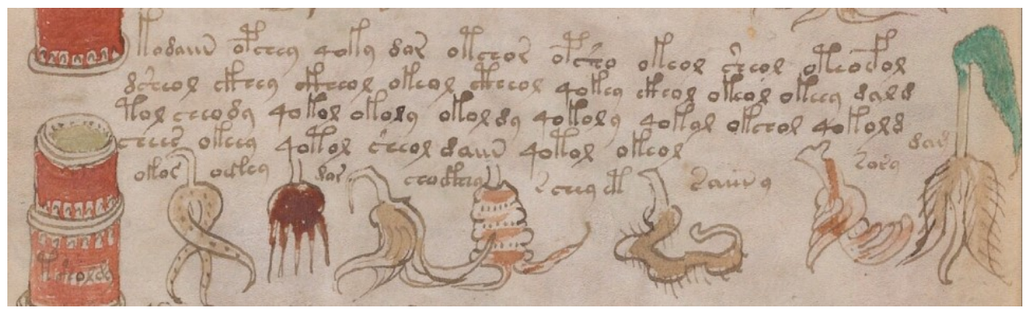

*Figure 3: a sample of the Voynich MS. Courtesy of the Beinecke Rare Book and Manuscript library, Yale University, New Haven CT.*

It is an excellent candidate for this type of analysis. Perhaps it is possible to detect vowels and consonants. The results of a 3+1 state application to an electronic transliteration of the Voynich MS text are shown in Figure 4 below.

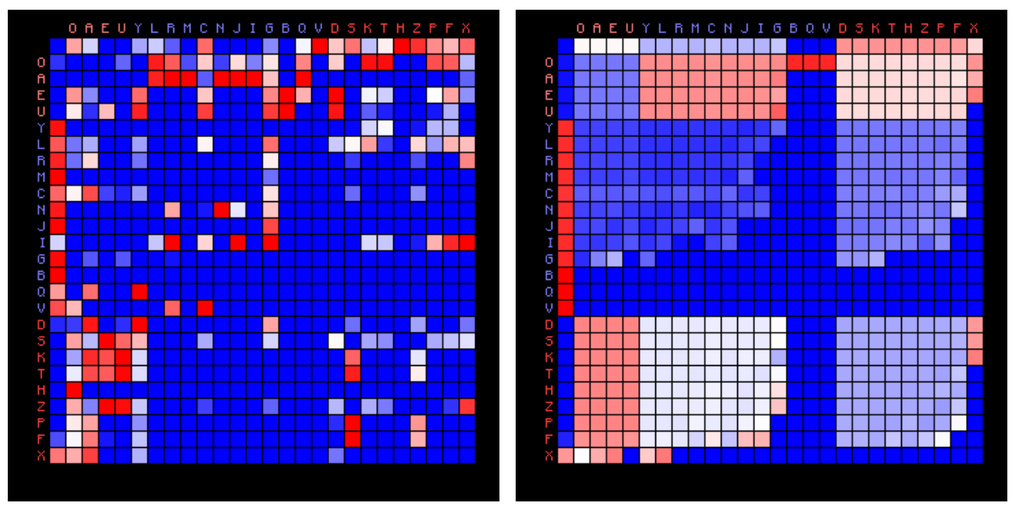


*Figure 4: Left: T matrix for Voynich MS text. Right: S matrix for the same text. The colour of the characters in the legends indicates the three different states, in addition to the space state.*

For the Voynich MS text the figures are not at all symmetric and only a much more limited set of character pairs seems to be allowed. This text behaves in a very different manner from the known plain texts. In particular, we see from the $S$ matrix that the space state is preferentially followed by state 3, state 1 by state 2, state 2 by space and state 3 by state 1. In summary:

Space → State 3 → State 1 → State 2 → Space

**Conclusions**

An alternative method for HMM analysis has been presented and initial results appear promising. The method is more efficient than the Baum-Welch algorithm, mainly because it optimises a different quantity. Baum-Welch maximises the probability that the string of plain text characters was generated by the Markov process, while the alternative method maximises the similarity of the character pair distributions. The possibility to classify the space character as a separate state appears to bring an advantage.

More work is needed on the search algorithm that finds the optimal $R$ matrix.

Application to the Voynich MS text shows that this text behaves differently from plain texts that exhibit a 'normal' vowel-consonant alternation. Instead, there appears to be a word pattern with preferential positions of characters within a word. Such patterns were already recognised by earlier students of the MS. Higher-state analyses will be required to investigate this in more detail.

**References**

1. Stamp, Mark: A Revealing Introduction to Hidden Markov Models. Department of Computer Science, San Jose State University, October 17, 2018.
2. See: http://www.voynich.nu/misc/misc05.html
3. See: https://en.wikipedia.org/wiki/Bhattacharyya_distance
4. See: http://www.voynich.nu and the presentation at this conference by S. Guzy
5. See: http://www.voynich.nu/extra/sol_hmm.html

*Poster presented at the International Symposium: "Alchemical Laboratories. Practices, texts, material relics", February 19-21 2020, in Vienna and Oberstockstall.*