Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications 33 (2007) 667–682
Expert Systems with Applications
www.elsevier.com/locate/eswa

An HMM for detecting spam mail ☆

José Gordillo, Eduardo Conde *

Facultad de Matemáticas, Department of Statistics and Operations Research, Universidad de Sevilla, CP 41012 C/Tarfia S/N, Sevilla, Spain

Abstract

Hidden Markov Models, or HMMs for short, have been recently used in Bioinformatics for the classification of DNA or protein chains, giving rise to what is known as Profile Hidden Markov Models. In this paper, we show that these models can also be adapted to the problem of classifying misspelled words by identifying its primary structure through statistical tools. This process leads to a new learning algorithm which is based in the parametrization of the set of recognizable words in order to detect any misspelled form of these words. As an application, a method to classify *spam* mails by means of the detection of the adulterated words, from a blacklist of words frequently used by spammers, is described.

Keywords: Analysis of algorithms; Artificial intelligence; Hidden Markov models

1. Introduction

At present, mailboxes of most e-mail users are flooded every day by advertisements, normally of doubtful veracity and morality. Following recent rough studies[1] (Center for Excellence, 2004) on the global costs associated to this activity, it can be cited, merely to illustrate the impact of spam, that one hundred thousand million spam e-mails are sent every day, 36% of Internet users receive at least 10 spam e-mails every day, a new e-mail account (set up to experiment) received spam within 540 s, 37% of US e-mail is spam, EU businesses spend €10 billion each year to deal with spam and, in the US, this cost grows to $21'6 billion a year, two-thirds of all US spam e-mails contain fraudulent offers, forged headers, or other false claims suggestive of criminal activity.

The lack of control in Internet about this kind of harmful practices for millions of e-mail users limits the possible solutions to this problem. Therefore, the user will be obliged to filter the messages received manually or by means of a computer tool. The task to be solved is a classification problem, where a message must be labelled as *spam mail* or *ham mail* (non-spam mail). The object to be classified will be usually a chain of text characters, extracted from the message, and it may be even the complete message.

The automatic classification of objects has been widely studied for the last decades, having many different applications, ranging from pattern recognition in Artificial Intelligence (see, for example, Barr & Feigenbaum, 1986), to medical diagnosis by means of Classification Trees (see Bock, 2002; Breiman, Friedman, Olshen, & Stone, 1984) or Bayesian Networks (see Cowell, Dawid, Lauritzen, & Spiegelhalter, 1999). Recently, a new classification methodology, based on Hidden Markov Models (HMMs), has been successfully used in different fields, such as Speech Recognition (see Logan & Moreno, 1997; Poza, Villarrubia, & Siles, 1991; Rabiner, 1989; Rabiner & Juang, 1993; Sun, 1998), Fingerprint Identification (see Senior, 1997), New Technologies (see Ypma & Heskes, 2002), Climatology (see Hughes & Guttorp, 1999), or Bioinformatics (see

☆ This research has been supported by the Spanish Science and Technology Ministry and FEDER Grant Nos. BFM2002-04525-C02-02 and BFM2002-11282-E.
* Corresponding author.
E-mail addresses: jgordillo@us.es (J. Gordillo), educon@us.es (E. Conde).
[1] Wikipedia. The Free Encyclopedia. http://en.wikipedia.org/wiki/Main_Page.

0957-4174/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2006.06.016

Abbas & Holmes, 2004; Eddy, 1998; Krogh, 1998; Mount, 2001). For consulting an exhaustive list of references of HMMs in several fields, see (Cappé, 2001).

In Bioinformatics, a special case of HMMS, called Profile Hidden Markov Models, has been used for the classification of DNA or protein chains (see Baldi & Brunak, 2001; Hughey & Krogh, 1996). This problem is complicated due to the big amount of information which must be handled in an efficient way. Moreover, data may be corrupted as a consequence of problems in the reading of databases or even mistakes made during the mitosis process in cells (substitutions, insertions or deletions of nucleotids or amino acids). Despite these problems, classification methods based on these models work successfully, as a proof of this, it is sufficient to realize the big number of publications in specialized journals of the area (see Abbas & Holmes, 2004 and the references included there).

The classification problem of spam mail is, in some sense, similar to the identification problem in DNA or protein chains. Firstly, in both cases there are chains of characters. Secondly, the appearance of mistakes in the case of biological sequences, holding the primary genetic structure of the chain, is similar to the deliberate adulteration (made by spammers) of keywords (free software, cheap viagra, adult entertainment, etc.), used by the anti-spam filters which work with *blacklists* to detect automatically a junk message. For instance, in the sentence *Your needed soffttwares at Rock Bottom prri ce!*, a set of characters has been inserted in the words *software* and *price*, to make difficult the detection and subsequent classification as a spam message through a computer tool based on blacklists. However, in the same way what it happened with the primary genetic structure in biological sequences, certain morphological structure must be conserved to allow any user to recognize perfectly the advertisement received. Because of this fact, classification via HMMs is well-adapted to this problem.

This model can also be applied to the design of spelling correction tools for text processors or to new options in information search engines on the Internet, when the searched word has been written wrongly and the tool suggests a modification. After all, all these situations have in common a classification problem of words or sentences, with partial information and a possibly corrupted morphology.

For solving this problem, we will model it through an HMM. We will describe how to assign a set of parameters to each one of the words generally included in junk mail, how to measure the likelihood of a chain of characters (possible variants of these forbidden words, the usual ones in spam mail), and how to fix the thresholds for accepting or removing a message according to the set of words which were identified as forbidden words. The two first questions will be solved by generalizing the methodology used in Profile Hidden Markov Models, obtaining a new learning algorithm which parametrizes the set of words of the blacklist. The threshold calculation and the subsequent classification problem will be studied at the end of the chapter, by taking into account the misclassification costs.

2. Modelling the problem

A *spam e-mail* is defined as an unsolicited mailing, generally for advertising certain products, most of the times uninteresting or of doubtful veracity, in a massive way, eluding thus costs which should be paid if the product is advertised legally on the net.

One of the most widely used methods for detecting spam is based on the blacklists, sets of words with a high frequency of appearance in junk mail. An anti-spam filter classifies a message as unsolicited when it contains some of these forbidden words written several times.

These filters can be cheated by writing a forbidden word with some mistakes deliberately (substitutions of some letters, insertions of symbols or letters and so on). The many ways in which a spammer can adulterate a given word make that the relative frequencies for these variants will be low, hiding that the frequency for the original term is really high and, therefore, spam indicator.

So, a list which contains the word ***money*** as forbidden, cannot classify as spam indicator the variants ***m0ney*** or ***mo.ney***. And including all these possible variants in the blacklist yields to a very big list which would not be efficient.

Our aim is to build a Hidden Markov Model $(\underline{X}, \underline{Y}) = \{(X_t, Y_t)_{t\in\mathbf{N}}\}$, with its corresponding set of parameters $\lambda = (A, B, \underline{\pi})$, for each forbidden word of a reduced list, which can recognize all the possible modifications of the original one.

2.1. Set of observed states Y

The set of possible observations for any word is formed by all the letters, numbers and symbols of a keyboard.

In what follows, we will illustrate the parametrization of a forbidden word using as an example the word ***money***. In order to simplify the emission matrix, the symbols will be grouped. Thus, we consider the set $\boldsymbol{L}$ containing all the letters which do not appear in ***money***. The rest of possible symbols will be contained in the set $\boldsymbol{Q} = \{.,@,-,0,:,\&,\ldots\}$.

Last, the symbol '*' will model the loss of one letter respect from the term which is being adulterated. For instance, if the observed sequence is ***mney***, there is a missing letter (***o***) in the second position.

To sum up, the set of observed states of the model is $Y = \{\boldsymbol{m},\boldsymbol{o},\boldsymbol{n},\boldsymbol{e},\boldsymbol{y},\boldsymbol{L},\boldsymbol{Q},*\}$.

2.2. Set of hidden states X

The hidden states are defined as a consequence of the three possible incidences during the generation of an observed character, including the loss of this character, in comparison with the modelled word:

- Deletions
  They model the situation where there are some missing letters respect from the word which is being adulterated.

For example, if the observed sequence is ***mony***, there has been a loss of the letter '***e***'.

- Insertions
  When new characters are included in the word. This is the case of the character '.' in the observed sequence ***mone.y***, or one of the ***n***'s in the sequence ***monney***.
- Substitutions
  Some letters in the original word are replaced by other characters, for example, the number '***0***' replaces the letter '***o***' in ***m0ney***. We will use this kind of states (the match states) also for modelling the case in which the letters are the same as the original ones (all the states in the sequence ***money*** would be matched).

Let R be the length of the model, which represents the number of letters in the original word (in our case, $R = 5$). Let us consider the following graph (see Fig. 1) for explaining all the possible transitions (arcs of the graph) between the hidden states (nodes of the graph), divided into the following three types:

- Delete states (represented by circles): $d_1, d_2, \ldots, d_R$.
- Insert states (represented by diamonds): $i_0, i_1, i_2, \ldots, i_R$.
- Match states (represented by squares): $m_1, m_2, \ldots, m_R$.

The insert state i_0 is included for modelling those situations in which there is some insertion before the first letter of the original word.

Furthermore, two artificial states are added, source and sink respectively, at the beginning and the end of the graph (the nodes called *bb* and *ee*), which help to clarify the graph structure and simplify the set of parameters; state *ee* is used for simplifying the initial probability vector $\underline{\pi}$, as will be seen in (3), and state *ee* makes the transition matrix A be well-defined in the last rows and gives it stochasticity.

As can be observed in the graph, from a match state m_r or delete state d_r, one can continue to the insert state with the same index i_r or to the following match or delete states, m_{r+1} or d_{r+1}. On the other hand, from an insert state i_r, there are possible transitions to the following match or delete states, m_{r+1} or d_{r+1}, but also to the same state i_r, due to the possible insertion of several characters between two letters of the original word. Hence, the length of the adulterated sequence is not fixed (and it may be infinite theoretically, although it does not occur in this kind of models).

Each path with origin in *bb* and final in *ee* can be potentially a hidden sequence which models what is happening in the head of the spammer when he is writing an adulterated word. In fact, an observed sequence can be compatible with several of these paths, each one contributing with a determined likelihood for the observed sequence according to the set of parameters λ. Let us see some examples for the model ***money*** (we will consider some variants of the original term ***money*** and different hidden sequences associated with these variants, according to the graph of Fig. 1, will be studied).

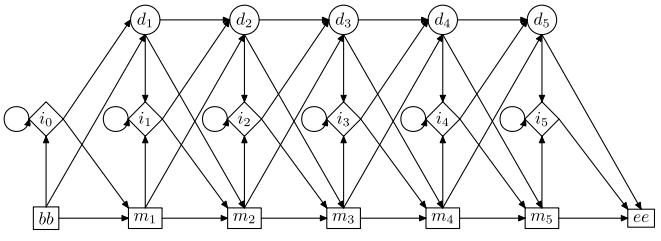

Fig. 1. Transitions between hidden states for a model with length $R = 5$.

Example 2.1. If the observed sequence is *m*0*ne..y*, the intuitively most likely hidden sequence and its corresponding emitted sequence are

Hidden sequence :	bb	m_1	m_2	m_3	m_4	i_4	i_4	m_5	ee
Emitted sequence :	$*$	***m***	***Q***	***n***	***e***	***Q***	***Q***	***y***	$*$

taking into account that 0 and . belong to the set Q.

However, another possibility, with different number of hidden states (later, the problem of choosing the suitable length of the hidden sequence will be explained) is

Hidden sequence :	bb	m_1	d_2	i_2	m_3	m_4	d_5	i_5	i_5	i_5	ee
Emitted sequence :	$*$	***m***	$*$	***Q***	***n***	***e***	$*$	***Q***	***Q***	***y***	$*$

which is also a feasible hidden sequence and, hence, must be considered.

Finally, the set of hidden states is $X = \{bb, i_0, d_1, i_1, m_1, \ldots, d_R, i_R, m_R, ee\}$. Thus, the number of hidden states is $3(R+1)$.

2.3. The set of parameters λ

In order to have a well-defined HMM, it is necessary to determine its corresponding set of parameters λ, formed by the transition matrix A, the emission matrix B and the initial probability vector $\underline{\pi}$.

Firstly, we will consider the transition matrix A, whose size is $3(R+1) \times 3(R+1)$. This matrix is quite sparse as a result of having a zero entry for each pair of hidden states which are not connected by an arc in the transition network (see Fig. 1).

The structure of the matrix is the following:

$$A = \begin{array}{c} \\ bb \\ i_0 \\ m_1 \\ d_1 \\ i_1 \\ \vdots \\ m_R \\ d_R \\ i_R \\ ee \end{array} \begin{array}{c} \begin{array}{ccccccccccc} bb & i_0 & m_1 & d_1 & i_1 & m_2 & d_2 & \ldots & i_R & ee \end{array} \\ \begin{pmatrix} 0 & a_{bbi_0} & a_{bbm_1} & a_{bbd_1} & 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & a_{i_0i_0} & a_{i_0m_1} & a_{i_0d_1} & 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{m_1i_1} & a_{m_1m_2} & a_{m_1d_2} & \ldots & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{d_1i_1} & a_{d_1m_2} & a_{d_1d_2} & \ldots & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{i_1i_1} & a_{i_1m_2} & a_{i_1d_2} & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & a_{m_Ri_R} & a_{m_Ree} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & a_{d_Ri_R} & a_{d_Ree} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & a_{i_Ri_R} & a_{i_Ree} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 1 \end{pmatrix} \end{array} \quad (1)$$

Since A is a stochastic matrix, one has that

$$\sum_{j\in X} a_{ij} = 1, \quad \forall i \in X$$

where a_{ij} are the elements of the matrix A.

Taking into account the matrix structure and the stochasticity assumption, the total number of parameters to be estimated is $4 + 6(R-1) + 3 = 6R + 1$.

For the sake of simplicity, we only consider the case in which there is not any extra information about the parameters. This information can be incorporated by including some restrictions over the parameters. For example, one can assign more likelihood to the element a_{bbm_1} than to the rest of the row because, generally, the observed sequences begin with the original letter, or likewise, the likelihood $a_{m_jm_{j+1}}$ (for any j) must be bigger than the rest of elements of the row (i.e., $a_{m_jm_{j+1}} \geqslant a_{m_ji_j}$ and $a_{m_jm_{j+1}} \geqslant a_{m_jd_{j+1}}$), because the most common is not to have any modification in the letter (match states are the most frequent).

Secondly, we will consider the emission probability matrix B. Let us remember there are $|Y|$ possible observations and the characters which did not appear in the original word (***money***) were grouped in the sets $\boldsymbol{L}$ (letters) and $\boldsymbol{Q}$ (rest of symbols). The symbol $*$ modelled the absence of observation in delete states, and likewise, this observed state is assigned to the artificial states bb and ee, because these ones do not emit any observation either.

Thus, we obtain the following emission matrix:

$$B = \begin{array}{c} \\ bb \\ i_0 \\ m_1 \\ d_1 \\ i_1 \\ \vdots \\ i_R \\ ee \end{array} \begin{array}{c} \begin{array}{cccccccc} \boldsymbol{m} & \boldsymbol{o} & \boldsymbol{n} & \boldsymbol{e} & \boldsymbol{y} & \boldsymbol{L} & \boldsymbol{Q} & * \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ b_{i_0}\boldsymbol{m} & b_{i_0}\boldsymbol{o} & b_{i_0}\boldsymbol{n} & b_{i_0}\boldsymbol{e} & b_{i_0}\boldsymbol{y} & b_{i_0}\boldsymbol{L} & b_{i_0}\boldsymbol{Q} & 0 \\ b_{m_1}\boldsymbol{m} & b_{m_1}\boldsymbol{o} & b_{m_1}\boldsymbol{n} & b_{m_1}\boldsymbol{e} & b_{m_1}\boldsymbol{y} & b_{m_1}\boldsymbol{L} & b_{m_1}\boldsymbol{Q} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ b_{i_1}\boldsymbol{m} & b_{i_1}\boldsymbol{o} & b_{i_1}\boldsymbol{n} & b_{i_1}\boldsymbol{e} & b_{i_1}\boldsymbol{y} & b_{i_1}\boldsymbol{L} & b_{i_1}\boldsymbol{Q} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{i_R}\boldsymbol{m} & b_{i_R}\boldsymbol{o} & b_{i_R}\boldsymbol{n} & b_{i_R}\boldsymbol{e} & b_{i_R}\boldsymbol{y} & b_{i_R}\boldsymbol{L} & b_{i_R}\boldsymbol{Q} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \tag{2}$$

Since the matrix B is also stochastic, the total number of parameters is $(2R+1)\times(|Y|-2)$.

Restrictions can be imposed over the elements of this matrix in the same way as for matrix A. For example, some linear restrictions can be added to assign more likelihood to $b_{m_1\boldsymbol{m}}$ than to the rest of the row (because the match states usually emit the same original letter), or likewise, the likelihood $b_{i_k\boldsymbol{Q}}$ must be bigger than the rest of the row (the insertions are usually symbols to make more understandable the word extracted from the message).

Finally, we will build the initial probability vector $\underline{\pi}$, whose structure is simple due to the definition of the artificial state bb, because all the hidden sequences have to start with this state, and hence, the vector concentrates all the probability on this initial state

$$\underline{\pi}^T = \begin{array}{c} \begin{array}{cccccc} bb & i_0 & m_1 & d_1 & \dots & ee \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \end{pmatrix} \end{array} \tag{3}$$

Defining the parameters (1)–(3), we have specified a model satisfying Assumptions 1–4.

As an example, let us consider the following matrices A and B for the model ***money***.

The non-trivial blocks of the matrix A are

$$A_1 = \begin{array}{c} \\ bb \\ i_0 \end{array} \begin{array}{c} \begin{array}{ccc} i_0 & m_1 & d_1 \end{array} \\ \begin{pmatrix} 0.05 & 0.9 & 0.05 \\ 0.2 & 0.75 & 0.05 \end{pmatrix} \end{array} \qquad A_2 = \begin{array}{c} \\ m_1 \\ d_1 \\ i_1 \end{array} \begin{array}{c} \begin{array}{ccc} i_1 & m_2 & d_2 \end{array} \\ \begin{pmatrix} 0.2 & 0.5 & 0.3 \\ 0.05 & 0.9 & 0.05 \\ 0.1 & 0.85 & 0.05 \end{pmatrix} \end{array}$$

$$A_3 = \begin{array}{c} \\ m_2 \\ d_2 \\ i_2 \end{array} \begin{array}{c} \begin{array}{ccc} i_2 & m_3 & d_3 \end{array} \\ \begin{pmatrix} 0.15 & 0.8 & 0.05 \\ 0.3 & 0.6 & 0.1 \\ 0.4 & 0.5 & 0.1 \end{pmatrix} \end{array} \qquad A_4 = \begin{array}{c} \\ m_3 \\ d_3 \\ i_3 \end{array} \begin{array}{c} \begin{array}{ccc} i_3 & m_4 & d_4 \end{array} \\ \begin{pmatrix} 0.15 & 0.6 & 0.25 \\ 0.25 & 0.7 & 0.05 \\ 0.4 & 0.4 & 0.2 \end{pmatrix} \end{array}$$

$$A_5 = \begin{array}{c} \\ m_4 \\ d_4 \\ i_4 \end{array} \begin{array}{c} \begin{array}{ccc} i_4 & m_5 & d_5 \end{array} \\ \begin{pmatrix} 0.35 & 0.55 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.3 & 0.6 & 0.1 \end{pmatrix} \end{array} \qquad A_6 = \begin{array}{c} \\ m_5 \\ d_5 \\ i_5 \end{array} \begin{array}{c} \begin{array}{cc} i_5 & ee \end{array} \\ \begin{pmatrix} 0.2 & 0.8 \\ 0.05 & 0.95 \\ 0.1 & 0.9 \end{pmatrix} \end{array}$$

The matrix B is the following:

$$B = \begin{array}{c} \\ bb \\ i_0 \\ m_1 \\ d_1 \\ i_1 \\ m_2 \\ d_2 \\ i_2 \\ m_3 \\ d_3 \\ i_3 \\ m_4 \\ d_4 \\ i_4 \\ m_5 \\ d_5 \\ i_5 \\ ee \end{array} \begin{array}{c} \begin{array}{cccccccc} \boldsymbol{m} & \boldsymbol{o} & \boldsymbol{n} & \boldsymbol{e} & \boldsymbol{y} & \boldsymbol{L} & \boldsymbol{Q} & * \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0.05 & 0.05 & 0 & 0 & 0 & 0.1 & 0.8 & 0 \\ 0.9 & 0.05 & 0 & 0 & 0 & 0 & 0.05 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0.2 & 0.1 & 0 & 0 & 0 & 0.1 & 0.6 & 0 \\ 0.05 & 0.6 & 0.05 & 0 & 0 & 0.1 & 0.2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0.05 & 0.15 & 0.05 & 0 & 0 & 0.3 & 0.45 & 0 \\ 0 & 0.05 & 0.75 & 0.05 & 0 & 0.1 & 0.05 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0.05 & 0.05 & 0 & 0.2 & 0.7 & 0 \\ 0 & 0 & 0.1 & 0.55 & 0.05 & 0.05 & 0.25 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0.05 & 0.05 & 0.1 & 0.8 & 0 \\ 0 & 0.05 & 0 & 0.05 & 0.75 & 0.05 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0.15 & 0.05 & 0.3 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Let us consider the observed sequence ***m0ne..y*** from Example 2.1, with the two feasible hidden sequences associated to the observed one in that case. Using the following notation:

$$\begin{array}{lllllllllll}
\underline{X}^{(1)} = & bb & m_1 & m_2 & m_3 & m_4 & i_4 & i_4 & m_5 & ee & \\
\underline{Y}^{(1)} = & * & m & Q & n & e & Q & Q & y & * & \\
\underline{X}^{(2)} = & bb & m_1 & d_2 & i_2 & m_3 & m_4 & d_5 & i_5 & i_5 & i_5 & ee \\
\underline{Y}^{(2)} = & * & m & * & Q & n & e & * & Q & Q & y & *
\end{array}$$

we will show the joint probability (hidden and emitted sequence) in the first case is bigger than in the second one.

So, for the first hidden sequence, one has that

$$\begin{aligned}
P_\lambda\left[\underline{X}^{(1)}, \underline{Y}^{(1)}\right] &= \pi_{bb} \cdot b_{bb*} \cdot a_{bbm_1} \cdot b_{m_1 m} \cdot a_{m_1 m_2} \cdot b_{m_2 Q} \cdot a_{m_2 m_3} \\
&\quad \cdot b_{m_3 n} \cdot a_{m_3 m_4} \cdot b_{m_4 e} \cdot a_{m_4 i_4} \cdot b_{i_4 Q} \cdot a_{i_4 i_4} \cdot b_{i_4 Q} \cdot a_{i_4 m_5} \\
&\quad \cdot b_{m_5 y} \cdot a_{m_5 ee} \cdot b_{ee*} \\
&= 1 \times 1 \times 0.9 \times 0.9 \times 0.5 \times 0.2 \times 0.8 \times 0.75 \\
&\quad \times 0.6 \times 0.55 \times 0.35 \times 0.8 \times 0.3 \times 0.8 \times 0.6 \\
&\quad \times 0.75 \times 0.8 \times 1 \\
&= 3.84 \times 10^{-4}
\end{aligned}$$

Whereas, for the second case,

$$\begin{aligned}
P_\lambda\left[\underline{X}^{(2)}, \underline{Y}^{(2)}\right] &= \pi_{bb} \cdot b_{bb*} \cdot a_{bbm_1} \cdot b_{m_1 m} \cdot a_{m_1 d_2} \cdot b_{d_2 *} \cdot a_{d_2 i_2} \cdot b_{i_2 Q} \\
&\quad \cdot a_{i_2 m_3} \cdot b_{m_3 n} \cdot a_{m_3 m_4} \cdot b_{m_4 e} \cdot a_{m_4 d_5} \cdot b_{d_5 *} \cdot a_{d_5 i_5} \\
&\quad \cdot b_{i_5 Q} \cdot a_{i_5 i_5} \cdot b_{i_5 Q} \cdot a_{i_5 i_5} \cdot b_{i_5 y} \cdot a_{i_5 ee} \cdot b_{ee*} \\
&= 1 \times 1 \times 0.9 \times 0.9 \times 0.3 \times 1 \times 0.3 \times 0.45 \\
&\quad \times 0.5 \times 0.75 \times 0.6 \times 0.55 \times 0.1 \times 1 \times 0.05 \\
&\quad \times 0.5 \times 0.1 \times 0.5 \times 0.1 \times 0.05 \times 0.9 \times 1 \\
&= 6.8505 \times 10^{-10}
\end{aligned}$$

Therefore, one has that $P_\lambda[\underline{X}^{(1)}, \underline{Y}^{(1)}] > P_\lambda[\underline{X}^{(2)}, \underline{Y}^{(2)}]$.

For calculating the probability of the sequence **m0ne..y** under the model originated by **money**, it is necessary to consider all the possible associated hidden sequences, to calculate their joint probability (hidden and emitted sequences) and add up all of them. However, some techniques are used to simplify this work (tools explained in the following sections).

In fact, we will study how to build the set of parameters of the model (Section 4), for later, given an observed sequence, classifying it (Section 5) as belonging or not belonging to the mentioned model (and hence, deciding if the sequence is a variant of the original term) by using the probability of the observed sequence under the corresponding model (Section 3).

3. Calculation of the probability of an observed sequence

In this section, we will study the problem of computing the probability of an observed sequence $\underline{Y}$ under a set of parameters λ.

Given an observed sequence $\underline{Y} = (y_1, \ldots, y_T)$, generated from a model λ with length R, we will denote S the length of the hidden sequence (without considering the artificial states, which will be denoted $X_0 = bb$ and $X_{S+1} = ee$). These three lengths may be different.

Fixed R, S, T, the number of each kind of states is the following:

- $D = S - T$ delete states.
- $I = S - R$ insert states.
- $M = R + T - S$ match states.

For the following analysis, it is necessary to know the length S previously. Obviously, in practice S is an unknown parameter, although one has that

- $R \leqslant S$, because there may be some insertions in the observed sequence $\underline{Y}$ respect from the original sequence.
- $T \leqslant S$, because there may be some deletions in the observed sequence.
- $D = S - T \leqslant R$, that is, the number of deletions cannot be bigger than the total length of the model.
- $I = S - R \leqslant T$, that is, the number of insertions cannot be bigger than the length of the observed sequence.

Consequently, $\max\{R, T\} \leqslant S \leqslant R + T$. Moreover, the value of S cannot be high if the observed sequence seeks to preserve its morphology in order to be understandable for any user.

Hence, S will be considered as a parameter, and one can solve the problem for different values of S, calculating the maximum likelihood of the observed sequence over the set of values of S. One can study all the values because there are only a few of them, contrary to the case of bionformatics, where long chains of nucleotids (or amino acids) were considered and, therefore, this method could not be an efficient way to solve the problem.

Another possibility would be to use the bayesian paradigm (see Berger, 1980; Bernardo & Smith, 1994; Leonard & Hsu, 2001), by giving an a priori distribution for the variable S: $P[S = s]$, with $\max\{R, T\} \leqslant s \leqslant R + T$. We calculate the probabilities of the observed sequence $\underline{Y}$ for the different values of S, $P[S = s|\underline{Y}]$, obtaining the a posteriori distribution for S, through Bayes' Theorem,

$$P[S = s^*|\underline{Y}] = \frac{P[\underline{Y}|S = s^*] \cdot P[S = s^*]}{\sum_{s=\max\{R,T\}}^{R+T} P[\underline{Y}|S = s] \cdot P[S = s]}$$

Finally, we take the a posteriori maximum likelihood value of S.

3.1. Modification of the forward algorithm

Given $\underline{Y} = (y_1, \ldots, y_T)$ an observed sequence and λ the set of parameters of the model, we have to calculate the probability of obtaining this observed sequence under this model, i.e., our problem is to calculate $P_\lambda[\underline{Y}]$.

A modification of the standard forward–backward algorithm (Algorithms 3.1 and 3.2), classically used in HMMs for these probability calculations (see MacDonald & Zucchini, 1997; Rabiner, 1989; Rabiner & Juang, 1993), is introduced for being applied to the case in which insertions and deletions are allowed.

Previously, it is necessary the following definition, which helps to define properly the variables in which the algorithm is based on.

Definition 3.1. $A_{s,t}$ is defined as the set of all the possible inclusions of $s-t$ missing observations, '*', between the first t observations $(y_1,\ldots,y_t)$ of the sequence $\underline{Y}$.

Observe that $\operatorname{card}(A_{s,t}) = \binom{s}{s-t}$.

Example 3.1. For $s=4$ and $t=2$, one has that

$$A_{4,2} = \{(y_1,y_2,*,*),(y_1,*,y_2,*),(y_1,*,*,y_2),\\ (*,y_1,y_2,*),(*,y_1,*,y_2),(*,*,y_1,y_2)\}$$

The necessary variables for the forward algorithm are defined in the following definition.

Definition 3.2. Given the observed sequence $\underline{Y}=(y_1,\ldots,y_T)$, and the model with set of parameters λ, the forward variables are defined as

$$\alpha_{s,t}(m_r,y_t) = P_\lambda[X_s = m_r,(Y_1,\ldots,Y_{s-1})\in A_{s-1,t-1},Y_s=y_t] \quad (4)$$

$$\alpha_{s,t}(i_r,y_t) = P_\lambda[X_s = i_r,(Y_1,\ldots,Y_{s-1})\in A_{s-1,t-1},Y_s=y_t] \quad (5)$$

$$\alpha_{s,t}(d_r,*) = P_\lambda[X_s = d_r,(Y_1,\ldots,Y_{s-1})\in A_{s-1,t},Y_s=*] \quad (6)$$

for $s=1,\ldots,S$, $t=0,\ldots,T$, $r=0,\ldots,R$.

These variables depend on the following three indexes which are used in the following way:

- The index s enumerates and orders the hidden sequence. Furthermore, this index will be used for performing the iteration step in Algorithms 3.1 and 3.2 (forward and backward algorithms with insertions and missing observations), which will be described later. This index varies from 1 to S.
- The index t points to the last element observed of the sequence $\underline{Y}$ (in the case of insert and match states, this observation is considered in the same time s). It varies from 1 to T, although the value $t=0$ is also allowed for the case of delete states, for modelling those sequences which start with one or several missing observations (omitted letters).
- The index r points to the stage that has been reached in the transition network (Fig. 1) for the hidden sequence. It varies from 1 to R, and the value $r=0$ is also allowed for insert states.

The following lemma will help us to understand the iteration step of Algorithm 3.1.

Lemma 3.1. *Let $\underline{Y}=(y_1,\ldots,y_T)$ be an observed sequence, and the set of parameters λ of a model with sets of states X, Y, and lengths R, S, T. Then, given the variables $\alpha_{s,t}(m_r,y_t)$, $\alpha_{s,t}(i_r,y_t)$, $\alpha_{s,t}(d_r,*)$, one has that*

$$\begin{aligned}\alpha_{s+1,t+1}(m_{r+1},y_{t+1}) = {} & [\alpha_{s,t}(m_r,y_t)\cdot a_{m_r,m_{r+1}}\\ & +\alpha_{s,t}(i_r,y_t)\cdot a_{i_r,m_{r+1}}\\ & +\alpha_{s,t}(d_r,*)\cdot a_{d_r,m_{r+1}}]\cdot b_{m_{r+1},y_{t+1}}\end{aligned} \quad (7)$$

$$\begin{aligned}\alpha_{s+1,t+1}(i_r,y_{t+1}) = {} & [\alpha_{s,t}(m_r,y_t)\cdot a_{m_r,i_r}+\alpha_{s,t}(i_r,y_t)\cdot a_{i_r,i_r}\\ & +\alpha_{s,t}(d_r,*)\cdot a_{d_r,i_r}]\cdot b_{i_r,y_{t+1}}\end{aligned} \quad (8)$$

$$\begin{aligned}\alpha_{s+1,t}(d_{r+1},*) = {} & \alpha_{s,t}(m_r,y_t)\cdot a_{m_r,d_{r+1}}+\alpha_{s,t}(i_r,y_t)\cdot a_{i_r,d_{r+1}}\\ & +\alpha_{s,t}(d_r,*)\cdot a_{d_r,d_{r+1}}\end{aligned} \quad (9)$$

Proof. See the Appendix. □

By using this lemma, we will obtain the following modification of the forward algorithm for insertions and missing observations, which allows us to calculate the probability $P_\lambda[\underline{Y}]$ (given the sequence $\underline{Y}$ and the model λ).

Algorithm 3.1 (Modified forward algorithm)

1. **Initialization step**
   - *Begin state* (bb)

     $\alpha_{0,0}(bb,*) := \pi_{bb}\cdot b_{bb,*} = 1$
   - *First variables*

     $\alpha_{1,1}(i_0,y_1) := a_{bb,i_0}\cdot b_{i_0,y_1}\cdot\alpha_{0,0}(bb,*) = a_{bb,i_0}\cdot b_{i_0,y_1}$

     And,

     $\alpha_{1,1}(m_1,y_1) := a_{bb,m_1}\cdot b_{m_1,y_1}$

     $\alpha_{1,0}(d_1,*) := a_{bb,d_1}$
2. **Iteration step**: $s=1,\ldots,(S-1)$.
   Given the variables $\alpha_{s,t}(m_r,y_t)$, $\alpha_{s,t}(i_r,y_t)$, $\alpha_{s,t}(d_r,*)$,
   - *Insert states*

     $$\begin{aligned}\alpha_{s+1,t+1}(i_r,y_{t+1}) := {} & [\alpha_{s,t}(m_r,y_t)\cdot a_{m_r,i_r}+\alpha_{s,t}(i_r,y_t)\cdot a_{i_r,i_r}\\ & +\alpha_{s,t}(d_r,*)\cdot a_{d_r,i_r}]\cdot b_{i_r,y_{t+1}}\end{aligned}$$

     for

     $1\leqslant s\leqslant S-1$

     $\max\{0,s-(S-T)\}\leqslant t\leqslant\min\{s,T-1\}$

     $\max\{s+1-(S-R),s-t\}\leqslant r$

     $\quad\leqslant\min\{s,R-(S-T)+(s-t)\}$
   - *Match states*

     $$\begin{aligned}\alpha_{s+1,t+1}(m_{r+1},y_{t+1}) := {} & [\alpha_{s,t}(m_r,y_t)\cdot a_{m_r,m_{r+1}}+\alpha_{s,t}(i_r,y_t)\\ & \cdot a_{i_r,m_{r+1}}+\alpha_{s,t}(d_r,*)\cdot a_{d_r,m_{r+1}}]\cdot b_{m_{r+1},y_{t+1}}\end{aligned}$$

     for

     $1\leqslant s\leqslant S-1$

     $\max\{0,s-(S-T)\}\leqslant t\leqslant\min\{s,T-1\}$

     $\max\{s-(S-R),s-t\}\leqslant r$

     $\quad\leqslant\min\{s,(R-1)-(S-T)+(s-t)\}$
   - *Delete states*

     $$\begin{aligned}\alpha_{s+1,t}(d_{r+1},*) := {} & \alpha_{s,t}(m_r,y_t)\cdot a_{m_r,d_{r+1}}+\alpha_{s,t}(i_r,y_t)\cdot a_{i_r,d_{r+1}}\\ & +\alpha_{s,t}(d_r,*)\cdot a_{d_r,d_{r+1}}\end{aligned}$$

for

$$1 \leqslant s \leqslant S-1$$
$$\max\{0,(s+1)-(S-T)\} \leqslant t \leqslant \min\{s,T\}$$
$$\max\{s-(S-R),s-t\} \leqslant r \leqslant \min\{s,R-(S-T)+(s-t)\}$$

3. **Output step**
   - *End state* (*ee*)
     The required probability is calculated as follows:

$$\alpha_{S+1,T}(ee,*) := P_\lambda[\underline{Y}] = \alpha_{S,T}(m_R,y_T)\cdot a_{m_R,ee} + \alpha_{S,T}(i_R,y_T)\cdot a_{i_R,ee} + \alpha_{S,T}(d_R,*)\cdot a_{d_R,ee}$$

This algorithm reduces the computational complexity of the original problem. Given $\underline{Y}=(y_1,\ldots,y_T)$ and fixed S, if all the possible hidden sequences, according to the graph, are considered to compute the probability of the observed sequence (by summing over all the joint probabilities $P[\underline{Y},\underline{X}^{(l)}]$), this calculation requires, at least $PR_S^{D,I,M}=\frac{S!}{D!I!M!}$ operations, which is the total number of paths. However, the number of operations, by means of this algorithm, is much smaller, as shown in the following result.

Property 3.1. *Given the observed sequence* $\underline{Y}=(y_1,\ldots,y_T)$, *and fixed* S *the length of the hidden sequence, the number of operations of this algorithm is* $\mathcal{O}(S\cdot\min\{T,D\}\cdot\min\{I,M\})$, *where* D, I *and* M *are, respectively, the number of delete, insert and match states in the hidden sequence.*

Proof. See the Appendix. □

3.2. Modification of the backward algorithm

Given an observed sequence $\underline{Y}=(y_1,\ldots,y_T)$ and the model λ, the backward algorithm (modified for insertions and missing observations) also solves the problem of the calculation of $P_\lambda[\underline{Y}]$, having the same computational complexity, but moreover, it adds the backward variables, which will be necessary in Algorithm 4.1 (a modification of the classical Baum–Welch algorithm), when we approach the learning problem.

Definition 3.3. $B_{s,t}$ is defined as the set of all the possible inclusions of $(S-s)-(T-t)$ missing observations, '*', between the last $T-(t-1)$ observations $(y_t,\ldots,y_T)$ of the sequence $\underline{Y}$.

$$\text{Observe that } \operatorname{card}(B_{s,t}) = \binom{S-(s-1)}{(S-s)-(T-t)}.$$

Definition 3.4. Given the observed sequence $\underline{Y}=(y_1,\ldots,y_T)$ and the set of parameters λ, the backward variables are defined as

$$\beta_{s,t}(m_r,y_t) = P_\lambda[(Y_{s+1},\ldots,Y_S)\in B_{s+1,t+1}|X_s=m_r] \quad (10)$$
$$\beta_{s,t}(i_r,y_t) = P_\lambda[(Y_{s+1},\ldots,Y_S)\in B_{s+1,t+1}|X_s=i_r] \quad (11)$$
$$\beta_{s,t}(d_r,*) = P_\lambda[(Y_{s+1},\ldots,Y_S)\in B_{s+1,t}|X_s=d_r] \quad (12)$$

for $s=1,\ldots,S$, $t=0,\ldots,T$, $r=0,\ldots,R$.

Before describing the modification for the backward algorithm (for insertions and missing observations), we need a previous lemma, analogous to Lemma 3.1 in the case of the forward variables.

Lemma 3.2. *Let* $\underline{Y}=(y_1,\ldots,y_T)$ *be an observed sequence, and* λ *the set of parameters of a model with sets of states* X, Y, *and lengths* R, S, T. *Then, given the variables* $\beta_{s+1,t+1}(i_r,y_{t+1})$, $\beta_{s+1,t+1}(m_{r+1},y_{t+1})$ *and* $\beta_{s+1,t+1}(d_{r+1},*)$, *one has that*

$$\beta_{s,t}(i_r,y_t) = a_{i_r,i_r}\cdot b_{i_r,y_{t+1}}\cdot\beta_{s+1,t+1}(i_r,y_{t+1}) + a_{i_r,m_{r+1}}\cdot b_{m_{r+1},y_{t+1}}\cdot\beta_{s+1,t+1}(m_{r+1},y_{t+1}) + a_{i_r,d_{r+1}}\cdot b_{d_{r+1},*}\cdot\beta_{s+1,t+1}(d_{r+1},*) \quad (13)$$

$$\beta_{s,t}(m_r,y_t) = a_{m_r,i_r}\cdot b_{i_r,y_{t+1}}\cdot\beta_{s+1,t+1}(i_r,y_{t+1}) + a_{m_r,m_{r+1}}\cdot b_{m_{r+1},y_{t+1}}\cdot\beta_{s+1,t+1}(m_{r+1},y_{t+1}) + a_{m_r,d_{r+1}}\cdot b_{d_{r+1},*}\cdot\beta_{s+1,t+1}(d_{r+1},*) \quad (14)$$

$$\beta_{s,t+1}(d_r,*) = a_{d_r,i_r}\cdot b_{i_r,y_{t+1}}\cdot\beta_{s+1,t+1}(i_r,y_{t+1}) + a_{d_r,m_{r+1}}\cdot b_{m_{r+1},y_{t+1}}\cdot\beta_{s+1,t+1}(m_{r+1},y_{t+1}) + a_{d_r,d_{r+1}}\cdot b_{d_{r+1},*}\cdot\beta_{s+1,t+1}(d_{r+1},*) \quad (15)$$

Proof. Analogously to Lemma 3.1. □

This lemma helps to describe the following modification of the backward algorithm for insertions and missing observations.

Algorithm 3.2 (Modified backward algorithm)

1. **Initialization step**
   - *End state* (*ee*)

$$\beta_{S+1,T+1}(ee,*) := 1$$

   - *First variables*

$$\beta_{S,T}(m_R,y_T) := a_{m_R,ee}\cdot b_{ee,*} = a_{m_R,ee}$$

   And,

$$\beta_{S,T}(i_R,y_T) := a_{i_R,ee}$$
$$\beta_{S,T+1}(d_R,*) := a_{d_R,ee}$$

2. **Iteration step**: $s=(S-1),\ldots,1$. Given the variables $\beta_{s+1,t+1}(i_r,y_{t+1})$, $\beta_{s+1,t+1}(m_{r+1},y_{t+1})$, $\beta_{s+1,t+1}(d_{r+1},*)$,
   - *Insert states*

$$\beta_{s,t}(i_r,y_t) := a_{i_r,i_r}\cdot b_{i_r,y_{t+1}}\cdot\beta_{s+1,t+1}(i_r,y_{t+1}) + a_{i_r,m_{r+1}}\cdot b_{m_{r+1},y_{t+1}}\cdot\beta_{s+1,t+1}(m_{r+1},y_{t+1}) + a_{i_r,d_{r+1}}\cdot b_{d_{r+1},*}\cdot\beta_{s+1,t+1}(d_{r+1},*)$$

   for

$$1 \leqslant s \leqslant S-1$$
$$\max\{1,s-(S-T)\} \leqslant t \leqslant \min\{s,T\}$$
$$\max\{s-(S-R),s-t\} \leqslant r \leqslant \min\{s-1,R-(S-T)+(s-t)\}$$

- *Match states*

$$\beta_{s,t}(m_r, y_t) := a_{m_r,i_r} \cdot b_{i_r,y_{t+1}} \cdot \beta_{s+1,t+1}(i_r, y_{t+1}) + a_{m_r,m_{r+1}} \cdot b_{m_{r+1},y_{t+1}} \cdot \beta_{s+1,t+1}(m_{r+1}, y_{t+1}) + a_{m_r,d_{r+1}} \cdot b_{d_{r+1},*} \cdot \beta_{s+1,t+1}(d_{r+1}, *)$$

for

$$1 \leqslant s \leqslant S-1$$
$$\max\{1, s-(S-T)\} \leqslant t \leqslant \min\{s, T\}$$
$$\max\{s-(S-R), s-t+1\} \leqslant r \leqslant \min\{s, R-(S-T)+(s-t)\}$$

- *Delete states*

$$\beta_{s,t+1}(d_r, *) := a_{d_r,i_r} \cdot b_{i_r,y_{t+1}} \cdot \beta_{s+1,t+1}(i_r, y_{t+1}) + a_{d_r,m_{r+1}} \cdot b_{m_{r+1},y_{t+1}} \cdot \beta_{s+1,t+1}(m_{r+1}, y_{t+1}) + a_{d_r,d_{r+1}} \cdot b_{d_{r+1},*} \cdot \beta_{s+1,t+1}(d_{r+1}, *)$$

for

$$1 \leqslant s \leqslant S-1$$
$$\max\{0, s-(S-T)\} \leqslant t \leqslant \min\{s-1, T\}$$
$$\max\{s-(S-R), s-t\} \leqslant r \leqslant \min\{s, R-(S-T)+(s-t)\}$$

3. **Output step**
   - *Begin state* (*bb*)
     The probability of the sequence $\underline{Y}$ can be calculated as follows:

$$\beta_{0,1}(bb, *) := P_\lambda[\underline{Y}|X_0 = bb] = P_\lambda[\underline{Y}] = a_{bb,i_0} \cdot b_{i_0,y_1} \cdot \beta_{1,1}(i_0, y_1) + a_{bb,m_1} \cdot b_{m_1,y_1} \cdot \beta_{1,1}(m_1, y_1) + a_{bb,d_1} \cdot b_{d_1,*} \cdot \beta_{1,1}(d_1, *)$$

The number of operations of this algorithm is the same as the forward algorithm's one.

Property 3.2. *The computational complexity of this algorithm is* $\mathcal{O}(S \cdot \min\{T, D\} \cdot \min\{I, M\})$.

Proof. Analogously to Property 3.1. □

As we have said, apart from the calculation of $P_\lambda[\underline{Y}]$, the computed variables can be used in the problem of learning the parameters of a model. This will be analyzed in the following section.

4. Maximum likelihood estimation of the parameters. The learning problem

The learning of the model is the method for parametrizing a forbidden word of the blacklist, by using for it a list with several morphological variants of this word, detected in a set of spam e-mails. Each forbidden word (each model) has a set of parameters assigned λ, which will be estimated by the *maximum likelihood method*, under the assumptions of an HMM.

Firstly, we will describe how to estimate λ by using only an observed sequence (variant of the forbidden word) for extending, later, the method to the case of having several observed sequences (a list of variants).

The election of the maximum likelihood estimator of the parameters vector λ is obtained by solving the following optimization problem:

$$\begin{aligned} \max_{\lambda=(A,B,\underline{\pi})} \quad & P_\lambda[Y_1 = y_1, \ldots, Y_T = y_T] \\ \text{s.t.} \quad & \sum_{j \in X} a_{ij} = 1, \quad i \in X \\ & \sum_{k \in Y} b_{jk} = 1, \quad j \in X \\ & \sum_{i \in X} \pi_i = 1 \\ & a_{ij}, b_{jk}, \pi_i \geqslant 0, \quad \forall i, j, k \end{aligned} \tag{16}$$

It is a global optimization problem, because the objective function is not concave, in general. The standard methodology used for HMMs obtains an approach solution of the optimization problem (16) by means of a local search method called *Baum–Welch Algorithm* (see Baum, Petrie, Soules, & Weiss, 1970; MacDonald & Zucchini, 1997; Rabiner, 1989; Rabiner & Juang, 1993).

Basically, the Baum–Welch method lies in improving the objective value of a set of parameters by using an auxiliary convex optimization problem. By applying iteratively this method, we can assert that the algorithm converges to a local optimum of the optimization problem (16).

The auxiliary problem which is solved in each iteration (see Baum et al., 1970) has the following concave objective function:

$$\lambda \hookrightarrow Q(\hat{\lambda}, \lambda) = \sum_{\underline{X} \in X^T} P_{\hat{\lambda}}[\underline{Y}, \underline{X}] \cdot \log P_\lambda[\underline{Y}, \underline{X}] \tag{17}$$

By using Jensen's inequality, one has that

Property 4.1 (Baum et al. (1970)). *If* $Q(\hat{\lambda}, \lambda) \geqslant Q(\hat{\lambda}, \hat{\lambda})$, *then* $P_\lambda[\underline{Y}] \geqslant P_{\hat{\lambda}}[\underline{Y}]$. *Moreover, one has that*

$$\left.\frac{\partial P_\lambda[\underline{Y}]}{\partial \lambda_i}\right|_{\hat{\lambda}} = \left.\frac{\partial Q(\hat{\lambda}, \lambda)}{\partial \lambda_i}\right|_{\lambda=\hat{\lambda}}, \quad \forall i$$

with λ_i *the components of* λ.

Therefore, by changing the objective function of the problem (16) by the auxiliary function $Q(\hat{\lambda}, \lambda)$, concave in λ, where $\hat{\lambda}$ is the best set of parameters obtained so far, and by solving this auxiliary problem, a solution improving $\hat{\lambda}$ will be obtained, except when $\hat{\lambda}$ is a local optimum of problem (16) (see Baum et al., 1970). This algorithm can be seen as an implementation of the standard algorithm EM (Expectation–Maximization), (see Dempster, Laird, & Rubin, 1977; Redner & Walker, 1984; Wu, 1983), whose *l*th iteration is

- **Step E.** Calculating the auxiliary function $Q(\lambda_{l-1}, \lambda_l)$, where λ_{l-1} is the model obtained in the previous iteration and

$$Q(\lambda_{l-1},\lambda_l)=\sum_{\underline{X}\in X^{\mathrm{T}}} P_{\lambda_{l-1}}[\underline{Y},\underline{X}]\cdot\log P_{\lambda_l}[\underline{Y},\underline{X}]$$

- **Step M.** Maximizing in λ_l the function $Q(\lambda_{l-1},\lambda_l)$, verifying
  $Q(\lambda_{l-1},\lambda_l)\geqslant Q(\lambda_{l-1},\lambda),\quad\forall\lambda$
  for any other set of parameters λ, obtaining the parameters $\pi_i^{(l)}$, $a_{ij}^{(l)}$, $b_{jk}^{(l)}$ of the new model.
- **Stop test.** Stopping if the improvement in the auxiliary function (17) is smaller than a previously fixed threshold, that is,
  if $Q(\lambda_l,\lambda_{l+1})-Q(\lambda_l,\lambda_l)<\epsilon,\quad$ with $\epsilon>0$ fixed
  Otherwise, returning to step **E**.

In the particular case of Profile HMM, the vector $\underline{\pi}$ must not be considered as a parameter because, according to (3), this vector concentrates all the probability on the source state *bb*. Moreover, the matrices A and B must satisfy some restrictions, described in (1) and (2), respectively.

Therefore, the optimization problem to solve is

$$\begin{aligned}\max_{\lambda=(A,B)}\quad & P_\lambda[Y_1=y_1,\ldots,Y_T=y_T]\\ \text{s.t.}\quad & \sum_{j\in X}a_{ij}=1,\quad i\in X\\ & \sum_{k\in Y}b_{jk}=1,\quad j\in X\\ & a_{ij},b_{jk}\geqslant 0,\quad\forall i,j,k,\\ & A\text{ satisfying (1)}\\ & B\text{ satisfying (2)}\end{aligned}\tag{18}$$

Firstly, we will describe how to optimize the auxiliary problem in which is based the **Step M** of the algorithm. In the standard methodology of HMMs, it is proved that the only critical point of the function $Q(\hat{\lambda},\lambda)$ satisfies the restrictions of the problem (16). The function $Q(\hat{\lambda},\lambda)$ is differentiable in λ and its critical point can be expressed in terms of the variables obtained for describing the forward–backward algorithm (Algorithm 3.1 and 3.2). Hence, the optimization problem described for **Step M** is reduced to the obtention of these variables.

In the model proposed herein, the solution of the optimization problem (18) will be also obtained in terms of the forward and backward variables, but it will be necessary to define several families of auxiliary variables, based on the previous ones. This technical task is described in the Appendix where the sets of variables $\delta_{s,t}(i,j)$, $\xi_s(i,j)$, $\eta_{s,t}(i,j)$ and $\gamma_s(i)$ are defined.

The modification of the Baum–Welch algorithm for allowing insertions and missing observations in the observed sequence can be stated as follows.

Algorithm 4.1 (Modified Baum–Welch algorithm)

1. Initialization.
   Let $\underline{\pi}_0$ be as described in (3) and $\lambda_0=(A_0,B_0,\underline{\pi}_0)$, where the pair (A_0,B_0) is the initial seed for the parameters which must be estimated, according to the problem (18). We do $\hat{\lambda}:=\lambda_0$.
2. Calculation of variables.
   Given the set of parameters $\hat{\lambda}$, we calculate the forward and backward variables (by using Lemma 3.1 and 3.2) and the variables $\delta_{s,t}(i,j)$, $\xi_s(i,j)$, $\eta_{s,t}(i,j)$ and $\gamma_s(i)$ (by the results established in the Appendix section).
3. Determination of the new model.
   The new set of parameters $\lambda=(A,B,\pi_0)$ is built as follows:
$$a_{ij}:=\frac{\sum_{s=0}^{S}P_\lambda[\underline{Y},X_s=i,X_{s+1}=j]}{\sum_{s=0}^{S}P_\lambda[\underline{Y},X_s=i]}=\frac{\sum_{s=0}^{S}\xi_s(i,j)}{\sum_{s=0}^{S}\gamma_s(i)},\quad i,j\in X\tag{19}$$
$$b_{jk}:=\frac{\sum_{t:y_t=k}\sum_{s=0}^{S}\eta_{s,t}(j)}{\sum_{s=0}^{S}\gamma_s(j)},\quad j\in X,\ k\in Y\tag{20}$$
4. Finalization.
   The algorithm is interrupted when the difference between the objective function of the auxiliary problem for the new set of parameters and the latter is smaller than a threshold ϵ (fixed previously), that is,
   $Q(\lambda,\hat{\lambda})-Q(\hat{\lambda},\hat{\lambda})<\epsilon.$

This algorithm verifies the same properties, respect from the problem (18), than the standard Baum–Welch algorithm respect from the problem (16), that is, if the initial seed is feasible and the threshold is chosen $\epsilon=0$, the algorithm converges to a local optimum of the problem (18).

Property 4.2. *Let $\underline{\pi}_0$ be as described in (3) and $\lambda_0=(A_0,B_0,\underline{\pi}_0)$ feasible for the problem (18), then, for every new iteration of Algorithm 4.1, a new feasible solution of the problem (18) is generated, and the sequence of solutions converges to a local optimum of the optimization problem.*

Proof. In order to show this result, it is sufficient to analyze the indexes r, s, t for which the variables used in expressions (19) and (20) are well-defined. Thus, we observe that the structure described in (1) and (2) for the transition and emission matrices, respectively, is conserved. Hence, the critical point in λ of the function $Q(\lambda,\hat{\lambda})$ will be feasible for the problem (18), if $\hat{\lambda}$ is also feasible.

By using Property 4.1 (see Baum et al., 1970), one has that the sequence of feasible solutions obtained with the algorithm converges to a critical point of the problem (18). □

The practical applicability of Algorithm 4.1 comes supported by the computational efficiency, as will be seen in the following proposition.

Property 4.3. *Each iteration of the modified Baum–Welch algorithm requires $\mathcal{O}(S\cdot\min\{T,D\}\cdot R\cdot|Y|)$ operations, where S is the length of the hidden sequence, T is the length of the observed sequence, D is the number of delete states, R*

is the length of the model and $|Y|$ is the cardinal of the set of observed states.

Proof. In each iteration, we must calculate the variables $\alpha_{s,t}(i,y_t)$, $\beta_{s,t}(i,y_t)$, $\delta_{s,t}(i,j)$, $\xi_s i,j$, $\eta_{s,t}(i)$ and $\gamma_s(i)$, all of them with complexity $\mathcal{O}(S\cdot\min\{T,D\}\cdot\min\{I,M\})$.

We also have to calculate the $6R+1$ parameters of the transition matrix, a_{ij}, each one with a number of operations equal to $\mathcal{O}(S)$. Thus, the total number of operations is $\mathcal{O}(S\cdot R)$.

Furthermore, there are $(2R+1)\times(|Y|-2)$ parameters of the emission matrix, b_{jk}, to be calculated. For each one, the number of operations is $\mathcal{O}(S\cdot\min\{T,D\})$, then, the total number is $\mathcal{O}(S\cdot\min\{T,D\}\cdot R\cdot|Y|)$.

Since $\min\{I,M\}\leqslant R$ (the number of match states cannot be bigger than the total length of the model), the computational complexity for each iteration is $\mathcal{O}(S\cdot\min\{T,D\}\cdot R\cdot|Y|)$. □

Finally, we consider the learning problem when there is a list of sequences of characters $\{\underline{Y}^{(1)},\ldots,\underline{Y}^{(H)}\}$ which represent variants of the word whose model is being parametrized. These sequences are supposed to be independent.

The optimization problem to solve, in this case, is

$$\begin{aligned}\max_{\lambda=(A,B)}\quad & \prod_{h=1}^{H}P_\lambda[\underline{Y}^{(h)}=(y_1^h,\ldots,y_{t_h}^h)]\\ \text{s.t.}\quad & \sum_{j\in X}a_{ij}=1,\quad i\in X\\ & \sum_{k\in Y}b_{jk}=1,\quad j\in X\\ & a_{ij},b_{jk}\geqslant 0,\quad \forall i,j,k\\ & A\text{ satisfying (1)}\\ & B\text{ satisfying (2)}\end{aligned}\tag{21}$$

which is also a global optimization problem (the objective function is not concave, in general).

The change in the parameters of the model by means of Baum–Welch algorithm is performed in a similar way (see Levinson, Rabiner, & Sondhi, 1983; Li, Parizeau, & Plamondon, 2000 for the standard case).

Let $\{\underline{Y}^{(1)},\ldots,\underline{Y}^{(H)}\}$ be a set of H observed sequences, mutually independent, with $Y^{(h)}=(y_1^h,\ldots,y_{T_h}^h)$, whose lengths $T_1,\ldots,T_H$ (and analogously for $S_1,\ldots,S_H$) can be different. We have to calculate the variables $\alpha_{s,t}^h(i,y_t)$, $\beta_{s,t}(i,y_t)$, $\delta_{s,t}(i,j)$, $\xi_s(i,j)$, $\eta_{s,t}(i)$ and $\gamma_{s,t}(i)$ (by the results established in the Appendix section) for $h=1,\ldots,H$. And the formulae for obtaining the parameters of the new model in each new iteration are

$$a_{ij}:=\frac{\sum_{h=1}^{H}\sum_{s=0}^{S_h}\xi_s^h(i,j)}{\sum_{h=1}^{H}\sum_{s=0}^{S_H}\gamma_s^h(i)},\quad i,j\in X\tag{22}$$

$$b_{jk}:=\frac{\sum_{h=1}^{H}\sum_{t:0\leqslant t\leqslant T_h;y_t^h=k}\sum_{s=0}^{S_h}\eta_{s,t}^h(j)}{\sum_{h=1}^{H}\sum_{s=0}^{S_h}\gamma_s^h(j)},\quad j\in X,\ k\in Y\tag{23}$$

5. The classification problem

After having built a model with its corresponding set of parameters λ for each word of the list of forbidden terms, our target problem will be to classify a message as spam mail or no-spam, according to the number of words, stemming from a forbidden term with its corresponding set of parameters assigned, appearing in the text.

Let $\lambda_1,\ldots,\lambda_L$ be a list of models, learnt by means of the modified Baum–Welch algorithm for several sequences, expressions (22) and (23), for each one of the L elements which form the list of forbidden words that we use.

Let $Y=(y_1,\ldots,y_T)$ be an observed sequence (a word of the text to be classified). For discovering if this sequence stems from one of the models (it is a variant of a forbidden word), we calculate, by means of the modified forward and backward algorithms (Algorithms 3.1 and 3.2), the probability of being emitted from the model λ_l (this probability is denoted q_l), that is,

$$q_l:=P_{\lambda_l}[Y_1,\ldots,Y_T]\tag{24}$$

for $l=1,\ldots,L$.

Thresholds $c_1,\ldots,c_L$ are fixed for the models, where each c_l is associated to the model (forbidden word) with set of parameters λ_l and it represents the minimum likelihood required to an observed sequence for being assigned to this model, or equivalently,

$$\text{If } q_l=P_{\lambda_l}[Y_1,\ldots,Y_T]>c_l,\quad \text{then } \underline{Y}\hookrightarrow\lambda_l\tag{25}$$

Then, the classification rule will be the following:

- Classify $\underline{Y}\hookrightarrow$ forbidden, if $\exists l: q_l>c_l$
- Classify $\underline{Y}\hookrightarrow$ no-forbidden, otherwise

Finally, a message will be classified as spam if the proportion of words classified as forbidden (respect from the total number of words in the message) is bigger than or equal to a fixed amount K. Otherwise, the message will be ham mail.

The parameters $c_1,\ldots,c_L$ and K will depend on the experience, but also on the decision of the user by fixing the misclassification costs (the smaller the parameters are, the bigger is the amount of spam messages detected, but the filter may be also removing, with higher probability, no-spam messages, due to the decreasing in flexibility).

6. Experimental study

Modified Forward–Backward Algorithm and Modified Baum–Welch Algorithm (Algorithms 3.1, 3.2 and 4.1) have been implemented by using the programming language *C#* (C sharp) on a computer with Pentium©4 CPU 3'06 GHz.

The computational effort that must be spent in order to implement our methodology has two clearly distinguishable phases.

The first one consists in the parametrization of each word of the blacklist along with the computation of

the thresholds by means of suitable learning samples. This phase is expensive from a computational viewpoint because, in particular, it needs the execution of the Baum–Welch learning algorithm. However, these calculations must be executed only once in order to initialize the method, or eventually when one wants to update the words of the blacklist or their learning samples.

Once that this phase has been done, the second phase consists in the regular use of the classification methodology, that is, using the Modified Forward–Backward Algorithm to evaluate the likelihood of a word, extracted from the e-mail, of being an obfuscated version of one of the words of the blacklist. The theoretical complexity of this algorithm was established in the Properties 3.1 and 3.2. In practice, we have checked the fast answer given by the algorithm, which can be explained by the short length of the observed sequences that must be handled (words). Experimentally, we have obtained that the evaluation of a sequence of 4–8 characters under a given model from the black list needs a mean of 5×10^{-4} s of CPU time. This means that one can check if a given sequence is, or not, an obfuscated version of, at least, one of the members of a blacklist with 100 words in a half of a tenth of a CPU second. Note that, according to what has been stated in the previous sections, one of the advantages of our methodology, is that the length of the blacklist need not to be too large, so that, a blacklist of 100 words should suffice to provide an operative system.

given word. We have considered the following four forbidden words belonging to a blacklist (*dollar*, *free*, *money* and *viagra*) and a word non-included in those lists (*monday*). By means of the modified Baum–Welch algorithm, their corresponding sets of parameters (for each one of the models) have been estimated, by using a set of fifteen sequences, variants of the original forbidden word, found in spam e-mails (for instance, variants as *d0llar* or *do11ar* have been used in the training of the model for *dollar*).

Moreover, since Baum–Welch algorithm is a local search method, a multistart technique has also been implemented for trying to avoid local optima, that is, for each model, a number of random initial seeds (matrices A_0 and B_0) is considered for doing the training of the model.

In the following table, we show the effects of a good choice of the number of initial seeds on the results of the learning problem. For the model *dollar*, an increasing number of random initial seeds is considered from 1 to 5000. We have studied the value of the objective function in each case, where the objective function is $P_\lambda[\underline{Y}^{(1)}, \ldots, \underline{Y}^{(15)}]$, along with the improvement (absolute and relative) produced in that value, where the absolute improvement is the difference between two consecutive values of the first row (which represents the values of the objective function), and for obtaining the relative improvement, one has to divide the absolute improvement by the value of the objective function for the former number of seeds, and to multiply by 100.

	1	10	30	50	100
Objective function	3.87E−53	3.65E−39	1.61E−37	1.80E−37	1.82E−37
Absolute improvement		3.65E−39	1.57E−37	1.87E−38	2.26E−39
Relative improvement		9.41E+15	4.31E+03	1.16E+01	1.26E+00
P[dollar]	0.038145	0.110032	0.082920	0.087903	0.092060
P[do11ar]	0.000602	0.051416	0.054038	0.051681	0.043788
P[d0llar]	0.002729	0.018483	0.016457	0.016614	0.021655
P[dotar]	5.57E−07	1.19E−08	4.34E−09	4.09E−09	6.38E−09
CPU time (s)	0.16	0.75	1.88	3.09	5.81
	250	500	1000	3000	5000
Objective function	3.48E−37	6.07E−36	7.62E−36	1.29E−35	2.02E−35
Absolute improvement	1.66E−37	5.72E−36	1.54E−36	5.27E−36	7.33E−36
Relative improvement	9.13E+01	1.64E+03	2.54E+01	6.92E+01	5.69E+01
P[dollar]	0.110696	0.109078	0.125112	0.126896	0.109700
P[do11ar]	0.034368	0.058246	0.058066	0.058632	0.061539
P[d0llar]	0.022947	0.019382	0.019362	0.024698	0.027134
P[dotar]	3.69E−09	1.98E−09	1.87E−09	1.62E−10	4.61E−10
CPU time (s)	14.14	27.92	55.59	167.16	276.08

Since the regular procedure of detecting obfuscated versions of a given blacklist over the e-mail text can be done efficiently, we will concentrate, in what follows, on illustrating the numerical behavior of the parametrization of a

Furthermore, the table also shows the probabilities, under the set of parameters built with each number of seeds, for the observed sequence *dollar* (the word that defines the model), two variants of the original word, *d0llar*

and *do11ar* (where the first one belongs to the training set, but not the second one), and a word that is quite similar, but does not belong to the model (not recognizable by a reader as dollar), *dotar*. These probabilities have been calculated by means of the modified forward–backward algorithm.

One can remark that, in general, the bigger the number of seeds is, the higher the probabilities for the sequences belonging to the model are, and the lower the probability of the sequence not belonging to the model. This can also be observed in Fig. 2, which represent the moving average (for the three last values of the number of seeds) for the probabilities of the four observed sequences considered.

Finally, the CPU time is also studied. As it was foreseeable, the increasing of the CPU time is linear as a function of the number of seeds considered.

Since the best improvement in the objective function (and in the probabilities of the sequences to classify) is produced at the beginning of the table, that is, when the number of seeds is quite small, we conclude that a big number of seeds is not necessary for training the model (we fix this value to 200 seeds for the following studies) because the improvement obtained would not be too significant and, however, the CPU time gets worse.

In the following table, we study the number of sequences of the training set necessary to obtain good results in the classification problems. From the set of parameters trained for the model *money* (with a training set of 15 sequences and 200 initial seeds), a different number of sequences belonging to this model have been simulated (by using MATLAB, which is also used in this experimental study). With these new training sets, we build new sets of parameters and we compute the probabilities of four observed sequences under these parameters: the original word (*money*), a variant (*m0ney*) used for training the first set of parameters, another variant (*mon3y*) not used in the original training set, and a word (*monday*) quite similar, but not belonging to the model.

	3	6	9	15	20	25	50
P[money]	0.080647	0.048479	0.190496	0.101048	0.133171	0.058925	0.096954
P[m0ney]	0.069316	0.009267	0.123492	0.002959	0.048269	0.022358	0.057877
P[mon3y]	0	1.70E−07	1.86E−05	0.002047	0.012631	0.001846	0.00091
P[monday]	0	0	1.23E−05	2.54E−08	6.33E−06	5.48E−06	1.19E−06

As can be seen, when the number of sequences is too small, the model only recognizes the sequences of the training set, but not too many sequences are necessary to build the model, since a model built with 15 or 20 sequences gets to discriminate correctly between sequences belonging and not belonging to the model (it is sufficient to remark the probabilities for the sequences in the table).

Finally, we consider the problem of classifying a set of observed sequences as belonging (or not) to the model *money*. Initially, a training set of 500 sequences is considered, 100 extracted from the set of parameters for *money* (via simulation) and the rest from the other four models (*dollar, free, monday* and *viagra*), 100 from each one.

For this classification problem, according to (25), one has to calculate a threshold c such that a new sequence (of the validation set) will be assigned to the model *money* if the probability under this set of parameters is bigger than c.

For calculating the threshold c, one can compute the probabilities of every sequence of the training set under the parameters of the model *money*, and to minimize the probabilities of type I and II errors (that is, the probabilities of obtaining false positives and false negatives, respectively). If we fix the probability of type II error as smaller than a certain amount α, the problem to be solved is

$$\begin{aligned} \max_{c} \quad & P[T|T] \\ \text{s.t.} \quad & P[T|F] < \alpha \end{aligned} \tag{26}$$

where $P[T|T]$ represents the probability of assigning a sequence to the model, say *money*, given that it really belongs to that model, and $P[T|F]$ represents the probability of assigning it to the model *money* when it does not belong to that model.

The problem has been solved for $\alpha = 0.05$ and $\alpha = 0.01$. The threshold is the result of considering the value of the 95th percentile (or the 99th percentile, respectively) for the probabilities of the sequences of the training set not belonging to the considered model. The values for the thresholds are 6.8274E−05 and $1.4668E{-}04$, respectively.

Afterwards, we consider a validation set formed by 100 sequences from the model *money* and 400 sequences from the other models (100 from each one).

The following table shows the result of the classification problem for this validation set, when the threshold has been calculated by imposing $\alpha = 0.05$.

	Assigned to 'money'	Assigned to 'others'
Variants of 'money'	97	3
Other words	5	395

If we consider the following measures (see Breiman et al., 1984, for instance):

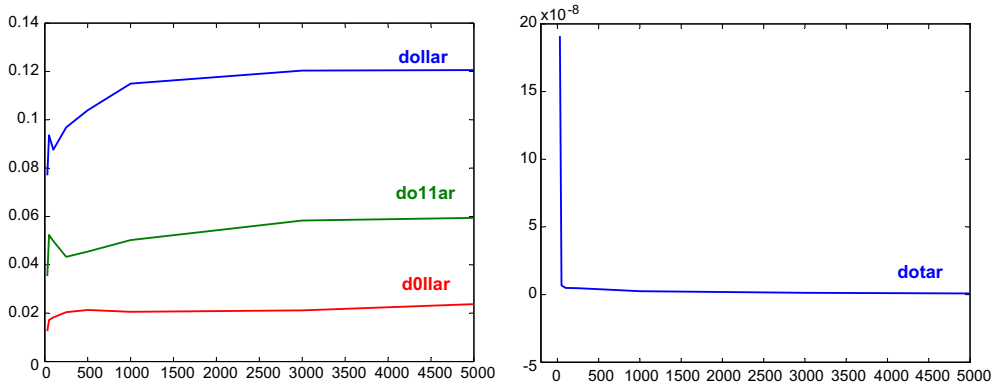

Fig. 2. Probability of three observed sequences of the model 'dollar' (left) and probability of the observed sequence 'dotar' under the model 'dollar' (right).

$$\text{Sensitivity} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}}$$

$$\text{Specifity} = \frac{\text{True negatives}}{\text{True negatives} + \text{False negatives}}$$

$$\text{Accuracy} = \frac{\text{Correctly classified}}{\text{Total of sequences}}$$

$$\text{Error} = \frac{\text{Misclassified}}{\text{Total of sequences}}$$

one has that the sensitivity in this case is 97%, the specificity is 98.75%, the accuracy is 98.4%, and the error, 1.6%.

And, in the case of fixing $\alpha = 0.01$, the table obtained for the classification problem is the following:

	Assigned to 'money'	Assigned to 'others'
Variants of 'money'	94	6
Other words	4	396

In this case, the sensitivity is 94%, the specificity is 99%, the accuracy is 98% and the error is 2%.

We can observe that in the first case, the sensitivity is bigger, because the classification model is more permissive and flexible than in the second case, where sensitivity is worse, but specificity is a little better. The choice of the parameter α will depend on the misclassification costs (if the client prefers to eliminate the maximum possible amount of spam mail, with the possibility of eliminating some ham mail too, or he prefers a more flexible model).

Similar results as the previously presented have been obtained for the rest of models, although the details have been omitted.

Appendix A

A.1. Proof of Lemma 3.1

The formula (7) will be proved.

By using the expression (4), one has that

$$\alpha_{s+1,t+1}(m_{r+1}, y_{t+1}) = P_\lambda[(Y_1, \ldots, Y_s) \in A_{s,t}, Y_{s+1} = y_{t+1}, X_{s+1} = m_{r+1}]$$

By considering all the possibilities for the hidden states X_s,

$$\begin{aligned}
&\alpha_{s+1,t+1}(m_{r+1}, y_{t+1}) \\
&\quad = P_\lambda[(Y_1, \ldots, Y_{s-1}) \in A_{s-1,t-1}, Y_s = y_t, X_s = m_r, \\
&\qquad Y_{s+1} = y_{t+1}, X_{s+1} = m_{r+1}] \\
&\qquad + P_\lambda[(Y_1, \ldots, Y_{s-1}) \in A_{s-1,t-1}, Y_s = y_t, \\
&\qquad X_s = i_r, Y_{s+1} = y_{t+1}, X_{s+1} = m_{r+1}] \\
&\qquad + P_\lambda[(Y_1, \ldots, Y_{s-1}) \in A_{s-1,t}, Y_s = *, \\
&\qquad X_s = d_r, Y_{s+1} = y_{t+1}, X_{s+1} = m_{r+1}]
\end{aligned}$$

Due to the properties of the conditioned probability,

$$\begin{aligned}
&\alpha_{s+1,t+1}(m_{r+1}, y_{t+1}) \\
&\quad = P_\lambda[(Y_1, \ldots, Y_{s-1}) \in A_{s-1,t-1}, \\
&\qquad Y_s = y_t, Y_{s+1} = y_{t+1} | X_{s+1} = m_{r+1}, X_s = m_r] \\
&\qquad \cdot P_\lambda[X_{s+1} = m_{r+1} | X_s = m_r] \cdot P_\lambda[X_s = m_r] \\
&\qquad + P_\lambda[(Y_1, \ldots, Y_{s-1}) \in A_{s-1,t-1}, Y_s = y_t, \\
&\qquad Y_{s+1} = y_{t+1} | X_{s+1} = m_{r+1}, X_s = i_r] \\
&\qquad \cdot P_\lambda[X_{s+1} = m_{r+1} | X_s = i_r] \cdot P_\lambda[X_s = i_r] \\
&\qquad + P_\lambda[(Y_1, \ldots, Y_{s-1}) \in A_{s-1,t}, Y_s = *, \\
&\qquad Y_{s+1} = y_{t+1} | X_{s+1} = m_{r+1}, X_s = d_r] \\
&\qquad \cdot P_\lambda[X_{s+1} = m_{r+1} | X_s = d_r] \cdot P_\lambda[X_s = d_r]
\end{aligned}$$

Finally, by using the conditional independence properties of any HMM, one has that

$$\begin{aligned}
&\alpha_{s+1,t+1}(m_{r+1}, y_{t+1}) \\
&\quad = P_\lambda[Y_{s+1} = y_{t+1} | X_{s+1} = m_{r+1}] \\
&\qquad \cdot [P_\lambda[(Y_1, \ldots, Y_{s-1}) \in A_{s-1,t-1}, Y_s = y_t | X_s = m_r] \\
&\qquad \cdot P_\lambda[X_s = m_r] \cdot a_{m_r, m_{r+1}} + P_\lambda[(Y_1, \ldots, Y_{s-1}) \in A_{s-1,t-1}, \\
&\qquad Y_s = y_t | X_s = i_r] \cdot P_\lambda[X_s = i_r] \cdot a_{i_r, m_{r+1}} \\
&\qquad + P_\lambda[(Y_1, \ldots, Y_{s-1}) \in A_{s-1,t}, Y_s = * | X_s = d_r] \\
&\qquad \cdot P_\lambda[X_s = d_r] \cdot a_{d_r, m_{r+1}}] = b_{m_{r+1}, y_{t+1}} \cdot [\alpha_{s,t}(m_r, y_t) \\
&\qquad \cdot a_{m_r, m_{r+1}} + \alpha_{s,t}(i_r, y_t) \cdot a_{i_r, m_{r+1}} + \alpha_{s,t}(d_r, *) \cdot a_{d_r, m_{r+1}}]
\end{aligned}$$

An analogous reasoning is followed for obtaining (8) and (9).

A.2. Proof of Property 3.1

Let us observe the iteration step, which is the main step of the algorithm, and we will study the cardinal of the set of indexes in which each variable is built.

The index s varies from 1 to S. The index t satisfies the restrictions

$$1 \leqslant t \leqslant T$$
$$s-(S-T) \leqslant t \leqslant s$$

Hence, for each s, we will have to built the variables defined in (4)–(6) a number of times equal to the minimum between T and $S-T=D$.

Finally, the index r satisfies

$$s-(S-R) \leqslant r \leqslant s$$
$$s-t \leqslant r \leqslant R-(S-T)+(s-t)$$

Then, for each s, t, the number of these variables to be built will be the minimum between $S-R=I$ and $R-S+T=M$.

Therefore, the total number of operations is $\mathcal{O}(S\cdot \min\{T,D\}\cdot\min\{I,M\})$.

A.3. Definition of variables for the modified Baum–Welch algorithm

In this subsection, we will define several sets of variables, obtained in terms of the forward (4)–(6) and backward (10)–(12) variables, which will be used to update, iteratively, the set of parameters of the HMM.

Definition A.1. Given the observed sequence $\underline{Y}=(y_1,\ldots,y_T)$, and the set of parameters λ, let us define the variables

$$\delta_{s,t}(i_r,i_r)=P_\lambda[X_s=i_r,X_{s+1}=i_r,Y_s=y_t,Y_{s+1}=y_{t+1},(Y_1,\ldots,Y_{s-1})\in A_{s-1,t-1},(Y_{s+2},\ldots,Y_S)\in B_{s+2,t+2}] \tag{27}$$

$$\delta_{s,t}(i_r,m_{r+1})=P_\lambda[X_s=i_r,X_{s+1}=m_{r+1},Y_s=y_t,Y_{s+1}=y_{t+1},(Y_1,\ldots,Y_{s-1})\in A_{s-1,t-1},(Y_{s+2},\ldots,Y_S)\in B_{s+2,t+2}] \tag{28}$$

$$\delta_{s,t}(i_r,d_{r+1})=P_\lambda[X_s=i_r,X_{s+1}=d_{r+1},Y_s=y_t,Y_{s+1}=*,(Y_1,\ldots,Y_{s-1})\in A_{s-1,t-1},(Y_{s+2},\ldots,Y_S)\in B_{s+2,\ t+1}] \tag{29}$$

Analogously, one can obtain similar definitions for the other cases (when the hidden state in time s is match or delete), only by taking into account that if the chain lies in a delete state d_r, the corresponding emission is '$*$'.

Let us study the relationship between these variables and the forward and backward ones.

Lemma A.1. *Let $\underline{Y}=(y_1,\ldots,y_T)$ be an observed sequence, and λ the set of parameters of a model with sets of states X,Y, and lengths R,S,T. Given the forward variables $\alpha_{s,t}(m_r,y_t)$, $\alpha_{s,t}(i_r,y_t)$, $\alpha_{s,t}(d_r,*)$, and the backward variables $\beta_{s+1,t+1}(m_{r+1},y_{t+1})$, $\beta_{s+1,t+1}(i_r,y_{t+1})$, $\beta_{s+1,t+1}(d_{r+1},*)$, for $1\leqslant s\leqslant S-1$, one has that*

- $\delta_{s,t}(i_r,i_r)=\alpha_{s,t}(i_r,y_t)\cdot a_{i_r,i_r}\cdot b_{i_r,y_{t+1}}\cdot\beta_{s+1,t+1}(i_r,y_{t+1})$, *for*
$$\max\{1,s-(S-T)\}\leqslant t\leqslant\min\{s,T-1\}$$
$$\max\{s+1-(S-R),s-t\}\leqslant r\leqslant\min\{s-1,R-(S-T)+(s-t)\} \tag{30}$$

- $\delta_{s,t}(i_r,m_{r+1})=\alpha_{s,t}(i_r,y_t)\cdot a_{i_r,m_{r+1}}\cdot b_{m_{r+1},y_{t+1}}\cdot\beta_{s+1,t+1}(m_{r+1},y_{t+1})$, *for*
$$\max\{1,s-(S-T)\}\leqslant t\leqslant\min\{s,T-1\}$$
$$\max\{s-(S-R),s-t\}\leqslant r\leqslant\min\{s-1,(R-1)-(S-T)+(s-t)\} \tag{31}$$

- $\delta_{s,t}(i_r,d_{r+1})=\alpha_{s,t}(i_r,y_t)\cdot a_{i_r,d_{r+1}}\cdot b_{d_{r+1},*}\cdot\beta_{s+1,t+1}(d_{r+1},*)$, *for*
$$\max\{1,(s+1)-(S-T)\}\leqslant t\leqslant\min\{s,T\}$$
$$\max\{s-(S-R),s-t\}\leqslant r\leqslant\min\{s-1,R-(S-T)+(s-t)\} \tag{32}$$

And analogous results can be obtained for the rest of cases.

Proof. The proof is done for $\delta_{s,t}(i_r,m_{r+1})$ (similar results can be obtained for the other variables with the same reasoning).

This variable is defined in (28) as

$$\delta_{s,t}(i_r,m_{r+1})=P_\lambda[X_s=i_r,X_{s+1}=m_{r+1},Y_s=y_t,Y_{s+1}=y_{t+1},(Y_1,\ldots,Y_{s-1})\in A_{s-1,t-1},(Y_{s+2},\ldots,Y_S)\in B_{s+2,t+2}]$$

Due to the properties of the conditioned probability, one has that

$$\begin{aligned}\delta_{s,t}(i_r,m_{r+1})&=P_\lambda[Y_s=y_t,Y_{s+1}=y_{t+1},(Y_1,\ldots,Y_{s-1})\in A_{s-1,t-1},(Y_{s+2},\ldots,Y_S)\in B_{s+2,t+2}|X_s=i_r,X_{s+1}=m_{r+1}]\\&\quad\cdot P_\lambda[X_{s+1}=m_{r+1}|X_s=i_r]\cdot P_\lambda[X_s=i_r]\end{aligned}$$

By using the conditional independence properties of any HMM,

$$\begin{aligned}\delta_{s,t}(i_r,m_{r+1})&=P_\lambda[(Y_1,\ldots,Y_{s-1})\in A_{s-1,t-1},Y_s=y_t|X_s=i_r]\\&\quad\cdot P_\lambda[X_s=i_r]\cdot P_\lambda[Y_{s+1}=y_{t+1},(Y_{s+2},\ldots,Y_S)\\&\quad\in B_{s+2,t+2}|X_{s+1}=m_{r+1}]\cdot P_\lambda[X_{s+1}=m_{r+1}|X_s=i_r]\end{aligned}$$

As a result of the conditional independence properties again, and by identifying the resulting expression with the forward (4)–(6) and backward (10)–(12) variables, we obtain

$$\begin{aligned}&\delta_{s,t}(i_r, m_{r+1})\\&\quad = P_\lambda[(Y_1,\ldots,Y_{s-1}) \in A_{s-1,t-1}, Y_s = y_t, X_s = i_r]\\&\qquad \cdot P_\lambda[X_{s+1} = m_{r+1} | X_s = i_r] \cdot P_\lambda[Y_{s+1} = y_{t+1} | X_{s+1} = m_{r+1}]\\&\qquad \cdot P_\lambda[(Y_{s+2},\ldots,Y_S) \in B_{s+2,t+2} | X_{s+1} = m_{r+1}] = \alpha_{s,t}(i_r, y_t)\\&\qquad \cdot a_{i_r,m_{r+1}} \cdot b_{m_{r+1},y_{t+1}} \cdot \beta_{s+1,t+1}(m_{r+1}, y_{t+1}) \qquad \square\end{aligned}$$

Observation A.1. For $s = 0$ and $s = S$, the values for these variables are the following:

- For $s = 0$,

$$\begin{aligned}&\delta_{0,0}(bb, i_0)\\&\quad = P_\lambda[X_0 = bb, X_1 = i_0, Y_0 = *, Y_1 = y_1,\\&\qquad (Y_2,\ldots,Y_S) \in B_{2,2}] = \alpha_{0,0}(bb,*) \cdot a_{bb,i_0}\\&\qquad \cdot b_{i_0,y_1} \cdot \beta_{1,1}(i_0, y_1) = a_{bb,i_0} \cdot b_{i_0,y_1} \cdot \beta_{1,1}(i_0, y_1) \qquad (33)\end{aligned}$$

because $\alpha_{0,0}(bb,*) = 1$.
Analogously,

$$\delta_{0,0}(bb, m_1) = a_{bb,m_1} \cdot b_{m_1,y_1} \cdot \beta_{1,1}(m_1, y_1) \qquad (34)$$

$$\delta_{0,0}(bb, d_1) = a_{bb,d_1} \cdot b_{d_1,*} \cdot \beta_{1,1}(d_1, *) \qquad (35)$$

- For $s = S$,

$$\begin{aligned}\delta_{S,T}(i_R, ee) &= P_\lambda[X_S = i_R, X_{S+1} = ee,\\&\qquad Y_S = Y_T, Y_{S+1} = *, (Y_1,\ldots,Y_{S-1})\\&\qquad \in A_{S-1,T-1}]\\&= \alpha_{S,T}(i_R, y_T) \cdot a_{i_R,ee} \cdot b_{ee,*} \cdot \beta_{S+1,T}(ee, *)\\&= \alpha_{S,T}(i_R, y_T) \cdot a_{i_R,ee} \qquad (36)\end{aligned}$$

because $b_{ee,*} = \beta_{S+1,T}(ee,*) = 1$.
Analogously,

$$\delta_{S,T}(m_R, ee) = \alpha_{S,T}(m_R, y_T) \cdot a_{m_R,ee} \qquad (37)$$

$$\delta_{S,T}(d_R, ee) = \alpha_{S,T}(d_R, *) \cdot a_{d_R,ee} \qquad (38)$$

The following group of variables do not depend on the index t and are calculated from the latter ones, by taking into account every element of the sequence $\underline{Y}$ which can have been observed in a determined time s (for delete states, we consider the last element of $\underline{Y}$ which has been already observed).

Definition A.2. Given the sequence $\underline{Y} = (y_1,\ldots,y_T)$ and the set of parameters λ, let us define the variable

$$\xi_s(i,j) = P_\lambda[X_s = i, X_{s+1} = j, \underline{Y}] \qquad (39)$$

Lemma A.2. *Let $\underline{Y} = (y_1,\ldots,y_T)$ be an observed sequence and λ the set of parameters of a model. Then, given the variable $\delta_{s,t}(i_r, i_r) = \alpha_{s,t}(i_r, y_t) \cdot a_{i_r,i_r} \cdot b_{i_r,y_{t+1}} \cdot \beta_{s+1,t+1}(i_r, y_{t+1})$, which exists for*

$$\begin{aligned}&1 \leqslant s \leqslant S-1\\&t_1 = \max\{1, s-(S-T)\} \leqslant t \leqslant t_2 = \min\{s, T-1\}\\&\max\{s+1-(S-R), s-t\} \leqslant r\\&\quad \leqslant \min\{s-1, R-(S-T)+(s-t)\}\end{aligned}$$

one has that

$$\begin{aligned}\xi_s(i_r, i_r) &= P_\lambda[X_s = i_r, X_{s+1} = i_r, \underline{Y}]\\&= \sum_{t=t_1}^{t_2} P_\lambda[X_s = i_r, X_{s+1} = i_r, Y_s = y_t, Y_{s+1} = y_{t+1},\\&\qquad (Y_1,\ldots,Y_{s-1}) \in A_{s-1,t-1}, (Y_{s+2},\ldots,Y_S)\\&\qquad \in B_{s+2,t+2}] = \sum_{t=t_1}^{t_2} \delta_{s,t}(i_r, i_r) \qquad (40)\end{aligned}$$

for

$$\begin{aligned}&1 \leqslant s \leqslant S-1\\&\max\{s+1-(S-R), s-t_2\} \leqslant r\\&\quad \leqslant \min\{s-1, R-(S-T)+(s-t_1)\}\end{aligned}$$

Analogously, for the rest of variables.

Observation A.2. For the times $s = 0$ and $s = S$, these variables are the same as the variables in Observation A.1, because the only possible values of the index t are, in each case, $t = 0$ and $t = T$, respectively. Thus,

- For $s = 0$, one has that $\xi_0(bb,j) = \delta_{0,0}(bb,j)$, for the three possible hidden states $j = i_0, m_1, d_1$.
- For $s = S$, one has that $\xi_S(j,ee) = \delta_{S,T}(j,ee)$, for the three possible hidden states $j = i_R, m_R, d_R$.

Thirdly, we define the following variables, which are formed from the variables defined in (27)–(29), by summing over the second hidden state as will be seen in Lemma A.3.

Definition A.3. Given the observed sequence $\underline{Y} = (y_1,\ldots,y_T)$ and the model λ, let us define the variables

$$\begin{aligned}\eta_{s,t}(i_r) = P_\lambda[&X_s = i_r, Y_s = y_t, (Y_1,\ldots,Y_{s-1}) \in A_{s-1,t-1},\\&(Y_{s+1},\ldots,Y_S) \in B_{s+1,t+1}] \qquad (41)\end{aligned}$$

$$\begin{aligned}\eta_{s,t}(m_r) = P_\lambda[&X_s = m_r, Y_s = y_t, (Y_1,\ldots,Y_{s-1}) \in A_{s-1,t-1},\\&(Y_{s+1},\ldots,Y_S) \in B_{s+1,t+1}] \qquad (42)\end{aligned}$$

$$\begin{aligned}\eta_{s,t}(d_r) = P_\lambda[&X_s = d_r, Y_s = *, (Y_1,\ldots,Y_{s-1}) \in A_{s-1,t},\\&(Y_{s+1},\ldots,Y_S) \in B_{s+1,t+1}] \qquad (43)\end{aligned}$$

for $s = 1,\ldots,S-1$, $t = 0,\ldots,T$, $r = 0,\ldots,R$.

Lemma A.3. *Let $\underline{Y} = (y_1,\ldots,y_T)$ be an observed sequence and λ the set of parameters of a model. Then, given the variables $\delta_{s,t}(i,j)$ for the suitable values of the indexes s,t,i,j, one has that*

$$\eta_{s,t}(i_r) = \delta_{s,t}(i_r, i_r) + \delta_{s,t}(i_r, m_{r+1}) + \delta_{s,t}(i_r, d_{r+1}) \qquad (44)$$

$$\eta_{s,t}(m_r) = \delta_{s,t}(m_r, i_r) + \delta_{s,t}(m_r, m_{r+1}) + \delta_{s,t}(m_r, d_{r+1}) \qquad (45)$$

$$\eta_{s,t}(d_r) = \delta_{s,t}(d_r, i_r) + \delta_{s,t}(d_r, m_{r+1}) + \delta_{s,t}(d_r, d_{r+1}) \qquad (46)$$

for $s = 1,\ldots,S-1$, $t = 0,\ldots,T$, $r = 0,\ldots,R$.

Observation A.3. For the initial and final times, one has that

- For $s = 0$, the only variable is $\eta_{0,0}(bb) = \delta_{0,0}(bb, i_0) + \delta_{0,0}(bb, m_1) + \delta_{0,0}(bb, d_1)$.
- For $s = S$, the variables are the same as those defined in (36)–(38), that is, $\eta_{S,T}(j) = \delta_{S,T}(j, ee)$, for every value $j = i_R, m_R, d_R$.

Finally, we consider the fourth group of variables.

Definition A.4. Given the observed sequence $\underline{Y} = (y_1, \ldots, y_T)$ and the set of parameters λ, let us define the variable

$$\gamma_s(i) = P_\lambda[X_s = i, \underline{Y}] \tag{47}$$

for every possible value $i \in X$.

Lemma A.4. *Let $\underline{Y}$ be an observed sequence and λ the set of parameters. Then, given the variables $\xi_s(i,j)$, for the corresponding values s, i, j in which the variables are well-defined, one has that*

$$\begin{aligned}\gamma_s(i) = P_\lambda[X_s = i, \underline{Y}] &= \sum_{\text{possible } j} P_\lambda[X_s = i, X_{s+1} = j, \underline{Y}] \\ &= \sum_{\text{possible } j} \xi_s(i,j)\end{aligned} \tag{48}$$

For example, the variable $\gamma_s(i_r)$ is calculated as follows:

$$\begin{aligned}\gamma_s(i_r) = P_\lambda[X_s = i_r, \underline{Y}] &= P_\lambda[X_s = i_r, X_{s+1} = i_r, \underline{Y}] + P_\lambda[X_s \\ &= i_r, X_{s+1} = m_{r+1}, \underline{Y}] + P_\lambda[X_s = i_r, X_{s+1} = d_{r+1}, \underline{Y}] \\ &= \xi_s(i_r, i_r) + \xi_s(i_r, m_{r+1}) + \xi_s(i_r, d_{r+1})\end{aligned}$$

Observation A.4. For the initial and final times, one has that

- For $s = 0$, there is only one variable, $\gamma_0(bb) = \xi_0(bb, i_0) + \xi_0(bb, m_1) + \xi_0(bb, d_1)$.
- For $s = S$, there are three variables, $\gamma_S(j) = \xi_S(j, ee)$, for the three possible hidden states $j = i_R, m_R, d_R$.

References

Abbas, A. E., & Holmes, S. P. (2004). Bioinformatics and management science: some common tools and techniques. *Operations Research, 52*, 165–190.

Baldi, P., & Brunak, S. (2001). Bioinformatics: the machine learning approach. MIT Press: Cambridge, MA.

Barr, A., & Feigenbaum, E. A. (1986). *The handbook of artificial intelligence. Advanced book program* (Vols. I–III). Reading, MA: Addison-Wesley Publishing Company.

Baum, L. E., Petrie, T., Soules, G., & Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics, 41*(1), 164–171.

Berger, J. O. (1980). *Statistical decision theory: foundations concepts and methods*. New York: Springer Verlag.

Bernardo, J. M., & Smith, A. F. M. (1994). *Bayesian theory*. Chichester: John Wiley and Sons.

Bock, H.-H. (2002). *Handbook of data mining and knowledge discovery, chapter data mining tasks and methods: classification*. Oxford: Oxford University Press, pp. 254–267.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Belmont: Wadsworth International Group.

Cappé, O. (2002). Ten Years of HMMs. Available from http://www.tsi.enst.fr/~cappe/docs/hmmbib.html.

Center for Excellence in Service at the University of Marylands Robert H. Smith School of Business and Inc Rockbridge Associates. (2004) National Technology Readiness Survey. Available from http://www.rhsmith.umd.edu/ntrs/NTRS_2004.pdf, February 2005.

Cowell, R. G., Dawid, A. P., Lauritzen, S. L., & Spiegelhalter, D. J. (1999). *Probabilistic networks and expert systems*. New York: Springer-Verlag.

Dempster, A. P., Laird, N. M., & Rubin, D. R. (1977). Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B, 39*, 1–38.

Eddy, S. R. (1998). Profile hidden Markov models. *Bioinformatics, 14*, 755–763.

Hughes, J. P., & Guttorp, P. (1999). A non-homogeneous hidden Markov model for precipitation occurrence. *Applied Statististics, 48*, 15–30.

Hughey, R., & Krogh, A. (1996). Hidden Markov models for sequence analysis: extension and analysis of the basic method. *Computing Applications in the Biosciences, 12*, 95–107.

Krogh, A. (1998). An introduction to hidden Markov models for biological sequences. *Computational Methods in Molecular Biology*, 45–63.

Leonard, T., & Hsu, J. S. J. (2001). *Bayesian methods: an analysis for statisticians and interdisciplinary researchers*. Cambridge: Cambridge University Press.

Levinson, S. E., Rabiner, L. R., & Sondhi, M. M. (1983). An introduction to the application of the theory of probabilistic functions of Markov process to automatic speech recognition. *Bell System Technical Journal, 62*, 1035–1074.

Li, X., Parizeau, M., & Plamondon, R. (2000). Training hidden Markov models with multiple observations – a combinatorial method. *IEEE Transactions on PAMI, 22*, 371–377.

Logan, B., & Moreno, P. (1997). Factorial hidden Markov models for speech recognition. Technical report, Cambridge Research Series, Cambridge.

MacDonald, I., & Zucchini, W. (1997). *Hidden Markov and other models for discrete-valued time series*. London: Chapman & Hall.

Mount, D. W. (2001). *Bioinformatics: sequence and genome analysis*. New York: Cold Spring Harbor.

Poza, M. J., Villarrubia, L., & Siles, J. A. (1991). Teorĺa y Aplicaciones del Reconocimiento Automático del Habla. Technical report, Telefónica, Investigación y Desarrollo.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE, 77*(2), 257–286.

Rabiner, L. R., & Juang, B. H. (1993). *Fundamentals of speech recognition*. New York: Prentice Hall.

Redner, R. A., & Walker, H. F. (1984). Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review, 26*, 195–239.

Senior, A. (1997). A hidden Markov model fingerprint classifier. In Proceedings of the 31st Asilomar conference on signals, systems and computers, pp. 306–310.

Sun, D. X. (1998). A support vector/hidden Markov model approach to phoneme recognition. In: ASA proceedings of the statistical computing section, pp. 125–130.

Wu, C. F. J. (1983). On the convergence properties of the EM algorithm. *The Annals of Statistics, 11*, 95–103.

Ypma, A., & Heskes, T. (2002). Categorization of web pages and user clustering with mixtures of hidden Markov models. In: Proceedings of the international workshop on web knowledge discovery and data mining, pp. 31–43.