APPENDIX 1

VITERBI ALGORITHM

Considering only sequence of states $S$ dropping the dependence on $W$ for simplicity. This does not reduce generality since states univocally individuate words and the probability of states may include the probability of words. The viterbi algorithm finds the sequence of states $S^*$ of the Markov model that can yield the given $T$-length sequence of observations $Y$ with highest probability, given the model.

The basic difference with the forward algorithm is that the latter evaluates the probability that the model gives a sequence $Y$, considering all the possible alternative paths. The viterbi algorithm instead yields the path with the highest probability. In the mathematical description of the algorithm, from the following quantity is introduced:

$$\delta_t(i) = \max_{s_1, s_2, \ldots, s_t} P(S=(s_1, s_2, \ldots, s_t = i), Y=(y_1, y_2, \ldots, y_t) | \Theta)$$

This score is assigned to a state $i$ at time $t$ of the algorithm and it is the probability of a sequence of states $S$ with $s_t=i$ as the last one, and observing $y_1, y_2, \ldots, y_t$. The score is similar to the forward probability of with the difference that in a single path only is considered. Note that the 1-best simplification is included in since in the case of many paths which have the state $j$ at time $t$, only the best one is considered.
\( \delta_t(i) \) can be calculated iteratively, due to the first-order Markov source, similarly to forward probability. \( \delta_{t+1}(j) \) is computed using \( \delta_t(i) \) as:

\[
\delta_{t+1}(j) = \max (t(j) a_{ij} b_j(y_t))
\]

This formula is the principle of dynamic programming applied to stochastic systems.

The arguments that maximize the best predecessor of \( j \) at time \( t+1 \), must be saved for every \( t \) and \( j \) to recover the optimal state sequence. The predecessor state of the state \( j \) at \( t \) is stored in an array \( \psi_1(j) \). The steps of the algorithms are:

1) Initialization

\[
\delta_1(i) = \pi_i b_i(y_1) \quad 1 \leq j \leq N \\
\psi_1(i) = 0 \quad 1 \leq j \leq N
\]

In the first step, to every state \( i \) is assigned a score obtained from the product of the initial probability \( \pi_i \) and the emission probability of the first observation vector \( y_1 \).

2) Recursive step

\[
\delta_t(i) = \max_{1 \leq i \leq N} (\delta_{t-1}(i) a_{ij} b_j(y_t)) \quad 1 \leq j \leq N \\
\psi_t(j) = \arg \max_{1 \leq i \leq N} (\delta_{t-1}(i) a_{ij}) \quad 1 \leq j \leq N
\]

At every recursive step \( t \), with \( t \in \{2, \ldots, T\} \) the best predecessor state \( i \) is searched for every state \( j \), based on the so-called transition score at \( t-1 \):

\[
\delta_{t-1}(i) a_{ij}
\]
In essence, the best predecessor state $i$ of a state $j$ is the state that has the highest transition score, calculated as the product of score accumulated by the predecessor state and the transition probability $a_{ij}$. The $t$-th element of the memory array $\psi_t(j)$ is used to record the best predecessor of state $j$ at time $t$.

3) Final step

$$p^* = \max_{1 \leq i \leq N} \hat{\delta}_t(i)$$

$$s_T^* = \text{Arg} \max_{1 \leq i \leq N} \hat{\delta}_t(i)$$

In the last step $T$, the best state $s_T^*$ is selected. Then, the best sequence is obtained by “backtracking” from $s_T^*$:

$$s_T^* = \psi_t(s_{t+1}^*) \quad t = T-1, T-2, \ldots, 1$$

With backtracking, the optimal sequence of states is obtained starting from $s_T^*$ and using $\psi_t(j)$ produced during the recursive steps.

Note that sums are never used in the algorithm. This suggests performing computation switching to logarithms. This choice reduces computation and avoids underflow problems that may be caused by the repeated multiplication of very small probability values.