Dynamically Weighted Hidden Markov Model for Spam Deobfuscation

Seunghak Lee
Department of Chemistry
POSTECH, Korea
boy3@postech.ac.kr

Iryoung Jeong
Department of Computer
Science Education
Korea University, Korea
crex@comedu.korea.ac.kr

Seungjin Choi
Department of Computer Science
POSTECH, Korea
seungjin@postech.ac.kr

Abstract

Spam deobfuscation is a processing to detect obfuscated words appeared in spam emails and to convert them back to the original words for correct recognition. Lexicon tree hidden Markov model (LT-HMM) was recently shown to be useful in spam deobfuscation. However, LT-HMM suffers from a huge number of states, which is not desirable for practical applications. In this paper we present a complexity-reduced HMM, referred to as *dynamically weighted HMM* (DW-HMM) where the states involving the same emission probability are grouped into super-states, while preserving state transition probabilities of the original HMM. DW-HMM dramatically reduces the number of states and its state transition probabilities are determined in the decoding phase. We illustrate how we convert a LT-HMM to its associated DW-HMM. We confirm the useful behavior of DW-HMM in the task of spam deobfuscation, showing that it significantly reduces the number of states while maintaining the high accuracy.

1 Introduction

Large vocabulary problems in hidden Markov models (HMMs) have been addressed in various areas such as handwriting recognition [A. L. Koerich and Suen, 2003] and speech recognition [Jelinek, 1999]. As the vocabulary size increases, the computational complexity for recognition and decoding dramatically grows, making the recognition system impractical. In order to solve the large vocabulary problem, various methods have been developed, especially in speech and handwriting recognition communities.

For example, one approach is to reduce the lexicon size by using the side information such as word length and word shape. However, this method reduces the global lexicon to only its subset so that the true word hypothesis might be discarded. An alternative approach is to reduce a search space in the large lexicon, since the same initial characters are shared by the lexical tree, leading to redundant computations. Compared to the flat lexicon where whole words are simply corrected, these methods reduce the computational complexity. However, the lexical tree still has a large number of nodes and its effect is limited. Other approaches involve various search strategies. The Viterbi decoding is a time-consuming task for HMMs which contain a large vocabulary (e.g., 20,000 words). The beam search algorithm [Jelinek, 1999] speeds up the Viterbi decoding by eliminating improper state paths in the trellis with a threshold. However, it still has a limitation in performance gain if a large number of states are involved (e.g., 100,000 states) since the number of states should still be taken into account in the algorithm.

Spam deobfuscation is an important pre-processing task for content-based spam filters which use words of contents in emails to determine whether an incoming email is a spam or not. For instance, the word ”viagra” indicates that the email containing such word is most likely a spam. However, spammers obfuscate words to circumvent spam filters by inserting, deleting and substituting characters of words as well as by introducing incorrect segmentation. For example, ”viagra” may be written as ”vi@graa” in spam emails. Some examples of obfuscated words in real spam emails are shown in Table 1. Thus, an important task for successful content-based spam filtering is to restore obfuscated words in spam emails to original ones. Lee and Ng [Lee and Ng, 2005] proposed a method of spam deobfuscation based on a lexicon tree HMM (LT-HMM), demonstrating promising results. Nevertheless, the LT-HMM suffers from a large number of states (e.g., 110,919 states), which is not desirable for practical applications.

Table 1: Examples of obfuscated words in spam emails.

con. tains forwa. rdlook. ing sta. tements
contains forwardlooking statements
th’e lowest rates in thkhe u.s.
the lowest rates in the us
D1scl a1mer Bel ow:
Disclaimer Below

In this paper, we present a complexity-reduced structure of HMM as a solution to the large vocabulary problem. The core idea is to group states involving the same emission probabilities into a few number of super-states, while preserving state transition probabilities of the original HMM. The proposed complexity-reduced HMM is referred to as *dynamically weighted HMM* (DW-HMM), since state transition probabilities are dynamically determined using the data structure which contains the state transition probabilities of the original HMM in the decoding phase. We illustrate how to construct a DW-HMM, given a LT-HMM, reducing the number of states dramatically. We also explain conditions for equivalence between DW-HMM and LT-HMM. We apply DW-HMM to the task of spam deobfuscation, emphasizing its reduced-complexity as well as performance.

2 Dynamically Weighted Hidden Markov Model

A hidden Markov model is a simple dynamic Bayesian network that is characterized by initial state probabilities, state transition probabilities, and emission probabilities. Notations for HMM are as follows:

1. Individual hidden states of HMM are denoted by $\{q_1, q_2, q_3, \ldots, q_K\}$, where K is the number of states. The hidden state vector at time t is denoted by $\boldsymbol{q}_t \in \mathbb{R}^K$.
2. Observation symbols belong to a finite alphabet, $\{y_1, y_2, \ldots, y_M\}$, where M is the number of distinct observation symbols. The observation data at time t is denoted by $\boldsymbol{y}_t \in \mathbb{R}^M$.
3. The state transition probability from state q_i to state q_j is defined by $P(q_j|q_i)$.
4. The emission probability of observation symbol y_i, given state q_j is denoted by $P(y_i|q_j)$.
5. The initial state probability of state q_i is denoted by $\Pi(q_i)$.

With these definitions, HMM is characterized by the joint distribution of states $\boldsymbol{q}_{1:T} = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_T\}$ and observed symbols $\boldsymbol{y}_{1:T} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\}$, which is of a factorized form

$$P(\boldsymbol{q}_{1:T}, \boldsymbol{y}_{1:T}) = \Pi(\boldsymbol{q}_1) P(\boldsymbol{y}_1|\boldsymbol{q}_1) \prod_{t=2}^{T} P(\boldsymbol{q}_t|\boldsymbol{q}_{t-1}) P(\boldsymbol{y}_t|\boldsymbol{q}_t).$$

We first illustrate the DW-HMM with a simple example. Figure 1 shows a transition diagram of a 8-state HMM and its associated DW-HMM that consists of four super-states, s_1, s_2, s_3, and s_4. In this example, the original 8-state HMM (as shown in the top in Figure 1) has four distinct emission probabilities, i.e.,

$$\begin{aligned} &P(y|q_1), \\ P(y|q_2) = &P(y|q_4) = P(y|q_6), \\ P(y|q_3) = &P(y|q_5) = P(y|q_7), \\ &P(y|q_8). \end{aligned}$$

The states involving the same emission probability, are grouped into a super-state, denoted by s in the DW-HMM (as shown in the left of the bottom in Figure 1). In such a case, we construct a DW-HMM containing 4 super-states, s_1, s_2, s_3, and s_4, where the following is satisfied:

$$P(y|s_1) = P(y|q_1),$$

$$\begin{aligned} P(y|s_2) &= P(y|q_2) = P(y|q_4) = P(y|q_6), \\ P(y|s_3) &= P(y|q_3) = P(y|q_5) = P(y|q_7), \end{aligned}$$

$$P(y|s_4) = P(y|q_8).$$

In the DW-HMM, the number of states is reduced from 8 to 4.

The data structure Φ (as shown in the right of the bottom in Figure 1) is constructed, in order to preserve the state transition probabilities defined in the original HMM. Each node in Φ is labeled by the super-state which it belongs to and state transition probabilities are stored, following the original HMM.

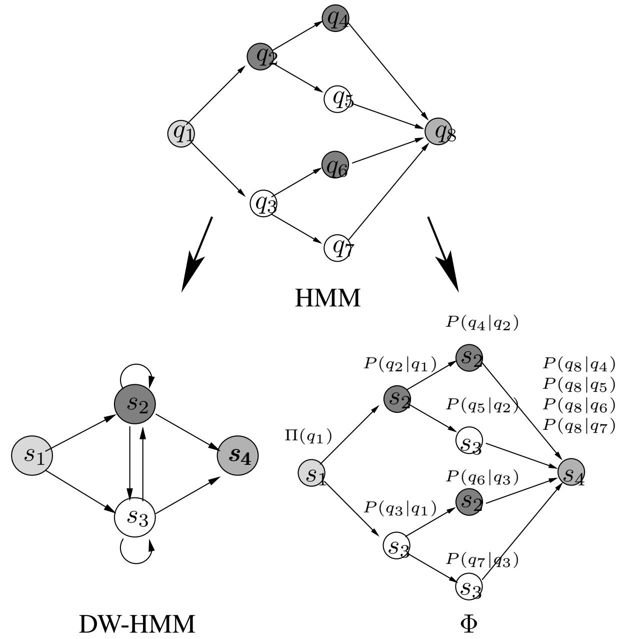

Figure 1: Transition diagrams for the 8-state HMM, its associated 4-state DW-HMM, and the data structure Φ, are shown. The original HMM contains 8 different states with 4 different emission probabilities. The nodes with the same emission probabilities are colored by the same gray-scale. DW-HMM contains 4 different super-states and the data structure Φ is constructed in such a way that the transition probabilities are preserved for the DW-HMM.

The trellis of the 4-state DW-HMM is shown in Figure 2. Transition probabilities are determined by searching Φ using a hypothesis, $\boldsymbol{s}_{1:t}$. To show the relation between DW-HMM and HMM, we define that super-state sequence of DW-HMM, $\boldsymbol{s}_{1:T}$, is corresponding to state sequence of HMM, $\boldsymbol{q}_{1:T}$, if

$$P(y|\boldsymbol{q}_i) = P(y|\boldsymbol{s}_i), \quad 1 \leq i \leq T.$$

Thus, in Figure 2, the hypothesis of the 8-state HMM corresponding to $\boldsymbol{s}_{1:4} = \{s_1, s_2, s_3, s_4\}$ is the state sequence, $\boldsymbol{q}_{1:4} = \{q_1, q_2, q_5, q_8\}$. Given an observation sequence, $\boldsymbol{y}_{1:4} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3, \boldsymbol{y}_4\}$, the joint probabilities of the state sequence and the observation sequence are as follows:

$$P(\boldsymbol{q}_{1:4}, \boldsymbol{y}_{1:4}) = \Pi(\boldsymbol{q}_1) P(\boldsymbol{y}_1|\boldsymbol{q}_1) \prod_{t=2}^{4} P(\boldsymbol{q}_t|\boldsymbol{q}_{t-1}) P(\boldsymbol{y}_t|\boldsymbol{q}_t),$$

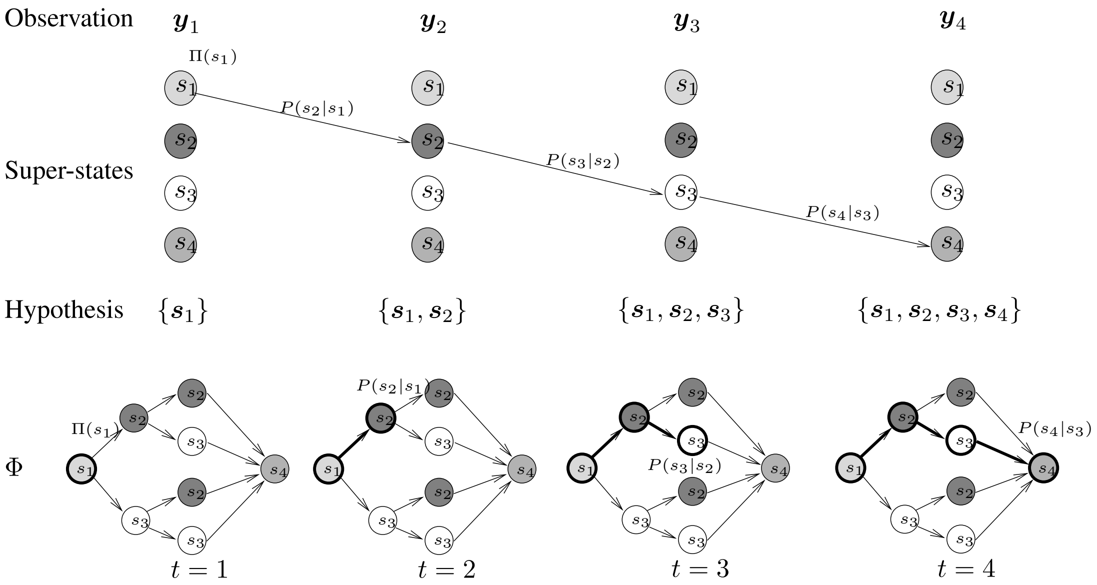

Figure 2: Trellis of the 4-state DW-HMM converted from the 8-state HMM. Transition probabilities are determined by searching a data structure, Φ, with a hypothesis, $\boldsymbol{s}_{1:t}$. The search process at a time instance is represented by the thick line in Φ.

$$P(\boldsymbol{s}_{1:4}, \boldsymbol{y}_{1:4}) = \Pi(\boldsymbol{s}_1)P(\boldsymbol{y}_1|\boldsymbol{s}_1)\prod_{t=2}^{4} P(\boldsymbol{s}_t|\boldsymbol{s}_{t-1})P(\boldsymbol{y}_t|\boldsymbol{s}_t).$$

Since $\boldsymbol{q}_{1:4}$ is the state sequence corresponding to $\boldsymbol{s}_{1:4}$, the emission probabilities involved in the joint probabilities are equal as follows:

$$\begin{aligned} P(y|q_1) &= P(y|s_1), \\ P(y|q_2) &= P(y|s_2), \\ P(y|q_5) &= P(y|s_3), \\ P(y|q_8) &= P(y|s_4). \end{aligned}$$

The DW-HMM searches Φ with the hypothesis, $\boldsymbol{s}_{1:t}$, determining the transition probabilities as follows:

$$\begin{aligned} \Pi(s_1) &= \Pi(q_1), \\ P(s_2|s_1) &= P(q_2|q_1), \\ P(s_3|s_2) &= P(q_5|q_2), \\ P(s_4|s_3) &= P(q_8|q_5). \end{aligned}$$

By the above equalities, the 8-state HMM and the 4-state DW-HMM have the same joint probability as follows:

$$P(\boldsymbol{q}_{1:4}, \boldsymbol{y}_{1:4}) = P(\boldsymbol{s}_{1:4}, \boldsymbol{y}_{1:4}).$$

In searching of a probability, $P(\boldsymbol{s}_t|\boldsymbol{s}_{t-1})$, we assume that state sequence $\boldsymbol{s}_{1:t}$ is unique in Φ. The constraint is not so strong in that if there are several state sequences which have exactly the same emission probabilities for each state, it is very possible that the model has redundant paths. Therefore, it is usual when the constraint is kept in HMMs.

The joint probability of state sequence $\boldsymbol{q}_{1:T} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_T\}$ and an observation sequence $\boldsymbol{y}_{1:T} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T\}$ of HMMs equals that of the corresponding super-state sequence $\boldsymbol{s}_{1:T} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_T\}$ and the same observation sequence $\boldsymbol{y}_{1:T}$ of converted DW-HMMs as follows:

$$P(\boldsymbol{q}_{1:T}, \boldsymbol{y}_{1:T}) = P(\boldsymbol{s}_{1:T}, \boldsymbol{y}_{1:T}).$$

The state transition probability of DW-HMM is defined as follows:

$$P(\boldsymbol{s}_t|\boldsymbol{s}_{t-1}) = \begin{cases} 0 & \text{if } \boldsymbol{s}_{1:t} \text{ does not match} \\ & \text{any paths in } \Phi \\ \omega(\boldsymbol{s}_{1:t}) & \text{otherwise,} \end{cases}$$

where $\boldsymbol{s}_{1:t}$ is state sequence decoded so far, $\boldsymbol{s}_{1:t} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_{t-1}, \boldsymbol{s}_t\}$. $\omega(\boldsymbol{s}_{1:t})$ is a weight function which returns transition probabilities from Φ. The weight function, $\omega(\boldsymbol{s}_{1:t})$, traces a hypothesis, $\boldsymbol{s}_{1:t}$, in Φ, and returns a state transition probability, $P(\boldsymbol{s}_t|\boldsymbol{s}_{t-1})$ that is stored in a node visited by $\boldsymbol{s}_t$. For example, in Figure 2, $\omega(\boldsymbol{s}_{1:3})$ determines the state transition probability, $P(\boldsymbol{s}_3|\boldsymbol{s}_2)$ at $t = 3$. The thick line in Φ shows that $\omega(\boldsymbol{s}_{1:3})$ traces the hypothesis $\boldsymbol{s}_{1:3}$, and returns the transition probability, $P(\boldsymbol{s}_3|\boldsymbol{s}_2)$, stored in the node, s_3, which is the same as the value of $P(q_5|q_2)$ as shown in Figure 1. If a node contains several state transition probabilities, we choose the correct one according to the node visited at $t - 1$. It should be noted that DW-HMM determines the transition probability by searching the data structure of Φ whereas HMMs find them in the transition probability matrix.

DW-HMMs converted from HMMs have the following characteristics. First, DW-HMMs are useful for HMMs which have a very large state transition structure and common emission probabilities among a number of states, such as LT-HMMs. When we convert a HMM to its associated DW-HMM, the computational complexity significantly decreases because the number of states is dramatically reduced.

Second, there is no need to maintain a transition probability matrix since the weight function dynamically gives transition probabilities in the decoding phase. Third, DW-HMMs are so flexible that it is easy to add, delete, or change any states in the state transition structure since only the data structure, Φ, needs to be updated. Fourth, the speed and accuracy are configurable by using beam search algorithm and N-best search [Jelinek, 1999] respectively, thus securing a desirable performance.

2.1 Conversion algorithm

To convert a HMM to DW-HMM, we should make a set of super-states, S, from the states of HMM which represent unique emission probabilities. When S consists of a small number of super-states and the original HMM's state transition structure is large, the DW-HMM is more efficient than HMMs. For example, a LT-HMM is a good candidate to convert to a DW-HMM, because it only has a few states with unique emission probabilities and contains large trie dictionary. The following explains the algorithm of converting a HMM to DW-HMM.

Algorithm: Conversion of HMM to DW-HMM

1. Make a set of super-states which have unique emission probabilities, $S = \{s_1, s_2, \dots, s_K\}$, from a HMM. In this step, the number of states of HMM is reduced as shown in Figure 1.

2. If there are any loops (self-transitions) in the HMM, add additional super-states. For example, if s_i has a loop, an additional s_j state is made. It makes it possible to distinguish between self-transitions and non-self-transitions.

3. Construct the DW-HMM associated with the HMM using super-states in S. State transitions are made between super-states if there exists a state transition in the HMM, from which the super-states are made. For example, in Figure 1, the 4-state DW-HMM has a state transition from s_2 to s_3 because q_2 may transition to q_5 in the 8-state HMM.

4. Make a data structure, Φ, to define a weight function, $\omega(\boldsymbol{s}_{1:t})$, which gives the transition probability of the DW-HMM. Φ contains the transition structure of the HMM and stores transition probabilities in each node. In making Φ, self-transitions in the HMM are changed as shown in Figure 3. The super-state that has a loop transitions to an additional super-state made from step 2 and it transitions to states where the original state goes. The structure of Φ and the state transition structure of the HMM may be different due to self-transitions.

5. Define emission probabilities of the DW-HMM's super-states which are the same as of the corresponding states of the HMM.

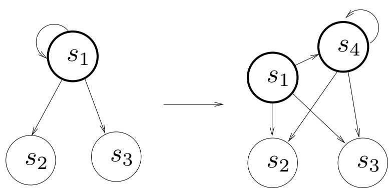

Figure 3: Representation of self-transitions of HMMs in Φ.

2.2 Conditions for equivalence

In the conversion process, DW-HMM conserves transition and emission probabilities. However, there is a difference between HMM and its associated DW-HMM when we use straightforward Viterbi algorithm. Figure 4 illustrates trellis of DW-HMM at $t = 3$ and the state s_1, the only state path $\{s_1, s_2, s_1\}$ that can propagate further since $\{s_1, s_3, s_1\}$ has a smaller probability than $\{s_1, s_2, s_1\}$. Therefore, the path $\{s_1, s_3, s_1\}$ is discarded in a purging step of the Viterbi algorithm. However, in case of HMM shown in Figure 5, at $t = 3$ and the state q_1, both state paths $\{q_1, q_2, q_1\}$ and $\{q_1, q_3, q_1\}$ can propagate to the next time instant because there are two q_1 states.

To ensure that the results of DW-HMM and HMM are the same, we adapt N-best search for DW-HMMs and we choose the best path at the last time instant in the trellis. Figure 6 shows that the two-best search makes two hypotheses at s_1 and $t = 3$. Both state paths $\{s_1, s_2, s_1\}$ and $\{s_1, s_3, s_1\}$ are kept to propagate further.

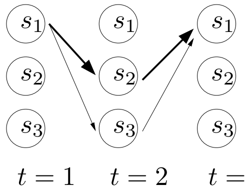

Figure 4: Trellis for DW-HMM.

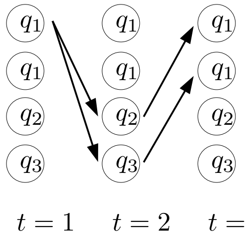

Figure 5: Trellis for HMM.

Figure 7 illustrates the case when two-best search is useful in LT-HMM. We denote the physical property of the states in the nodes, showing which states have the same emission

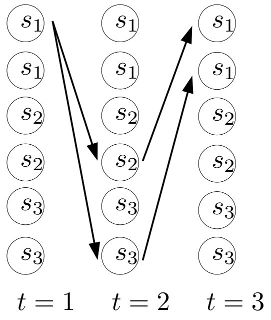

Figure 6: Trellis for two-best search.

probability. q_1 and q_N are the initial and final state of the LT-HMM, respectively. Assuming that there exist two hypotheses at $t = 4$, $\{q_1, a, b, a\}$, $\{q_1, a, c, a\}$, the LT-HMM allow them to propagate further. However, the DW-HMM cannot preserve both at $t = 4$ if we use one-best search. Since the states labeled "a" in the LT-HMM are grouped into a super-state, one hypothesis should be selected at $t = 4$. Thus, in case of the DW-HMM, if the answer's state sequence is $\{q_1, a, c, a, c, i, a, q_N\}$, and only one hypothesis, $\{q_1, a, b, a\}$, is chosen at $t = 4$, we will fail to find the answer. We address the problem with N-best search. If we adapt two-best search, two hypotheses, $\{q_1, a, b, a\}$, $\{q_1, a, c, a\}$, are able to propagate further, thereby preserving the hypothesis for the answer.

HMM and converted DW-HMM give exactly the same results if we adapt the N-best search where N is the maximum number of states of HMMs that compose the same super-state and propagate further at a time instant. However, for practical purposes, N does not need to be large. Our application for spam deobfuscation shows that the DW-HMM works well when N is just two.

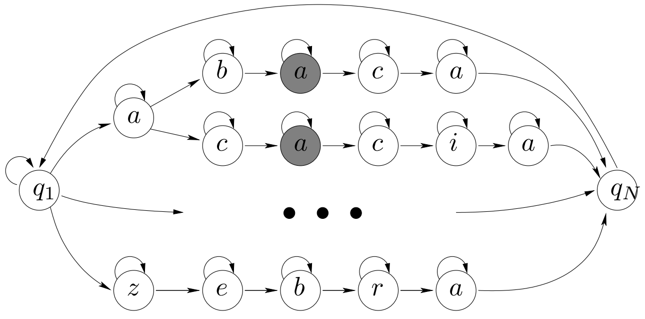

Figure 7: Transition diagram of LT-HMM. By using two-best search, its associated DW-HMM is able to keep two states labeled "a" at a time instance.

3 Application

A lexicon tree hidden Markov model(LT-HMM) for spam deobfuscation was proposed by Lee and Ng [Lee and Ng, 2005]. It consists of 110,919 states and 70 observation symbols, such as the English alphabet, the space, and all other standard non-control ASCII characters. We transform the LT-HMM to the DW-HMM which consists of 55 super-states and the same observation symbols as the model of Lee and Ng. Transition probabilities of the DW-HMM are determined in the decoding phase by a weight function equipped with Φ, which is a data structure of lexicon tree containing transition probabilities of the LT-HMM. Null transitions are allowed and their transition probabilities are also determined by the weight function. It recovers deletion of characters in the input data. [1]

The set of individual hidden super-states of the DW-HMM converted from the LT-HMM is as follows:

$$S = \{s_1, s_2, s_3, \ldots, s_{55}\},$$

where s_1 is an initial state and states in $\{s_2, \ldots, s_{27}\}$ are match states. States in $\{s_{28}, \ldots, s_{54}\}$ are insert states and s_{55} is the final state. A match state is the super-state representing the letters of the English alphabet and an insert state is the state stems from the self-transitions of the LT-HMM. Since self-transitions of the LT-HMM reflect insertion of characters in obfuscated words, the state is named an insert state. The final state is the state which represents the end of words.

Transition probability of the DW-HMM is as follows:

$$P(\boldsymbol{s}_t|\boldsymbol{s}_{t-1}) = \begin{cases} 0 & \text{if } \boldsymbol{s}_{1:t} \text{ does not match} \\ & \text{a prefix in } \Phi \\ \omega(\boldsymbol{s}_{1:t}) & \text{otherwise.} \end{cases}$$

The structure of Φ is made using lexicon tree and each node of Φ has the transition probability of the LT-HMM.

Here, a hypothesis $\boldsymbol{s}_{1:t}$ starts from $\boldsymbol{s}_1$ which is the initial state decoded. The DW-HMM converted from the LT-HMM has only one initial state and it may appear many times in the hypothesis since the input data is a sentence. To reduce the computational cost in searching Φ, we can use the subsequence of $\boldsymbol{s}_{1:t}$ to search Φ since it is possible to reach the same node of Φ if a subsequence starts from the initial state at any time instant.

We deobfuscate spam emails by choosing the best path using decoding algorithms given observation characters. For example, given the observation characters, "vi@", the best state sequence, $\{s_1, s_{23}, s_{10}, s_2\}$, is chosen which represents "via".

We define emission probabilities as follows:

$$P(\boldsymbol{y}_t|\boldsymbol{s}_t) \text{ when } \boldsymbol{s}_t \text{ is a match state}$$
$$= \begin{cases} \rho 1 & \text{if } \boldsymbol{y}_t \text{ is a corresponding character} \\ \rho 2 & \text{if } \boldsymbol{y}_t \text{ is a similar character} \\ \rho 3 & \text{otherwise} \end{cases}$$

$$P(\boldsymbol{y}_t|\boldsymbol{s}_t) \text{ when } \boldsymbol{s}_t \text{ is a insert state}$$
$$= \begin{cases} \sigma 1 & \text{if } \boldsymbol{y}_t \text{ is a corresponding character,} \\ & \text{a similar character,} \\ & \text{or not a letter of the English alphabet} \\ \sigma 2 & \text{otherwise} \end{cases}$$

$$P(\boldsymbol{y}_t|\boldsymbol{s}_t) \text{ when } \boldsymbol{s}_t \text{ is a final state}$$
$$= \begin{cases} \psi 1 & \text{if } \boldsymbol{y}_t \text{ is a white space} \\ \psi 2 & \text{if } \boldsymbol{y}_t \text{ is not a letter of the English alphabet} \\ \psi 3 & \text{otherwise.} \end{cases}$$

[1] More than one consecutive character deletion is not considered due to the severe harm in readability.

An emission probability mass function is given according to the type of the observation. Here, a corresponding character is a physical property of each node. For example, if a node represents a character ”a”, the letter ”a” is the corresponding character. A similar character is given by Leet, [2] which lists analogous characters. For example, ”@” is a similar character of the letter ”a.”

3.1 Decoding

The straightforward Viterbi algorithm is used to find the best state sequence, $\boldsymbol{s}_{1:T} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_T\}$, for a given observation sequence $\boldsymbol{y}_{1:L} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_L\}$. Here, T and L can be different due to null transitions.

The accuracy and speed are configurable for DW-HMM by using the N-best search and the beam search algorithm. N-best search makes it possible to improve the accuracy in that multiple super-states are preserved in the decoding phase. For our model, we select the best path rather than N most probable paths at the final time instant in the trellis. However, it increases computational complexity at a cost of the improved accuracy.

The time complexity of the Viterbi algorithm is $O(K^2T)$, where K is the number of states and T is the length of the input. When the N-best search is used, the time complexity is $O((NK)^2T)$. To address the speed issue, the beam search algorithm is used and it improves the speed without losing much accuracy. Although LT-HMMs are also able to speed up by using beam search algorithm, the large number of states limits the speed to some extent.

In our experiment, when we use the Viterbi algorithm, the rate of process was 246 characters/sec. We speed up the deobfuscation process at a rate of 2,038 characters/sec with a beam width of 10 by using the beam search algorithm.

When it comes to the complexity of the weight function, $\omega(\boldsymbol{s}_{1:t})$, it is almost negligible since trie has $O(M)$ complexity where M is the maximum length of words in the lexicon.

3.2 Parameter learning

Our model has several parameters, $\vec{\theta}$ [3], which should be optimized. We adapt greedy hillclimbing search to get local maxima. By using a training set which consists of obfuscated words and corresponding answers, we find the parameter set which locally maximize the log likelihood.

$$\vec{\theta_0} = \arg\max_{\vec{\theta}} \sum_n \log(P(\boldsymbol{s}_{1:t}, \boldsymbol{y}_{1:t})|\vec{\theta}),$$

where $(\boldsymbol{s}_{1:t}, \boldsymbol{y}_{1:t})$ is a pair of obfuscated observations in the training data and corresponding answer’s state sequence. $\vec{\theta_0}$ is the locally optimized parameter set and n is the number of lines of the training set. η and ϵ determine the probability of the self-transition and null transition respectively [Lee and Ng, 2005].

We start optimizing parameters, $\vec{\theta}$, from initial values which are set according to their characteristics. For example, the parameter for the self-transition should be significantly smaller than the parameter for non-self-transition, because the insertion of characters is less frequent than correctly written letters. From the starting point, we optimize each value in $\vec{\theta}$ and get $\vec{\theta_0}$ which is locally maximized around the initial values of the parameters.

4 Experimental results

In our experiment, we define transition probabilities, $P(\boldsymbol{q}_t|\boldsymbol{q}_{t-1})$, and make Φ using a English dictionary (83,552 words) and large email data from spam corpus. [4] Parameters of our model are optimized with actual spam emails containing 65 lines and 447 words. We perform an experiment with actual spam emails which contain 313 lines and 2,131 words, including insertion, substitution, deletion, segmentation, and the mixed types of obfuscation. Almost all the words are included in the lexicon that we use. Table 2 exhibits some examples of various types of obfuscation.

Table 2: Some examples of various types of spam obfuscation.

Type	Obfuscated words	Deobfuscated words
Insertion	ci-iallis sof-tabs	cialis softabs
Substitution	ultr@ @liure	ultra allure
Deletion	disolves under	dissolves under
Segmentation	assu mptions	assumptions
Mixed	in-ves tment	investment

Our experiments are performed using various decoding methods. We use the one-best and two-best searches with various beam widths and evaluate the results in terms of the accuracy and speed. Table 3 shows the accuracy of the results of spam deobfuscation when two-best search with a beam width of five is used. It represents that our model performs well for the insertion, substitution, segmentation, and the mixed types of obfuscation. However, considering that the deletion type of obfuscation is rare in real spam emails, our model has not trained it well.

Table 3: Accuracy of DW-HMM with two-best search and a beam width of five.

Type	Success	Total	Accuracy
Insertion	420	454	0.925
Substitution	246	266	0.925
Deletion	5	19	0.263
Segmentation	129	137	0.942
Mixed	118	133	0.887
All obfuscation	672	721	0.932
All words	2066	2131	0.969

[2] Leet is defined as the modification of written text, see (en.wikipedia.org/wiki/Leet) website.

[3] $\vec{\theta} = \{\eta, \epsilon, \rho1, \rho2, \rho3, \sigma1, \sigma2, \psi1, \psi2, \psi3\}$.

[4] We use 2005 TREC public spam corpus. For more information, see (plg.uwaterloo.ca/˜gvcormac/treccorpus/) website.

Figure 8 shows the effect of N-best search and beam width for the overall accuracy and computation speed [5] of our model. Overall accuracy is defined as the fraction of correctly deobfuscated words and the computation speed is the number of processed characters per second. As the beam width increases, the overall accuracy rises a little only when the one-best search with a beam width of five is used. However, it turns out that the two-best search significantly improves the overall accuracy compared to the one-best search. The two-best search with a beam width of five shows the accuracy of 96.9% and the processing speed of 2,136 characters/sec. The experimental results show that two-best search with a beam width of five is a desirable configuration maintaining the high accuracy and a low computational cost.

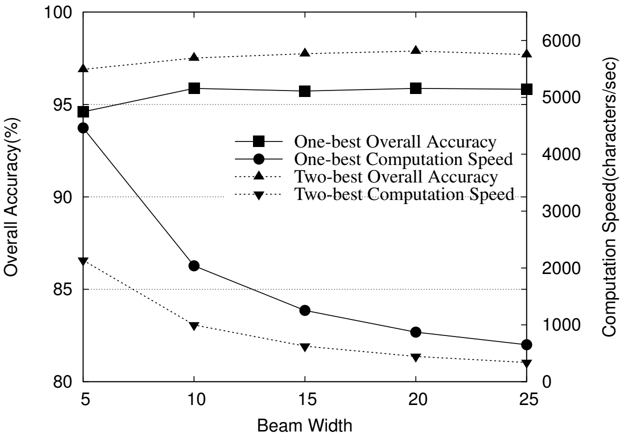

Figure 8: Effect of beam width and N-best search on the speed and overall accuracy.

5 Conclusions

We have presented dynamically weighted hidden Markov model (DW-HMM) which dramatically reduced the number of states when a few sets of states had distinct emission probabilities. The states sharing the same emission probabilities were grouped into super-states in DW-HMM. State transition probabilities in DW-HMM were determined by a weight function which reflects the original state transitions maintained in the data structure Φ, rather than a large transition probability matrix. We have shown how an HMM is converted into its associated DW-HMM, retaining a few number of super-sates. We have applied DW-HMM to the task of spam deobfuscation, where the LT-HMM was replaced by the DW-HMM. Our experimental results showed that it improves the speed from 10 characters/sec to 207 characters/sec when a straightforward Viterbi algorithm is applied. DW-HMM can be applied to diverse areas where a highly structured HMM is used with a few distinct emission probabilities. For example, in speech and handwriting recognition areas, our model may be used to address the large vocabulary problems. We can also use DW-HMM where the state transition structure frequently changes since it is easy to maintain such changes for DW-HMM.

Acknowledgment

Portion of this work was supported by Korea MIC under ITRC support program supervised by the IITA (IITA-2005-C1090-0501-0018).

References

[A. L. Koerich and Suen, 2003] R. Sabourin A. L. Koerich and C. Y. Suen. Large vocabulary off-line handwriting recognition: A survey. *Pattern Analysis and Applications*, 6(2):97–121, 2003.

[Chow and Schwartz, 1989] Y. Chow and R. Schwartz. The n-best algorithm: An efficient procedure for finding top n sentence hypotheses. In *Proc. Speech and Natural Language Workshop*, pages 199–202, Cape Cod, Massachusetts, 1989.

[H. Murveit and Weintraub, 1993] V. Digalakis H. Murveit, J. Butzberger and M. Weintraub. Large vocabulary dictation using SRIs DECIPHER speech recognition system: Progressive search techniques. In *Proc. ICASSP*, pages 319–322, Minneapolis, 1993.

[Jelinek, 1999] F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, Cambridge, Massachusetts, 1999.

[Lee and Kim, 1999] H. Lee and J. Kim. An HMM-based threshold model approach for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):961–973, October 1999.

[Lee and Ng, 2005] H. Lee and A. Y. Ng. Spam deobfuscation using a hidden Markov model. In *Proc. 2nd Conference on Email and Anti-Spam*, Stanford University, CA, USA, 2005.

[Lifchitz and Maire, 2000] A. Lifchitz and F. Maire. A fast lexically constrained Viterbi algorithm for on-line handwriting recognition. In *Proc. 7th International Workshop on Frontiers in Handwriting Recognition (IWFHR-7)*, pages 313–322, Amsterdam, The Netherlands, September 2000.

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

[S. Procter and Mokhtarian, 2000] J. Illingworth S. Procter and F. Mokhtarian. Cursive handwriting recognition using hidden Markov models and a lexicon-driven level building algorithm. *Vision, Image, and Signal Processing*, 147(4):332–339, August 2000.

[5] Computation speed is evaluated when DW-HMM is executed on PC with Pentium 4 1.66GHz and 512MB RAM.