BUILDING STOCHASTIC LANGUAGE MODEL NETWORKS BASED ON SIMULTANEOUS WORD/PHRASE CLUSTERING

Kallirroi Georgila, Nikos Fakotakis, George Kokkinakis

Wire Communications Laboratory
Electrical and Computer Engineering Dept.
University of Patras, Greece
{rgeorgil, fakotaki, gkokkin}@wcl.ee.upatras.gr

ABSTRACT

In this paper we present a novel method for creating stochastic networks for language modelling in spoken dialogue systems. This is accomplished by taking a set of sentences (created manually, derived from simulation experiments, from using the system itself, the application grammar, or a combination of these methods), and training a Hidden Markov Model (HMM), which incorporates all the information about the structure of these sentences. Our technique has the great advantage that during the creation of the HMM, classes containing words or phrases with semantic-syntactic similarities are formed automatically. After all the training data has been used the final HMM is transformed to a stochastic network. The states and observations of the HMM correspond to the word/phrase classes and words/phrases respectively. The nodes of the stochastic network are the word/phrase classes and the arcs are the state transition probabilities of the HMM. The observation probabilities of the HMM correspond to the probabilities within the classes of the stochastic network. Our method has been tested using data from an Interactive telephone-based Directory Assistance Services system[1] and a call-routing spoken dialogue system[2] and has shown the expected advantages.

1. INTRODUCTION

Statistical language models based on n-grams are broadly used in large vocabulary speech recognition systems, e.g. in automatic dictation applications, whereas in spoken dialogue systems finite-state networks are employed, especially in system-driven applications where the user's utterances usually do not deviate from the expectations of the current dialogue state. Spoken dialogue systems deal with specific task-oriented domains, where it is difficult to obtain sufficient data in order to create robust statistical language models. Thus, finite-state networks, are basically developed by using grammars. However, in mixed-initiative or user-driven dialogues where the speaker is not restricted by the dialogue structure and does not conform to any grammatical formalism, automata perform poorly. Therefore, it seems appropriate to combine both types of models. Finite-state networks lead to a better word accuracy for grammatically correct utterances whereas stochastic models are much more robust for spontaneous speech.

In [1], a finite-state automaton and a robust stochastic model are combined to form a language model in the same way as combining HMMs. Both models are represented as category based bigrams. In [2] a back-off n-gram language model is represented through a non-deterministic stochastic finite-state network, which is called Variable N-gram Stochastic Automaton (VNSA). To increase the robustness of the estimation of transition probabilities, word classes and compound words are used. A word class is a set of words having semantic-syntactic similarities (e.g. the set of first names, surnames, etc.). A compound word is a sequence of words with very strong correlation (e.g. *I'd like to*, *thank you*, etc.). The word classes and compound words are defined by hand and then the training data is tagged with the labels of these word classes and compound words.

In this paper, we propose a method for creating stochastic networks thus retaining the advantages of both finite-state networks and statistical language models. The main advantage of our technique is that word/phrase classes are created automatically during the construction of a HMM that consequently is transformed to a stochastic network. In the aforementioned and other existing systems, clusters are created manually or if automatic techniques are used, the clustering procedure is independent of the construction of the finite-state networks or statistical n-grams. That is, networks and stochastic models require already formed clusters in order to become more compact and robust. Our algorithm does not require the existence of classes but creates them automatically and simultaneously with the construction of the stochastic network. Consequently the adaptation of the language models to additional data in the same domain or to another application becomes much easier and efficient with low development costs.

The paper is organized as follows: The algorithm is described in Section 2. In Section 3 an example of the method's application is given, together with experimental results. Finally, a short summary and some conclusions are given in section 4.

2. ALGORITHM DESCRIPTION

The flow chart of our method is shown in Figure 1. At first a set of sentences is selected to train the initial HMM. For every new sentence S the Viterbi algorithm is activated to check whether this sentence could be extracted by the current HMM. The probability assigned to the sentence S is compared with a threshold T, which is defined for the HMM.

a. If the probability assigned to S exceeds or is equal to T, then S is considered to fit in the existing HMM paths. Unknown observations, that is words or phrases, are able to match with

[1] The EU Project LE-48315, IDAS.

[2] This system is developed by the Greek company Knowledge S.A.

existing states, i.e. word or phrase clusters, and become members of them. In this way, the clustering procedure takes place simultaneously with the construction of the HMM. Taking into consideration the modified clusters and sentences that are subsets of *S*, the HMM is updated. That is, the observation probabilities within the existing states (clusters) are reestimated and new states may be added (for the parts of sentences that cannot match with existing states). Subset sentences of *S* are the sentences, all the words of which are contained in sentence *S*. Thus *S* is included in its subsets. b. If the probability assigned to *S* is smaller than *T*, the already existing states (clusters) are not updated, but new ones are created to incorporate the subset sentences of *S* into the HMM. In either case (a) or (b), a new threshold for the updated HMM is estimated and replaces *T*. Then a new sentence is selected, the probability of which is going to be compared with the updated threshold. The procedure iterates until no more sentences are available. Throughout the iterations, phrases are formed (taking into consideration syntactic and semantic restrictions), during each sentence's processing, that is before Viterbi is applied.

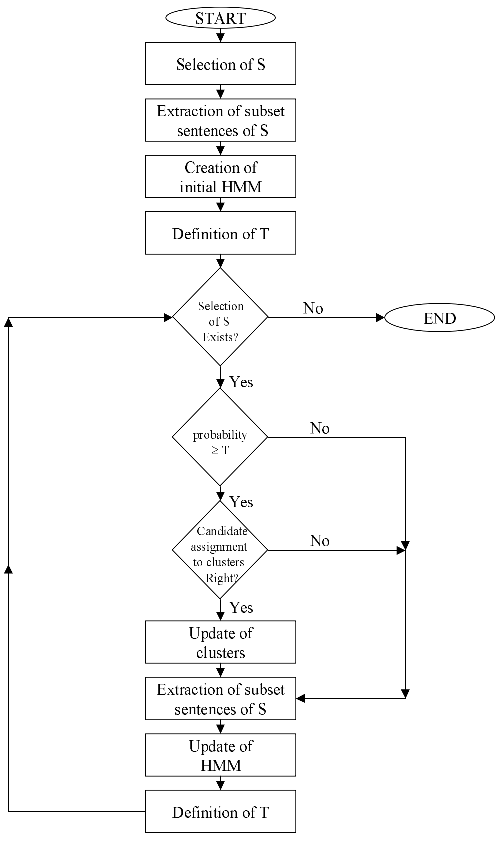

Figure 1: The flow chart of the algorithm.

After the final HMM has been constructed, it is transformed to a stochastic network. The nodes of the network are the word/phrase classes, i.e. the states of the HMM and the arcs are the state transition probabilities of the HMM. The observation probabilities of the HMM correspond to the probabilities within the classes (sub-networks) of the stochastic network. The algorithm is explained in detail in the following.

2.1. Training data

A set of sentences is used as training data. These sentences can be derived from simulation experiments, from the system itself, from the application grammar, be manually created or be produced by a combination of these methods.

2.2. HMM definition

The type of the HMM we use is discrete as defined in [3]. If a word/phrase follows another word/phrase the transition probability a_{ij} between their classes that is the HMM states *i* and *j* is greater than zero, otherwise it is equal to zero. In cases where $a_{ij} \neq 0$, two types of transition probabilities are considered: transitions with equal probability from one state to another or probabilities derived from the number of times a word/phrase class appears after another. To be more precise, if a word/phrase class *u* is followed by *n* word/phrase classes in the training data, then the probability that a word/phrase class *w* occurs after the word/phrase *u*, in the case of equal probabilities, would be $P(w \mid u)=1 / n$ (1). On the other hand, if the number of times class *w* follows *u* is considered, then $P(w \mid u) = N(u, w) / N(\text{u})$ (2) where $N(u, w)$ is the number of occurrences of class *w* after class *u* and $N(u)$ the number of occurrences of class *u*. In the same way, observations, i.e. words/phrases can have equal probabilities within a state (class), or the probabilities are formed according to the frequency of occurrence of the words/phrases. In the former case if a word/phrase *w* belongs to a class $C(w)$, which has *n* members, then the probability of this word/phrase in the class is $P(w \mid C(w)) = 1 / n$ (3). In the latter case $P(w \mid C(w)) = N(w) / N(C(\text{w}))$ (4) where $N(w)$ is the number of occurrences of word/phrase *w* and $N(C(w))$ the number of occurrences of class $C(w)$.

2.3. Initialization

The longest sentence *S* is selected, that is the one with the biggest number of words/phrases, which will be used together with its subset sentences to train the initial HMM. In this way it is ensured that the initial HMM will have as many states as possible and it would be more likely for a new sentence to fit in an existing path than add new states to the model. If not the longest sentence were selected but a medium or small length one, then the initial HMM would have fewer states and observations. Thus a sentence that would be a superset of the one upon which the HMM has been trained, would fail to match with the existing paths and new states would be added, although these states could match with already existing ones.

The number of states *N,* in the initial HMM, is equal to the number of words/phrases of *S*. The number of observations *M* is equal to *N+1*. The redundant observation stands for any unknown word/phrase. Transition and observation probabilities are estimated using equations (1)-(4). The observation of the unknown word/phrase has a probability of *0.000001* in all states. The reason that *0.000001* is used instead of *0* is to avoid overflows in computations. The initial state probability π_i is 1 only when *i* is the *enter* state and *0* for all other states.

After the initial HMM has been built, our next step is the definition of 4 decision thresholds that will be used to decide if a new sentence will fit in an existing path or create a path of its own. The 4 thresholds correspond to the occurrence of 0, 1, 2 and 3 unknown words/phrases respectively. Only 4 thresholds are considered because if a new sentence has more than 3 unknown words/phrases it is more likely that it will fail to follow an existing path. The procedure for the thresholds' definition is as follows. The Viterbi algorithm finds the optimal state sequence that produces the longest sentence *S*. The resulting probability is considered as the first threshold. One of the observations in the above sequence is replaced by the unknown word/phrase observation. In the same way the Viterbi algorithm gives the second threshold for sentences that contain one unknown word/phrase. By replacing two or three observations with the unknown word/phrase observation, the third and fourth thresholds are defined.

2.4. Iterative procedure

The longest sentence of the remaining sentences is selected and transformed into an observation sequence. The Viterbi algorithm again finds the optimal state sequence with the corresponding probability. According to the number of unknown observations the above probability is compared with one of the 4 thresholds. If it exceeds or it is equal to the appropriate threshold *T* then the new observation can match with one of the existing states (classes), and become a member of this class. In order to find the candidate matches the observation and state sequences (derived from the Viterbi algorithm) must be compared. It should be noted here that if we want to compare similar things, we must replace observations with the states (classes) where they belong and then compare this state sequence with the state sequence extracted by Viterbi.

In order to ensure that an observation will not match to a class by mistake, the algorithm gets all the training sentences that contain this observation and computes their probabilities using Viterbi. If the probabilities are greater than or equal to the appropriate threshold, and if the unknown observation matches with the same state in the greatest portion of the training sentences, which contain these observations, e.g. more than 70%, it is considered that the observation can become a member of the class. This check is performed by comparing the observation and state sequences (derived from the Viterbi algorithm). Again the observations must be replaced by the states (classes) where they belong and then compared with the state sequence extracted by Viterbi. The above percentage (e.g. 70%) is set empirically. Higher values lead to very strict clusters and more states, while lower ones are responsible for loose clusters and fewer states. If the check fails, the procedure is continued in the same way as if the probability had not exceeded the threshold in the first place.

So if the probability is greater than or equal to the appropriate threshold and it has been found that unknown observations match with existing states in e.g. more than 70% of the sentences, the HMM is updated as follows. All the sentences that have been used so far together with the new sentence *S* (which caused the match) and the subset sentences of *S* form a set for training the HMM. The number of states remains the same if all the unknown words/phrases of *S* are clustered to existing states, or one state is added for every unknown word/phrase that cannot match with an existing state. The number of unknown words/phrases is added to the previous number of observations to give the current number of observations in the HMM. Transition and observation probabilities are reestimated according to equations (1)-(4) taking into account the new data and clusters. On the other hand, if the probability does not exceed *T* or no unknown word/phrase matches with an existing state for e.g. more than 70% of the sentences, then the unknown words/phrases are considered as new states of the HMM. Again the new data is used for updating the HMM parameters.

After the new HMM has been constructed the longest of the remaining sentences is selected and the steps described above are repeated. The procedure stops when there are no other training sentences.

2.5. Stochastic network construction

The states and observations of the HMM correspond to word/phrase classes and words/phrases respectively. The stochastic network has word/phrase classes as nodes, which are connected with arcs denoting the state transition probabilities of the HMM. The observation probabilities of the HMM correspond to the probabilities within the classes of the stochastic network. During the HMM training, information about the correspondence of the vocabulary words to the observations of the HMM and the contents of the clusters (states), is stored. Thus the stochastic network can be derived from the final HMM.

3. EXAMPLE AND RESULTS

In order to test and improve our method we have used data from the EU project IDAS and a call-routing spoken dialogue system developed by Knowledge S.A. A simplified example is given to clarify the algorithm, in the dialogue state where the user asks for a person in a call-routing system. For simplicity only word classes and not phrase ones are considered. An indicative name *X* is used. The example is given in English so that it is better understood. In Figure 2a a manually created network is depicted. This network is compared to the one created automatically by our method. We consider as training sentences all the paths of the manually created network.

The sentence with the greatest number of words is extracted: *enter I would like to speak to X if that is ok exit* (Sentence 1). The subset sentences of Sentence 1 are: *enter I would like to speak to X exit, enter X if that is ok exit, enter X exit, enter if that is ok I would like to speak to X exit, enter if that is ok X exit.* This set of sentences is used for training the initial HMM. The number of states is equal to the number of words, that is 13 and the number of observations is 14. Each word is associated with an observation (e.g. the word *would* with observation *12*). The 14th observation stands for unknown words. By replacing the words in sentence 1 with their corresponding observations in the HMM we get: *1 3 12 6 10 8 11 13 4 9 5 7 2*. It should be noted that the 2 occurrences of the word *to* are considered as 2 observations. If we replace 1, 2 or 3 observations with the

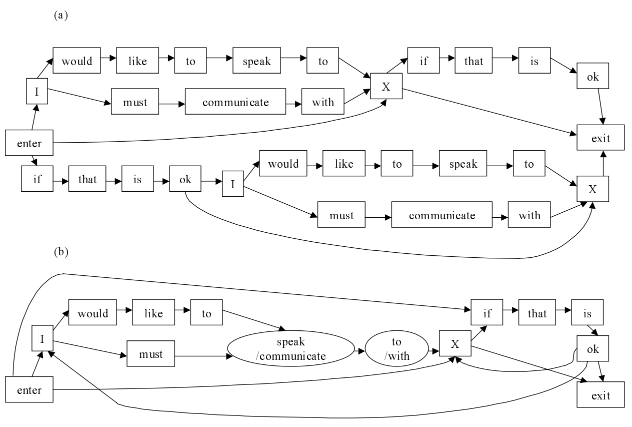

Figure 2: Word networks, (a) manually created and (b) derived from our method.

unknown observation 14 we get the 3 following sequences that are used as input by the Viterbi algorithm to estimate the thresholds for 1, 2 and 3 unknown words: *1 3 12 6 10 14 11 13 4 9 5 7 2* (first)*, 1 3 12 6 10 14 14 13 4 9 5 7 2* (second), *1 3 12 6 10 14 14 14 4 9 5 7 2* (third). The unknown observation could have equally well replaced other observations. That is, the positions of the replacements are not important. In this way, the initial HMM is created and the 4 thresholds are estimated. The next step of the method is to select the longest sentence from the remaining ones: *enter I must communicate with X if that is ok exit* (Sentence 2). We transform the above sentence to a sequence of observations and count the number of unknown observations so that the appropriate threshold is set: *1 3 14 14 14 13 4 9 5 7 2.* In this case we have only 3 unknown observations. The Viterbi algorithm gives that the above observation sequence is extracted from the HMM with a probability greater than the threshold set for 3 unknown observations and suggests the 4th and 5th observations as candidates for matching with states 8 and 11 respectively. Note that in order to do the matching the Viterbi makes some shifts since sentence 1 has 13 observations and sentence 2 only 11. To ensure that this matching does not happen by mistake, we take all the sentences, which are subsets of Sentence 2, and if the matching continues to occur in e.g. more than 70% of the sentences, the word *communicate* becomes a member of the cluster formed by *speak* and the word *with* matches with the second *to*. Now the new HMM has 14 states and the number of observations is 17. State transition and observation probabilities are updated accordingly and the new thresholds are set. We proceed with the longest sentence of the remaining ones and the procedure is continued until no more training sentences exist.

The resulting network is depicted in Figure 2b. As we see it is non-acyclic and produces a superset of the sentences of the manually created network. Some of the additional sentences are correct, e.g. *enter I must speak with X exit* (cannot be derived from the network in Figure 2a). The prediction of paths not contained in the training sentences is one of the main advantages of our method. Other additional sentences are grammatically wrong, e.g. *enter I must communicate to X exit*, *enter if that is ok X if that is ok exit*. However, if we take into account the fact that speech does not conform to strict grammatical rules, these sentences, although incorrect, may lead to accuracy improvement, since more sentences are predicted by the language model. If we had used phrases, the phrase *would like to* would have been matched with *must*, which does not happen if only words are used. Thus phrase-based clusters supersede word-based ones in terms of compactness and efficiency. Phrases can be formed, by using simple or more complicated rules, text chunkers, parsers etc.

In Table 1, we can see the test results on 4 lattices (L1-L4) of the call-routing spoken dialogue system. The first and second columns give the nodes/links of the manually created network (M) and the network (A) derived automatically from our method respectively. We can also see the number of clusters for networks M and A and how many incorrect or partially correct clusters are found in A. For each cell we have two rows, the first for word classes and the second for phrase-based ones.

	Nodes /Links (M)	**Nodes /Links (A)**	**Clusters (M)**	**Clusters (A)**	**Incorrect clusters (A)**
L1	72/138 51/119	28/102 19/89	5 6	5 6	0 1
L2	48/79 32/61	19/94 12/73	4 4	5 4	1 0
L3	40/65 29/53	32/74 23/61	2 3	2 3	0 0
L4	29/41 21/39	22/47 18/45	2 3	2 3	1 0

Table 1: Test results from the call-routing system.

4. SUMMARY AND CONCLUSIONS

In this paper, we presented a method for creating stochastic networks for language modelling in spoken dialogue systems. The main advantage of our algorithm is that word/phrase classes are created automatically during the construction of a HMM model, which will be transformed to a stochastic network. In addition, the resulting network can produce sentences that were not included in the training data. Our method has been tested using sentences from IDAS and a call-routing dialogue system and has shown the expected advantages. Future work will focus on the improvement of the algorithm, the use of phrases based on complicated rules and the exploration of how the resulting network of our method affects recognition performance.

5. REFERENCES

1. W. Eckert, F. Gallwitz and H. Niemann, "Combining stochastic and linguistic language models for recognition of spontaneous speech, *ICASSP '96*, pp. 423-426, Atlanta, GA.

2. G. Riccardi, E. Bocchieri and R. Pieraccini, "Non-deterministic stochastic language models for speech recognition", *ICASSP '95*, pp. 237-240, Detroit, MI.

3. L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proc. of the IEEE*, Vol. 77, No. 2, Feb. 1989, pp. 257-285.