Efficient Stochastic Finite-State Networks for Language Modelling in Spoken Dialogue Systems

Kallirroi Georgila, Nikos Fakotakis, George Kokkinakis

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

Abstract

In this paper we present a novel method for creating language models for Spoken Dialogue Systems (SDS). The idea is based on combining the linguistic structure and the limited requirements for training data of grammar-based models with the robustness of stochastic models regarding spontaneous speech. Our algorithm requires a set of sentences as input, in order to train a Hidden Markov Model (HMM). Classes containing words or phrases with semantic-syntactic similarities are formed automatically and simultaneously with the construction of the HMM. The states and observations of the HMM correspond to the word/phrase classes and words/phrases respectively. The resulting HMM incorporates grammatical structure provided by large context dependencies as well as coverage of ungrammatical spontaneous sentences provided by statistical estimations. The HMM is transformed to a Stochastic Finite-State Network (SFSN), which allows for variable history sizes with no specific upper limit. We used data from 3 different SDSs to evaluate the algorithm. The experiments carried out, resulted in precision and recall values regarding the classes formed, of 0.97 and 0.76 in average, respectively. There was also a reduction of perplexity (16.15% in average) compared to bigrams and a gain in recognition performance (keyword accuracy) of 6.2% compared to grammar-based models and 5.4% compared to bigrams.

1. Introduction

Statistical language models based on *n*-grams are broadly used in large vocabulary speech recognition systems, e.g. in automatic dictation applications. However, reliable *n*-gram estimation requires a very large training corpus and/or sophisticated smoothing techniques. In addition, the search space of a Viterbi continuous speech decoder grows exponentially with the order of *n*. On the other hand, linguistic models can easily describe large context dependencies. However, they are known to be very restrictive. 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, grammar-based models perform poorly. Therefore, it is desirable to combine both types of models. Grammar-based models lead to a better word accuracy for grammatically correct utterances whereas stochastic models are much more robust against effects of spontaneous speech.

A number of researchers have proposed ways to use linguistic knowledge in speech recognition [1][2]. This paper is an extension of our work in [3] where we proposed a method for creating a SFSN, which integrates statistical estimations with grammatical constraints, thus retaining the advantages of both statistical and grammar-based language models. In the aforementioned systems, *n*-grams are derived from grammars, or two separate models, a linguistic and a stochastic one, are interpolated to form a new language model. Our algorithm incorporates in its structure the integration of linguistic and stochastic features. That is the resulting model is built in a straightforward way from the training data and not by the combination of a grammar-based model and a statistically estimated one.

Another main advantage of our technique is that word/phrase classes are created automatically during the construction of the HMM. In most existing systems classes are created manually or if automatic techniques are used the clustering procedure is independent of the construction of the final models. That is in all the aforementioned cases, the language models require already formed clusters in order to become more compact and robust. Our algorithm does not require the preexistence of classes but creates them automatically and simultaneously with the HMM construction.

The resulting HMM of our method is transformed to a SFSN where the nodes 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 (sub-networks) of the stochastic network. The use of stochastic automata to represent statistical language models has been recently proposed [4][5] with the aim to handle accurate language models in a one-step decoding procedure. In [4] a back-off *n*-gram language model is represented through a non-deterministic Stochastic Finite-State Automaton (SFSA), which is called Variable *N*-gram Stochastic Automaton (VNSA). Each state *S* is associated with a *m*-tuple observed in the training set, $v_1, \ldots, v_m$ with $0 \leq m < n$ (*n* is the order of the automaton). The *m*-tuple is called history of the state *S*. In [5] the use of smoothed *K*-Testable Language in the Strict Sense (*K*-TLSS) regular grammars allowed to obtain a deterministic SFSA. *K* stands for the same meaning as *N* in *N*-gram and each state of the automaton represents a word chain of up to *K-1*. Our algorithm produces a deterministic SFSN where each state (node) is defined by a word/phrase, and a state-transition (arc) by a probability. In VNSAs, and SFSAs based on *K*-TLSS, the history size has a value of up to *N-1* and *K-1* respectively. Our algorithm allows for longer distance dependencies to be considered, and results in variable history sizes with no specific upper limit. The upper limit depends on the number of words/phrases of the sentences used as training data and the way these sentences are associated, and in many cases the complete history is retained.

In [3] the experiments carried out showed the efficiency of our algorithm regarding the construction of compact SFSNs with correctly formed clusters compared to the grammar-based ones. In this paper we go 3 steps further. First, new parameters are inserted so that the algorithm is improved. Second, we prove again the efficiency of our method conducting large-scale experiments, using data from 3 different applications, estimating precision and recall values for the clusters formed, and at the same time we consider perplexity and smoothing techniques. Third, we explore how the resulting language model of our method affects recognition performance. In all tests the produced networks are compared not only with grammar-based models but also with bigrams.

The paper is organized as follows: In Section 2 a concise description of the algorithm is given, considering that its detailed presentation is included in [3]. However, some of its new features are presented in detail. Experimental results concerning precision and recall (regarding clusters), and perplexity are given in Section 3. The effects of the produced models on recognition performance are explored in Section 4. Finally, some conclusions are given in Section 5.

2. Algorithm description

At first a set of sentences is selected to train the initial HMM. 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. Our algorithm takes the set of sentences for granted, regardless of how they are produced. However, as it will be shown in the tests carried out, the best results are obtained by mixing sentences taken from the use of the system with sentences derived from grammar-based networks. 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 (Case 1), or if a part of S fits in an existing HMM path (Case 2), then unknown observations of S, that is words/phrases, are able to match existing states, i.e. word/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 existing states). Subset sentences of S are the sentences, all the words/phrases of which are contained in sentence S. The word/phrase order may be considered or not be taken into account. b. If the probability assigned to S is smaller than T and no parts of S fit in existing HMM paths, 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 may be formed (by using simple rules or by taking into consideration sophisticated syntactic and semantic restrictions), during each sentence's processing, that is before Viterbi is applied. After the final HMM has been constructed, it is transformed to a SFSN.

The type of the HMM we use is discrete. Two types of

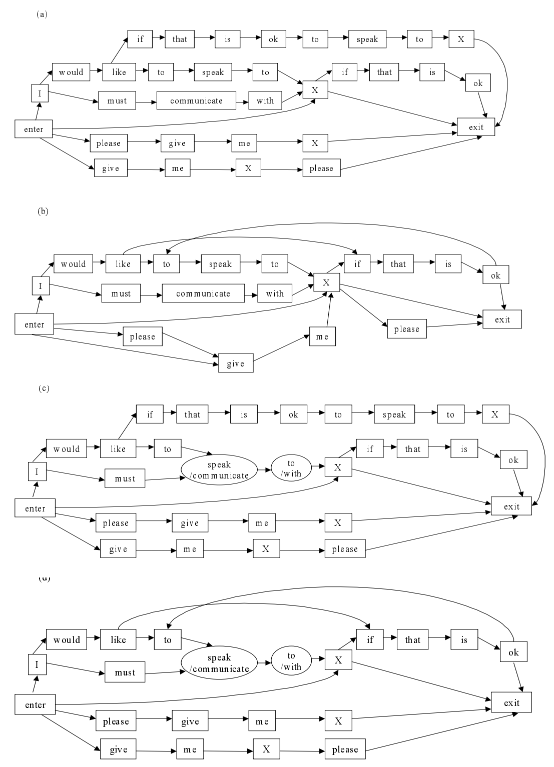

Figure 1: (a) Grammar-based network, (b) bigram, (c) hybrid network (WPO), and (d) hybrid network (NWPO).

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. Thus if a word/phrase class u is followed by n word/phrase classes in the training data, then for the case of equal probabilities, the probability that a word/phrase class w occurs after the word/phrase u 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 $P(w \mid u) = N(u, w) / N(u)$ (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 $P(w \mid C(w)) = N(w) / N(C(\mathrm{w}))$ (4) where $N(w)$ is the number of occurrences of word/phrase w and $N(C(w))$ the number of occurrences of class $C(w)$, that is the sum of occurrences of the words, which belong to class $C(w)$.

In case where the word/phrase order is retained (WPO–Word/Phrase Order), if S is the sequence of words/phrases v_1, v_2, v_3, ..., v_n then a subset sentence of S would have the form v_i, v_j, v_k, ..., v_m $1 \le i < j < k < ... < m \le n$. If the word/phrase order does not pose a constraint (NWPO–No Word/Phrase Order), the subset sentences of S are v_i, v_j, v_k, ..., v_m $1 \le i, j, k, m \le n$. Every time Viterbi is activated, when we use the longest of the training sentences as the new sentence S and have the NWPO case, then more sentences will become subset sentences directly, and the computation time will be reduced. In Figure 1, a grammar-based network, the corresponding bigram and the two hybrid networks derived from our method are depicted. In Figure 1c (WPO) in most paths the complete history is retained. However, In Figure 1d (NWPO) some part of history is lost due to the existence of loops. In general WPO allows for greater history size than NWPO.

In Case 2 where only a part and not the whole sentence matches an existing path straightforwardly or by shift, the candidate matches between new observations and existing states, may be accepted according to some criteria such as frequency of occurrence, position, number of words, word order, if a word/phrase sequence appears more than once in the path etc. If these criteria are very strict, then it is more likely that the candidate matches will be rejected, which will result in a model where grammatical structure supersedes stochastic features. On the other hand, loose criteria will allow matches that do not conform to grammatical rules and may also cause insertions of loops. That is the resulting network will come closer to the *n*-gram structure. Some additional criteria could also be added so that the clusters are correctly formed e.g. words are divided in functional and non-functional words or their Part-Of-Speech (POS) could be considered. Thus a functional word cannot be clustered with a non-functional one and words that do not have the same POS cannot belong to the same class. In the same way phrases of different types may not be allowed to be in the same cluster even if all the other criteria are met. These additional constraints (apart from POS) have been taken into account in tests and have resulted in improved performance.

3. Language model evaluation

In order to test our algorithm we used data from 3 different SDSs: ACCeSS (EU project LE-1 1802, a system for the automation of call center services of a car insurance company), IDAS (EU project LE-48315, an Interactive telephone-based Directory Assistance Services system), and a call-routing SDS developed by Knowledge S.A. We used data from 49 dialogue states (38 of ACCeSS, 7 of IDAS and 4 of the call-router).

Three sets of experiments were carried out. In the first one (Test 1) we considered as training data for our algorithm, the sentences derived from the grammars of the 3 applications. This aims at comparing grammar-based networks with our hybrid models under the same conditions that is with exactly the same training data. The appropriate grammar is loaded according to the SDS and the dialogue state. We carried out experiments with word/phrase classes for both WPO and NWPO. Two types of probability estimations were considered. In the former case, which we will call T1, equations (1) and (3) were used to compute the transition and within class probabilities respectively. In the latter case (T2), we applied equations (2) and (4). Phrases were formed without using sophisticated syntactic or semantic rules but by considering words with very strong correlation (e.g. *I would like to*, etc.). When we extracted the phrases for our training set, we modified the grammar networks to take the phrases into account so that we have phrase-based grammar networks too.

The precision and recall parameters comprise a valid metric for evaluating the performance of our algorithm regarding the clusters formed. We define as *C* the number of correct clusters formed by our method, *T* the total number of clusters formed by using our algorithm, and *TC* the total number of correct clusters, which can be derived from the training data. Then: Precision = C / T and Recall = C / TC.

It is very crucial that the precision is high so that no ill-formed clusters are created since this would result in associating irrelevant words/phrases and in the end in increasing perplexity. Thus very strong thresholds are set to ensure that only correct clusters are created. In Table 1, the precision and recall values are depicted. Computing the average is not an accurate but an indicative metric in our case since the 49 networks are not equivalent in structure. Sometimes a T1 network can have different precision and recall from the corresponding T2 network. We have observed that often the T2 networks have higher precision but lower recall than the T1 ones. That is they are more reliable in forming correct clusters but on the other hand as their probabilities are based on the exact number of occurrences, sometimes they fail to match words/phrases, which are strongly correlated but that do not have equivalent occurrences. In the same way in the WPO case the precision is higher since the word/phrase order is taken into consideration in forming clusters. However, networks derived from the NWPO case tend to have higher recall values. Moreover, phrase (P) networks generally outperform word (W) ones.

	Test 1		**Test 2**		**Test 3**	
	WPO	NWPO	WPO	NWPO	WPO	NWPO
	Precision					
W-T1	0.97	0.96	0.93	0.93	0.96	0.96
W-T2	0.98	0.97	0.94	0.93	0.97	0.96
P-T1	0.97	0.97	0.94	0.94	0.97	0.95
P-T2	0.98	0.97	0.95	0.95	0.97	0.96
	Recall					
W-T1	0.77	0.78	0.74	0.75	0.76	0.76
W-T2	0.77	0.77	0.74	0.74	0.75	0.75
P-T1	0.77	0.79	0.75	0.75	0.76	0.76
P-T2	0.76	0.78	0.73	0.74	0.75	0.76

Table 1: Precision and recall values.

	Perplexity Increase vs. grammars (%)		**Perplexity Reduction vs. bigrams (%)**	
	WPO	NWPO	WPO	NWPO
W-T1	7.34	8.57	17.11	15.85
W-T2	7.22	8.25	17.19	15.96
P-T1	6.89	8.18	17.36	16.13
P-T2	6.81	7.92	17.54	16.25

Table 2: The perplexity (%) in hybrid networks compared to grammar-based ones and bigrams (Test 1).

Table 2 depicts the average increase in perplexity of our hybrid networks compared to the grammar-based ones and the average reduction compared to bigrams. Perplexity in the grammar-based and our hybrid networks is estimated by following paths backwards and multiplying the inverse branching factor at each step. Perplexity in grammar-based networks is smaller than in hybrid ones. However, a very small perplexity indicates that the language model is not robust against utterances not included in the training data. According to the experiments, T2 networks have lower perplexity than T1 ones. Networks of WPO case have lower perplexity values than the ones of NWPO case and phrase-based networks have generally lower perplexity than word-based ones.

In the second set of experiments (Test 2), we considered as training sentences data derived from the use of the system itself, to compare our models with bigrams. The reason is that the power of bigrams arises from the fact that they give reliable estimations when trained with real data. Thus it would not be appropriate to compare our models with bigrams using sentences derived only from grammars. Data is split in two parts (80% for training, 20% for testing) so that perplexity is computed by using a test set different from the training set. Since the test data may contain events not seen in the training sentences, smoothing techniques should be applied. We used the Witten-Bell discounting scheme. If we have a node A connected to a node B, then n is the number of occurrences of links "A *" and t is the number of the distinct links "A *" that exist. We consider only the occurrences of the specific node and not of the word or phrase associated with it, because the word/phrase may appear in more than one nodes. For events that have been seen $P(w \mid h) = c / (n + t)$ (where w is a word, h is the history and c is the number of occurrences of w in the context h). For unseen events $P(w \mid h) = t / (n + t)$. Table 3 shows the average perplexity reduction in our hybrid networks compared to bigrams. The perplexity reduction vs. bigrams is a little higher in Test 1 compared to Test 2. A reasonable explanation would be that the performance of bigrams is better in Test 2 since the training sentences are real data derived from the use of the system itself and not by a grammar. The average precision and recall values for the clusters formed are shown in Table 1. There is a reduction compared to the values of Test 1 caused by the spontaneous nature of the training data in Test 2, which complicates clustering.

	Test 2		**Test 3**	
	WPO	NWPO	WPO	NWPO
W-T1	15.28	13.39	15.71	14.20
W-T2	15.42	13.63	15.85	14.44
P-T1	15.55	14.18	16.02	14.57
P-T2	15.69	14.22	16.15	14.91

Table 3: The average perplexity reduction (%) in hybrid networks compared to bigrams (Tests 2 and 3).

In the third experiment (Test 3) we considered as training sentences data derived from grammars mixed with sentences derived from the use of the system. Table 3 shows the perplexity reduction. Again smoothing was applied. Table 1 depicts the average precision and recall values for the clusters formed. There is a reduction compared to the values of Test 1 but an increase compared to Test 2 since sentences derived from grammars are included in the training data.

4. Recognition performance

In order to investigate how the networks produced by our algorithm affect recognition performance, tests were carried out with data from the call-routing dialogue system. We used 500 recordings spoken by real users, corresponding to the system prompt "*Who would you like to speak with?*". In Table 4 we can see the keyword accuracy for grammar-based (G) networks, hybrid ones and bigrams (2g). The keyword accuracy is the percentage of the sentences where the keyword (*name*) was recognized correctly. The hybrid networks give the best recognition rates due to the fact that they retain the predictability of the grammar-based networks and at the same time they are more robust for spontaneous speech. The columns correspond to the methods of building the models and the training data. This of course does not apply to grammar-based networks and that is why they have the same accuracy in all tests. If the best percentages of grammar-based networks, hybrid ones and bigrams are considered, the gain in recognition performance is 6.2% compared to grammar-based networks and 5.4% compared to bigrams.

	Test 1		**Test 2**		**Test 3**	
W-G	78.0					
P-G	78.2					
	WPO	NWPO	WPO	NWPO	WPO	NWPO
W-T1	80.8	81.4	81.0	81.2	81.6	82.0
W-T2	82.0	82.2	81.6	82.0	82.4	83.0
P-T1	82.4	83.0	82.4	82.6	83.2	83.8
P-T2	82.8	83.2	82.6	82.8	84.0	84.4
W 2g	77.6		78.4		78.8	
P 2g	78.0		78.6		79.0	

Table 4: Keyword recognition accuracy (%).

5. Conclusions

In this paper we presented a method for creating a SFSN for language modelling in SDSs. Word/phrase classes are created automatically during the construction of the SFSN. The resulting SFSN incorporates linguistic knowledge and information provided by statistical estimations and allows variable history sizes with no specific upper limit. The tests carried out, proved the efficiency of our algorithm regarding precision and recall values for the clusters formed. In addition, they showed a considerable reduction in perplexity compared to bigrams, which if it is combined with the gain in recognition performance against both grammar-based networks and bigrams, makes our method appropriate for building efficient language models for SDSs.

Future work will focus on exploring the amount of data necessary for constructing efficient networks. We will also modify our algorithm so that it deals with higher order n-grams, which will result in lower perplexity values. Higher order n-grams will improve the NWPO case, which gave the best recognition results, by reducing the probability of creating wrong loops and clusters. Thus only the NWPO case will be considered since its drawbacks will be diminished. Moreover, enhanced smoothing techniques will be investigated.

6. References

[1] Jurafsky, D., Wooters, C., Segal, J., Stolcke, A., Fosler, E., Tajchman, G., Morgan, N., "Using a stochastic context-free grammar as a language model for speech recognition", *ICASSP'95*, Vol. 1, pp. 189-192.

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

[3] Georgila, K., Fakotakis, N., Kokkinakis, G., "Building stochastic language model networks based on simultaneous word/phrase clustering", *ICSLP 2000*, Vol. 1, pp. 122-125.

[4] Riccardi, G., Pieraccini, R., Bocchieri, E., "Stochastic automata for language modeling", *Computer Speech and Language*, Vol. 10, pp. 265-293, 1996.

[5] Bordel, G., Varona, A., Torres, M.I., "K-TLSS(S) language models for speech recognition", *ICASSP'97*, Vol. 2, pp. 819-822.