Stochastic Finite State Automata Language Model triggered by Dialogue States

*Yannick Esteve *, Frederic Bechet *, Alexis Nasr **, Renato De Mori *,*

* LIA - University of Avignon

www.lia.univ-avignon.fr

** LIM - University of Aix-Marseille II

alexis.nasr@lim.univ-mrs.fr

Abstract

Within the framework of Natural Spoken Dialogue systems, this paper describes a method for dynamically adapting a Language Model (LM) to the dialogue states detected. This LM combines a standard n-gram model with Stochastic Finite State Automata (SFSAs). During the training process, the sentence corpus used to train the LM is split into several hierarchical clusters in a 2-step process which involves both explicit knowledge and statistical criteria. All the clusters are stored in a binary tree where the whole corpus is attached to the root node. Each level of the tree corresponds to a higher specialization of the sub-corpora attached to the nodes and each node corresponds to a different dialogue state. From the same sentence corpus, SFSAs are extracted in order to model longer contexts than the ones used in the standard n-gram model. A set of SFSAs is attached to each node of the tree as well as a sub-LM which combines a bigram trained on the sub-corpus of the node and the SFSAs selected. A first decoding process calculates a word-graph as well as a first sentence hypothesis. This first hypothesis will be used to find the optimal node in the LM tree. Then, a rescoring process of the word graph using the LM attached to the node selected is performed. By adapting the LM to the dialogue state detected, we show a statistically significant gain in WER on a dialogue corpus collected by France Telecom R&D [1].

1. Introduction

Automatic Speech Recognition (ASR) systems usually contain one general Language Model (LM) for the sentences which can be uttered by various speakers. The larger the training corpus of the LM is, the better the recognition rate is expected to be. This approach gives very good results when the context of an application is rather homogeneous. For dialogue applications the context may change with the dialogue state involving a dynamic change of n-gram probabilities. A discussion of various approaches to LM in this case is presented in [3]. This paper describes a new kind of dialogue state dependent LM.

In the case of Natural Spoken Dialogue systems, different dialogue states can generate several kind of sentences which share some common characteristics. Clustering a dialogue training corpus into several sets in order to estimate several LMs can improve the recognition process, like it was shown in [2] and [6]. Nevertheless, several questions remain opened:

- what is the best dialogue state clustering ?
- because in natural spoken dialogue, the speakers have no constraints and the dialogue states are not mutually exclusive, can we take the risk of choosing only one model ?
- if several LMs are used at the same time, how can we select and combine them ?

In a previous paper [5] a LM which combines a traditional bigram model with Stochastic Finite State Automata (SFSA) was described. This model takes into account long contexts (at least longer than two words) characteristic of some regularities in the dialogue training corpus. We present here an extension of this method which consists of linking sets of SFSAs with different dialogue states. The dialogue states considered are obtained by means of a clustering method using both explicit knowledge and statistical analysis. Clusters are organized in a hierarchical tree. A first decoding process is performed with a general bigram model in order to obtain a first sentence hypothesis as well as a word-graph. Then, a rescoring process is performed on the word graph by means of the sub-LM and the SFSAs associated to the dialogue state detected using the first sentence hypothesis. This rescoring method dynamically adapt the recognition process to the dialogue state detected.

2. Clustering the training corpus

In order to train the LMs for the natural spoken dialogue systems, we use text corpora made of speech transcriptions from real sessions between a user and a system. These sessions are made of queries, answers, questions or comments expressed by a user during the dialogue process. When the dialogue is not constrained, users are allowed to express their needs in any way, and the systems have to handle the specific difficulties of spontaneous speech. This high variability is balanced by the limited semantic domain of most dialogue applications (time schedule, directory assistance, etc ...) and by some strong regularities in the sentence structures used by speakers during a dialogue session. For example, most queries start with a sentence pattern like: *I + [would like, want, look for, ...]*. Even if these regularities depend on the targeted application, it is possible to determine dialog independent structures which can be used to cluster a training corpus into several sub-corpora sharing some common regularities.

After a quick presentation of the data on which the experiments were carried out , we will present a method which automatically cluster a training corpus into sub-corpora in two steps: a first rule- based clustering separates the training corpus into four sets of sentences ; then, a perplexity-based clustering process splits in a hierarchical way each set in order to produce a tree of sub-corpora. The root node contains the whole corpus and each level corresponds to a specialization of the corpora attached to the nodes.

[1] This research is supported by France Telecom's R&D under the contract 971B427

2.1. Data presentation

We used in this study a corpus provided by France Telecom R&D which is made of speech transcriptions obtained with the AGS system [8]. The AGS application is a vocal server which gives information about weather or job searching. The AGS corpus contains:

- a training corpus, made of 9842 sentences which have been uttered by several speakers and transcribed manually ;
- a test-set of 1419 word-graphs produced by France Telecom R&D speech recognition system (the word-graphs contain acoustic scores for each word) ;
- the exact transcriptions of the 1419 sentences contained in the test-set.

The vocabulary used in the training corpus contains 823 words.

2.2. Rule-based sentence clustering

Most of the training corpus sentences depend on the dialogue history. For example, it is easy to detect queries made by a user just after the prompt of the system. For this reason it seems natural to use system interventions in order to cluster sentences uttered by users. However, users may not follow the dialogue strategy planned by the system. For example, even if the dialogue management asks a direct question whose answer is either *yes* or *no*, the user instead decide to repeat its query or ask another question. Because of this variability, one needs a large number of sessions between a dialogue system and several users in order to obtain reliable statistics of system and user interventions.

The AGS corpus was too small to estimate these statistics, thus it was decided to cluster the sentences based only on their structures and not on their relations with the dialogue history. The first step in the clustering process consists of tagging and parsing the training corpus in order to extract tags. For example, the sentence: *"I am looking for the phone number . . . "* will be processed as:

```
[(I PRP) (am VBP) (looking_for VB)] VP
[(the DT) (phone_number NN)] NP
```

A set of hand-written rules, using lexical information, Part-Of-Speech tokens and syntactic structures are used to label each sentence according to four main classes:

- $REQUEST$: contains all the sentences which could have been uttered just after the prompt of the system.
- $QUESTION$: contains all the questions asked by a user after an answer from the system. These questions can represent some dialogue command (e.g. *"could you repeat ?"*) or concern some information previously given by the system (e.g. *"yes, is it accessible from Brittany"*).
- $ANSWER$: contains the answers from a user to the questions asked by the system (e.g. *"yes, the weather broadcast"*).
- $OTHER$: contains all the sentences which could not have been labeled with the previous classes. We found here some dialogue command from the user (e.g. *"cancel", "bye", "thanks"*), some non linguistic phenomenon (e.g. *Uh*) or some personal reactions toward the system (e.g. *"oh, I give up", "shut up", . . .*).

Table 1 shows the distribution of the sentences according to these four classes in the training and test corpora.

corpus	$REQU.$	$ANSW.$	$QUES.$	$OTHER$
training	$26,6\%$	$51,4\%$	$13,6\%$	$8,4\%$
test	$22,8\%$	$51,4\%$	13%	$12,8\%$

Table 1: Sentence repartition according to the rule-based classification

2.3. Perplexity-based sentence clustering

This classification method allows us to split the corpus associated to each optimization criteria which have a direct influence on the recognition process. We decided to use *perplexity* criteria because even if there is no evidence that a gain in perplexity will result in a Word Error Rate (WER) reduction, these two quantities are generally related.

During this clustering process we use a binary tree structure where each node is associated with four types of information: a regular expression, a cluster of sentences which match the regular expression, a bigram LM trained on these sentences and the perplexity value calculated on the same sentences with this bigram LM. This method is inspired by the Semantic Classification Tree building method proposed in [4] and the toolkit which implement this particular version of SCT has been developped within the SMADA project [1].

This method is not supervised, meaning that we don't know the optimal number of classes to detect. The minimal set of clusters corresponds to the four classes already presented. The maximal number of classes is restricted by the minimum size of data required to train a bigram model. The algorithm for tree growing is achieved as follow:

- at the initialization step, the root node of the tree contains the whole training corpus (tagged by our POS tagger) ;
- the first four nodes in the tree corresponds to the four classes: each of them is associated to the set of sentences labeled with the corresponding class and an empty regular expression coded $< + >$;
- then, the regular expression of each node is augmented by replacing each symbol $+$ by a token w (which can be any words of the vocabulary or any POS produced by the tagger) in four different contexts: w, $+w$, $+w+$, $w+$;
- for each token and each context, the corpus linked to the node is split in two: the sub-corpus which matches the new regular expression and the one which doesn't ;
- a bigram LM is trained on each produced sub-corpus , for each node, the regular expression is kept which generates the biggest gain between the perplexity attached to the node and the ones obtained on the sub-corpora with the bigram LMs produced ;
- this regular expression augmentation is repeated until we notice no gain in perplexity by specializing more the LMs or when the size of the sub-corpora is under a given threshold.

An example of a tree obtained by the above procedure is given in figure 2. Only the first levels of this tree are shown. They were obtained with the AGS training corpus by giving a value of 300 sentences to the threshold which controls the minimum size of the sub-corpora. This tree contains 43 internal nodes and 45 leaves, which represent a set of 88 sub-corpora and sub-LMs.

2.4. Evaluation of the clustering method

To evaluate our clustering method we compare the gain in perplexity obtained by using the sub-LM attached to a node instead of the general LM on the corpus corresponding to the same node. In order to do that, the sentences of the AGS test corpus have been analyzed by the binary tree calculated on the training corpus. Each sentence follows a path, from the root node to a leaf, by matching the regular expression of each node of the path with the sentence: if the matching is correct, the next node in the path will be the left son of the node, otherwise it will be the right one. While the path gets longer, the sub-LMs get more specialized. We wanted to check if this specialization was linked with a decrease in the perplexity value of the sentences of the test corpus. Figure 1 shows that the gain in perplexity is proportional to the depth of the LM used in the tree. By using this method until we reach a leaf, the gain is highly significant: the perplexity drops from 13.44 to 9.26 which is a relative gain of more than 30%. We present also in this figure the gain obtain on the training corpus

The following questions are needed to be addressed at this point: will these gains in perplexity produce a gain in WER ? How can we decide which node of the tree to use during the decoding process ?

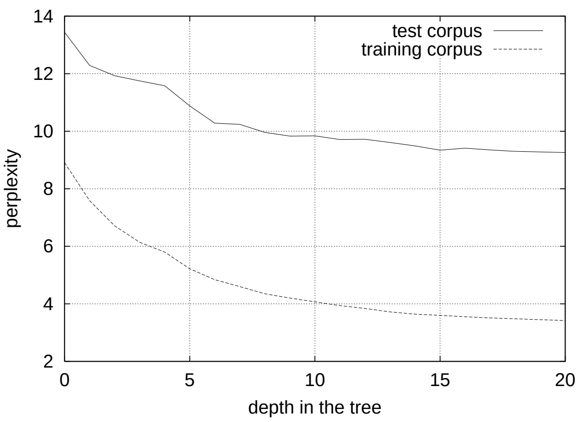

Figure 1: Perplexity as a factor of the depth in the tree

3. Building Stochastic Finite State Automata

The integration of Stochastic Finite State Automata (SFSA) in a n-gram model is motivated by two main objectives: modeling long distance contexts (longer, at least than the *n* histories of the n-gram models) and increasing the generalization capacities of our model. A SFSA can be seen as a particular representation of a class of syntactic phrases. By selecting and grouping together phrases in order to build a SFSA, we are able to model variable length contexts. By replacing in the training corpus each phrase corresponding to a SFSA by the identifier of the SFSA, we are able to build hybrid n-gram models where a *gram* can be either a word or a SFSA.

The extraction method of SFSA from a corpus is briefly summarized here:

1. the training corpus is first tagged and parsed as presented in section 2.2;
2. the phrases extracted after the parsing process are sorted according to their lengths and frequencies;
3. to build a class, we put together the phrases selected which occur with the same right and/or left context ;
4. then, the last step consists of merging some of the classes, according to various statistical criteria (like close internal distribution) in order to increase their generalization capacities.

The classes of phrases obtained are then compiled into SFSAs. The probabilities assigned to each transition in the automata are estimated on the set of phrases which have been used to calculate the automata.

Finally, the SFSAs are integrated in a standard bigram LM by tagging the training corpus with symbols corresponding to the set of automata chosen. More details about SFSAs extraction and integration in a bigram model are presented in our previous paper [5].

4. Decoding process

4.1. Choice of appropriate dialogue state

To choose the most appropriate set of SFSAs to recognize a sentence, the dialogue state in which the system is has to be hypothesized using the LM tree. A first decoding process is performed with the general LM trained on the whole corpus in order to produce a word graph and a first sentence hypothesis H_1.

Then, the hypothesis H_1 is applied to the LM tree previously presented. As we have seen before, each node of this tree characterizes a dialogue state and is associated to a training sub-corpus, a specialized LM and a set of appropriate SFSAs. By selecting the appropriate node thanks to H_1, we are able to adapt the recognition process to the dialogue state detected.

This selection is made as follow: Starting at the root level (level 0), we calculate the perplexity of H_1 with the two sub-LMs associated to each son of the root node. The node selected is the one which offers the lowest value of perplexity for H_1. This process is then repeated in an iterative way on the node selected until we arrive in a leaf node or if no gain in perplexity is observed by further specializing the LM attached to the node. This process is shown in figure 2 on the sentence *"je voudrais le serveur ... "*. The node selected in this example is the node 17. At the end, the node selected is considered as the most appropriate to represent the dialog state in which H_1 have been uttered. A second decoding process can then be performed by means of the sub-LM and the set of SFSAs attached to the selected node.

4.2. Decoding process with SFSAs

The second decoding process is performed with the SFSAs and the specific LM chosen after the first pass. Several methods have been proposed to merge FSA and standard bigram models, the one chosen here allows us to use a standard Viterbi algorithm like in [7]. Using SFSAs at the decoding process implicates to detect phrases in the word-graph (because in our work, phrases are not included in the lexicon of the recognition system which produces the word-graphs). These phrases represent the SFSAs. So, detecting a phrase in the word-graph allows us to use the SFSA(s) which contains this phrase.

The decoding process uses the Viterbi algorithm. To implement this algorithm, for each node of the word-graph, a structure which manage partial theories is created. These structures,

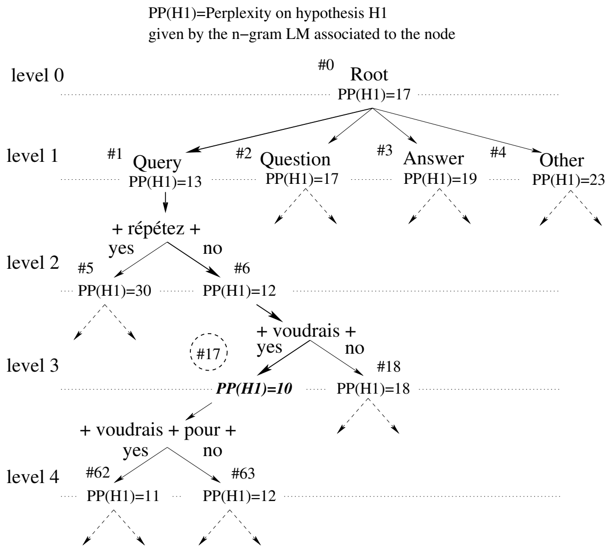

Figure 2: Example of a LM tree and the choice of a node

called Partial Theory Structures are updated by the Viterbi algorithm.

The phrase detection step is performed before running the algorithm. When a phrase is detected, ending at a node n, the Partial Theory Structure of n is modified to take into account the SFSA.

At the end of this process, the SFSAs are included in the Partial Theory Structure of the nodes as if they were words, and the Viterbi algorithm can be used in a standard way.

Whereas this is not the main objective of the method presented in this paper, we can notice that in reducing the number of SFSAs used for the decoding process, we reduce the number of phrases to detect and so speed up the use of SFSAs-based n-gram models.

5. Experiments

In order to evaluate our method, we carried out a set of test experiments. By using a general bigram model on the AGS test corpus, we obtain a WER of 24.70%. This is our baseline model. The confidence interval at 95% on this score is $[23.73, 25.67]$.

By selecting a sub-LM and a set of SFSAs with the method presented in 4.1 we perform a rescoring of the word graph and we obtain a WER of 23.29% as it is presented in table 2. This relative gain of 5.70% of WER is statistically significant according to the confidence interval presented.

Baseline	SFSA	Gain
24.70%	23.29%	$+5.70\%$

Table 2: Comparison of word error rates with baseline and SFSA model

Table 3 shows the results according to the class of the sentences recognized. It is interesting to notice that the biggest gains are obtain on the classes $REQUEST$ and $ANSWER$. These classes corresponds to sentences which have a lot of regularities in their syntactic structures and these regularities seem to be well modeled by our SFSA model.

LM	$REQU.$	$ANSW.$	$QUES.$	$OTHER$
baseline	19.39%	24.7%	26.9%	56.32%
SFSA	18.01%	23.53%	25.21%	54.09%
gain	7.11%	4.73%	6.28%	3.95%

Table 3: WER according to the class of sentences

6. Conclusions

This paper describes a rescoring method of a word graph which is automatically adapted to the dialogue state detected. Each dialogue state corresponds to a node in a tree. Each node contains a sub-LM with SFSAs characteristic to this dialogue state. It is important to point out that dialogue states are not always mutually exclusive. Thanks to the hierarchical structure of the tree, the decision module can choose a node in the upper levels of the tree which groups several of them.

We have now to include, in our method, the information relative to the history of the dialogue. By using the dialogue history in order to increase the robustness of the dialogue state detection process, the rescoring method will be less dependent of the errors made by the first decoding process.

7. References

[1] Frederic Bechet, Els Den Ols, Lou Boves, and Juergen Sienel. Introduction to the ist-hlt project speech driven multimodal automatic directory assistance (smada). In *6th International Conference on Spoken Language Processing ICSLP'2000*, volume 2, pages 731–734, 2000.

[2] David Carter. Improving language models by clustering training sentences. In *Applied Natural Language Processing - ANLP'94*, 1994.

[3] Yannick Esteve, Frederic Bechet, and Renato de Mori. Dynamic selection of language models in a dialog system. In *6th International Conference on Spoken Language Processing ICSLP'2000*, volume 1, pages 214–217, 2000.

[4] Roland Kuhn and Renato de Mori. The application of semantic classification trees to natural language understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):449–460, May 1996.

[5] Alexis Nasr, Yannick Estève, Frédéric Béchet, Thierry Spriet, and Renato de Mori. A language model combining n-grams and stochastic finite state automata. In *Eurospeech*, volume 5, pages 2175–2178, Budapest, Hungary, 1999.

[6] Giuseppe Riccardi and Allen L. Gorin. Stochastic language adaptation over time and state in natural spoken dialogue systems. *IEEE Transactions on Speech and Audio*, 8,1, 2000.

[7] Giuseppe Riccardi, Roberto Pieraccini, and Enrico Bocchieri. Stochastic automata for language modeling. *Computer Speech and Language*, 10:265–293, 1996.

[8] D. Sadek, A. Ferrieux, A. Cozannet, P. Bretier, F. Panaget, and J. Simonin. Effective human-computer cooperative spoken dialogue: the ags demonstrator. In *ICSLP'96, USA*, 1996.