Stochastic Language Modelling for Recognition and Generation in Dialogue Systems

Kallirroi Georgila - Nikos Fakotakis - George Kokkinakis

Wire Communications Laboratory
University of Patras, 265 00 Rion, Patras, Greece
{rgeorgil, fakotaki, gkokkin}@wcl.ee.upatras.gr

ABSTRACT: *A human-computer dialogue system should be able to handle spontaneous ungrammatical sentences in the analysis-recognition stage and at the same time generate grammatically correct responses. Moreover, the constraint of not having enough domain-dependent training material should be overcome. In this paper we present a method able to cope with the above phenomena. It is an incremental data-driven process based on the Viterbi algorithm, which given as input a set of sentences, produces a Hidden Markov Model (HMM) that incorporates grammatical structure provided by large context dependencies as well as coverage of ungrammatical spontaneous sentences provided by statistical estimations. Furthermore, the algorithm generalises the sample data, thereby reducing the required amount of training samples for acquiring reliable models. Adjustment of parameters may lead to a model where stochastic features supersede grammatical structure and the contrary. In the first case, the output HMM can be used as a robust flexible language model in the analysis-recognition stage. In the second case, the HMM becomes appropriate for generating valid system responses without the need of grammars.*

RÉSUMÉ: *Un système de dialogue entre l' homme et l' ordinateur devrait être capable de gérer des phrases spontanées agrammaticales dans le stade de l' analyse et de la récognition et, au même temps, de générer des réponses grammaticalement correctes. On doit aussi resoudre le problème de n' avoir que de training matériel limité pour chaque application specifique. Une méthode capable de se débrouiller entre ces deux difficultés est presentée dans ce travail. Il s' agit d' un procès données–dirigé incrémental, fondé sur l' algorithme Viterbi, qui, donné comme input un ensemble de phrases, produit un HMM qui incorpore structure grammaticale procurée par larges dependences de context et couverture de phrases spontanées agrammaticales procurées par estimations statistiques. En outres, l' algorithme généralise les données d' echantillon et reduit la quantité de training données exigée pour obtenir des modèles précises. Ajustement des paramètres peut guider à un modèle où les caractéristiques stochastiques déplacent la structure grammaticale et au contraire. En prenier cas, l' output HMM peut être utilisé comme un modèle de langage flexible et robuste dans le stade de l' analyse et de la récognition. En deuxième cas, le HMM devient approprié à générer des réponses du systéme valides sans avoir besoin de grammaire.*

KEY WORDS: *language modelling, language generation, HMM, finite-state automata*

MOTS-CLÉS: *modéliser de langage, generation de langage, HMM, automates d' état fini*

1. Introduction

In spoken human-computer dialogues, especially mixed-initiative and user-driven, it is very important that the system can handle certain linguistic phenomena that frequently occur in spontaneous speech such as filled pauses, hesitation and correction. Grammar-based models are known to be very restrictive and cannot adequately deal with the above phenomena. On the other hand, statistical language models based on *n*-grams are much more flexible against effects of spontaneous speech. However, reliable *n*-gram estimation requires a very large training corpus and/or sophisticated smoothing techniques. Dialogue systems deal with specific task-oriented domains, where it is difficult to obtain sufficient data in order to create robust statistical language models, especially in the first phases of their development where no information by the use of the system itself is available. Therefore, it seems appropriate to combine both types of models. Grammar-based models give better results for grammatically correct utterances whereas stochastic models are much more robust to spontaneous speech.

A number of researchers have proposed ways to combine linguistic and stochastic knowledge in the speech recognition process. In some approaches grammars were used in order to compute *n*-gram statistics and they were combined with traditional *n*-grams elicited from a corpus [JUR 95; ECK 96]. Other attempts towards this direction, exploited syntactic information in forming grammatical fragments of various types, and supplanted the standard *n*-gram with *n*-grams of fragment types [MET 93; LLO 95; MOO 95; POP 97; TSU 98; NAS 99]. Link grammars [FON 95] and syntactic information [PER 96; CHE 99; WU 99] were used, in an attempt to improve the language model by providing it with long distance dependencies not captured in the *n*-gram statistics. Although it is not connected straightforwardly to our approach, we should state here that high-level semantic information may also be used to incorporate large-span constraints. Two approaches in that direction were based on word triggers [LAU 93] and Latent Semantic Analysis (LSA) [BEL 98].

The naturalness and perceived intelligence of a spoken dialogue interface does not depend only on its ability to recognise and analyse user utterances correctly but also on the quality of the Natural Language Generation (NLG) module. NLG has been extensively studied for "single-interaction" systems, e.g. summarisers, translators, and report generators, but little is known about effective ways of performing NLG in dialogue systems [GAL 01]. Current approaches to NLG have limited success when applied to conversational systems. In dialogue systems the language use must be extensive and varied [STE 99; GAL 01]. Moreover, the NLG module has to be robust against missing and incomplete data [KNI 95; GAL 01], a problem that occurs often in spoken dialogue systems, and it should operate in a fraction of real time.

The NLG component of most dialogue systems is based on templates. Template-based systems require little linguistic expertise and are cost-effective solutions to

NLG in the early stages of prototyping. However, templates tend to become unmanageable as the system grows, since the number of templates needed to cover all situations while maintaining a reasonable quality can become quite large. Templates are application specific and have to be entirely rewritten when switching to a new application domain. Furthermore, the sentences they produce lack the variability and robustness to missing data, needed by conversational systems, since they merely rely on slot-filling techniques [GAL 01; RAM 01]. General-purpose rule-based generation systems [HOV 88; FAW 92; BAT 97] sidestep these problems, but they tend to be difficult to adapt to task-specific applications due to their generality, sophistication, and their need for a large amount of linguistic knowledge [RAM 01]. In addition, typical rule-based generators do not achieve real-time performance and are unsuitable for dialogue systems [STE 99; OH 00; GAL 01]. Recently, attempts have been made to overcome the problem of both template-based and rule-based systems by introducing a hybrid approach incorporating both models [BUS 98; STE 99; AXE 00].

According to [RAM 92; USZ 96; REI 00], NLG consists of 3 layers: (1) Text Planning, (2) Sentence Planning, and (3) Surface Realisation. Determining the system's intent or communicative goal in dialogue systems is out of the scope of the generator and is usually handled by the dialogue manager. The dialogue structure of a conversational system may be formed either heuristically or by using statistical methods [MOO 93; LEV 97; REI 97]. Some approaches towards sentence planning are described in [GAL 01], [RAM 01] and [WAL 01].

To date, only limited use of statistically derived resources has been made for surface realisation in NLG. In the current approaches, rule-based systems are combined with stochastic models, that is a grammar generates possible strings for a particular semantic input and a statistical language model acts as a filter that prunes and ranks the alternative strings [KNI 95; LAN 98; BAN 00; GAL 01; HUM 01]. In [OH 00], a purely stochastic surface realisation is proposed. Moreover, [RAT 00] introduces trainable methods for surface generation. Nevertheless, research on using grammars for NLG continues [GAR 01].

In this paper we present a method, which can be used for language modelling both in the analysis-recognition stage of a dialogue system and in the phase of generating responses. It should be stated here that our technique deals only with surface realisation. It is an incremental data-driven process based on the Viterbi algorithm. It takes as input a set of sentences and produces a HMM or equivalently a Stochastic Finite-State Network (SFSN), which integrates statistical estimations with grammatical constraints, thus retaining the advantages of both approaches. The incorporated grammatical structure allows for large context dependencies to be considered. That is, the state history varies and is not limited to the value denoted by the order of the HMM. The output HMM may produce the above set of sentences, given as input, together with other sentences not appearing in the training data. This is accomplished both by the creation of word/phrase classes and the stochastic features of the algorithm. Therefore, the algorithm generalises the sample data,

thereby reducing the required amount of training samples for acquiring reliable models. Adjustment of parameters may lead to a model where stochastic features supersede grammatical structure and the contrary. In the first case, the output HMM can be used as a robust language model in the analysis-recognition stage, and in the second case as a model for generating valid system responses without the need of grammars. If we use the resulting HMM in text-based dialogue systems as a language model for recognition of written input, which usually follows grammatical structure, then the parameters of the algorithm should be adjusted as in the response generation case. Nevertheless, the proposed technique is more appropriate for spoken input and output due to its stochastic nature. If the produced HMM is used as a language model in the recognition stage, then the input training sentences represent possible speaker utterances in a dialogue state. On the other hand, in the NLG process, the set of training data consists of alternative forms of system prompts for the same dialogue state.

In [GEO 00; GEO 01] we applied our method to language modelling in the recognition stage of spoken dialogue systems. The current work is an extension of [GEO 00] and [GEO 01]. In the present paper, we explain our technique in detail, focussing on key implementation issues, which are not obvious from the general algorithm description and that were not presented in our previously mentioned works. Furthermore, our method is compared not only with bigrams but also with class-based bigrams, and additional results concerning word recognition accuracy are provided. Finally our algorithm is not only investigated from the perspective of language modelling in the recognition stage but also as a means to generate grammatically correct system responses without the need of grammars and with limited training samples.

The paper is organised as follows: The proposed approach is described in detail in section 2. Experimental results, both for the recognition and the generation stages, are given in section 3. Finally, a summary and some conclusions are provided in section 4.

2. The proposed approach

2.1. Comparison with related work

In the majority of the systems mentioned in section 1 (analysis-recognition stage), *n*-grams are derived from grammars, or two separate models, a linguistic and a stochastic one, are interpolated to form a new language model, and the process of forming grammatical fragments that will be combined with statistical estimations is separate from computing the *n*-grams. 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. [TSU 98] and [NAS 99] represented grammatical fragments as Finite-State Automata (FSA). Our approach

also uses FSA since the resulting HMM is equivalent to a SFSN where the nodes are the word/phrase classes and the arcs are the state-transitions of the HMM, labelled with the corresponding transition probabilities. 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 [RIC 96; MOH 97] with the aim to handle accurate language models in a one-step decoding procedure. In [RIC 96] 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). In [BOR 97; VAR 99; VAR 00] the use of smoothed *K*-Testable language in the Strict Sense (*K*-TSS) regular grammars allowed obtaining 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. According to [HOP 79], non-deterministic are the automata that allow more than one transition from a state, of the same input symbol. Thus in non-deterministic automata, for a given string of symbols there are several possible sequences of states that recognise it. In the SFSNs produced by our method, we cannot have different state-sequences for the same string of symbols. In VNSA and *K*-TSS models, the history size has a value of up to *N-1* and *K-1* respectively. Our algorithm is structured in such a way that 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. A large value of history size indicates a high degree of grammatical structure whereas a lower value entails broader coverage of ungrammatical spontaneous sentences.

Another main advantage of our technique is that word/phrase classes are created automatically during the construction of the HMM. In [ECK 96; POP 97] classes are created manually, in contrast with other existing systems [BRO 92; JAR 93; KNE 93; RIE 96; SMA 96; NAS 99], where automatic techniques are used and the clustering procedure is independent of the construction of the final models. That is classes are merged and/or words move from one class to another, and the procedure iterates until a criterion is optimised e.g. maximum likelihood, perplexity, average mutual information. Then the language model probabilities are estimated according to the formed classes. Thus 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 construction of the HMM. Each word/phrase may belong to more than one cluster, since it is allowed to have multiple instances of the same word/phrase in the HMM. The merging operation preserves the ability to generate all the training data. However, new sentences not included in the training samples may also be generated, which is considered as generalisation of the sample data. Moreover, in case we acquire additional sentences, the HMM is not built from scratch using all the training data (old data together with additional data). That is the

HMM created with the old set of data is used as the initial model in the training procedure. The new data will be used to update the current HMM iteratively until the final model is built. Consequently the adaptation of the language model to additional data in the same domain or to another application becomes much easier and efficient with low development costs.

As it was mentioned in section 1, most approaches that integrate statistical knowledge for surface realisation are based on the following framework: they use a generation-specific grammar and construct a lattice representing all possible strings that the grammar allows for a particular semantic input. Then, in a separate stage, statistics are used to rank the alternatives in terms of "fluency", determined by similarity to *n*-grams in the corpus [KNI 95; LAN 98; GAL 01]. However, these approaches have some limitations, e.g. their statistical nature may lead to ungrammatical sentences, lexical co-occurrences etc. In order to deal with such phenomena [BAN 00] used sub-tree structures, which allow the handling of long distance dependencies. Moreover, [HUM 01] used a statistical language model derived from a full grammar to constraint the simplified version of this grammar, which was responsible for NLG.

By adjusting the appropriate parameters, our algorithm can retain a high level of grammatical structure and cover large context dependencies. Furthermore, it does not require the existence of a grammar in order to function. That is, in our case we do not have a stochastic language model that filters the output of a grammar, but a model that can be used in a straightforward way to generate the dialogue system's responses. Even, if the resulting HMM of our approach is used as a filter for a grammar, as in the previous works [KNI 95; LAN 98; BAN 00; GAL 01; HUM 01], it is expected to give better results than *n*-grams since it incorporates grammatical knowledge in its structure. Nevertheless, it is preferred to apply the produced HMM directly to NLG, since this is the basic advantage of our approach in the response generation case. Again, as in the analysis-recognition stage, the fact that the algorithm generalises from sample data reduces the amount of training sentences required for forming reliable models. The use of clusters and the statistical nature of the algorithm will compensate for missing samples.

The proposed technique has been applied to language modelling both in the recognition and the generation phases of dialogue systems. However, it could also be viewed from another perspective that is as a method for inducing the structure of HMMs from sample observation sequences. The Baum-Welch algorithm, which is used for training HMMs, assumes that the number of the HMM states is fixed, and uses initially random parameters that are iteratively updated until a point is found where the sample likelihood is locally maximal. A more general problem is to additionally find the best HMM topology, which includes not only estimating the transition and observation probabilities but also the number of states. The Baum-Welch algorithm could be used again on fully connected models of varying sizes, picking the model size and topology with the highest posterior probability. (Maximum likelihood estimation is not useful for this comparison since larger

models usually fit data better). However, this approach is very costly and Baum-Welch may get stuck at sub-optimal local maxima. We handle the structure learning problem in an incremental way. An initial model is constructed and the model size is adjusted as new data appears. New observations create new states or merge with existing ones. Therefore, in every iteration the number of states may increase or remain unchanged and the HMM probabilities are updated accordingly. Beyond being incremental, our algorithm is data-driven in that the training data itself completely determines the initial model shape, in contrast with the Baum-Welch algorithm where the initial model is completely uninformed by the data. Other approaches close to ours are the ones described by [THO 86] and [STO 93]. However, these approaches apart from the algorithmic differences they have, compared to our method, were also applied to different tasks.

2.2. Algorithm description

The flow chart of our method is shown in Figure 1. In this section, we describe the algorithm concisely and, in the following ones, we explain the steps it consists of in detail.

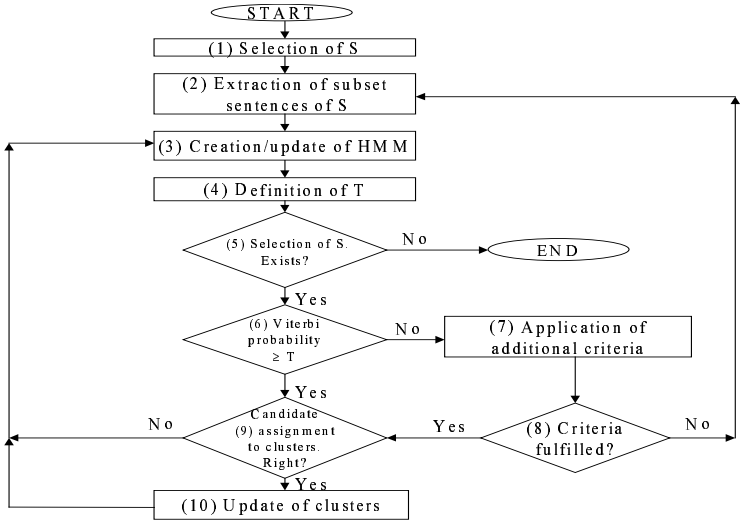

Figure 1. *The flow chart of the algorithm.*

At first a set of sentences is selected to train the initial HMM. For every new sentence *S* (preferably the longest, as it will be explained in section 2.4), the Viterbi algorithm is activated, to check whether this sentence could be extracted by the current HMM, following one of its paths. The probability assigned to the sentence *S* is compared with a threshold *T*, which is defined for the HMM. The estimation of *T* will be described in detail in section 2.4. (a): If the probability assigned to *S* exceeds

or is equal to *T*, or if a part of *S* fits in an existing HMM path, then unknown observations of *S*, that is words or phrases, are able to match 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 existing states). The transition probabilities are also updated. 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. *S* is included in its subsets. (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. In our tests, phrases were formed by considering words with very strong correlation (e.g. *I would like to*, *identity card*, etc.), that is word tuples, which are strongly recurrent in the language and can be thought as a single lexical entry. These phrases were manually selected. The phrase formulation is the only step where language-dependent knowledge is applied. In order to compare the output of our algorithm with a grammar-based model represented as a finite-state network, we consider the final HMM as a SFSN (see section 2.1 for details).

The type of the HMM we use is discrete. 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 the case 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

$$P(w \mid u) = 1 / n \qquad (1)$$

On the other hand, if the number of times class *w* follows *u* is considered, then

$$P(w \mid u) = N(u, w) / N(u) \qquad (2)$$

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 v belongs to a class $C(v)$, which has n members, then the probability of this word/phrase in the class is

$$P(v \mid C(v)) = 1 / n \tag{3}$$

In the latter case

$$P(v \mid C(v)) = N(v) / N(C(v)) \tag{4}$$

where $N(v)$ is the number of occurrences of word/phrase v and $N(C(v))$ the number of occurrences of class $C(v)$, that is the sum of the number of occurrences of its members.

2.3. Training data

A set of sentences is used as input to the algorithm. These sentences can be derived from simulation experiments (Wizard of Oz). They can also be generated by the use of the system itself for the analysis-recognition stage. That is, utterances spoken by users are recorded and transcribed, in order to be used for extending and improving the current language models. Our algorithm takes the set of sentences for granted, regardless of whether they are derived from simulation experiments, from the system itself, from the application grammar, are manually created or are produced by a combination of these methods. However, as it will be shown in the tests carried out, the best language model for recognition is obtained, by mixing sentences taken from the use of the system with grammatically correct sentences. On the other hand, in the NLG process the best model is the one that is based on grammatically correct sentences. There is no requirement for a grammar but if a simple grammar is available it could be used for generating sentences that would be directed, as input, to our algorithm. This is what we did in section 3 in order to compare our models with grammar-based ones under exactly the same conditions. In free phrase-order languages such as Greek, a chunker could be used for the split of sentences into phrases and their ordering so that more training data is available.

2.4. Initialisation

The initialisation stage includes steps 1 to 4, as depicted in Figure 1. The longest sentence S is selected, that is the one with the biggest number of words/phrases (step 1), which will be used together with its subset sentences to train the initial HMM. As it has already been mentioned, subset sentences of S are the sentences, all the words/phrases of which are contained in sentence S. There are two cases: in the first case where the word/phrase order is retained (WPO – Word/Phrase Order), if S is the sequence of words $v_1, v_2, v_3, \ldots, v_n$ then a subset sentence of S could have the

form v_i, v_j, v_k, ..., v_m $1 \le i < j < k < ... < m \le n$. In the second case where the word/phrase order does not pose a constraint (NWPO – No Word/Phrase Order), the subset sentences of S have the form v_i, v_j, v_k, ..., v_m $1 \le i, j, k, m \le n$. If $S1$ is a candidate sentence to be included in the subset sentences of S, and if a word/phrase appears n times in S and m times in $S1$, then a restriction that applies to both cases (WPO and NWPO) is that $m \le n$ (step 2). By using the longest sentence, we ensure that more sentences become subset sentences directly without applying Viterbi. Thus the total computation time for the construction of the final HMM is reduced significantly, especially when much training data is involved. The gain is higher in the WPO case because in every iteration fewer sentences are selected as subset sentences of S due to the word/phrase order constraint and therefore more iterations are required. Nevertheless, the computational cost is not so important since the procedure is offline. That is the HMMs are not created online during the recognition and generation stages.

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) to (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 (step 3).

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 (step 4). 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, and 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 e.g. if v is the unknown observation and v_1, ..., v_{i-1}, v_i, v_{i+1}, ..., v_n is the observation sequence, the new sequence will become v_1, ..., v_{i-1}, v, v_{i+1}, ..., v_n where $1 \le i \le n-2$ and the Viterbi algorithm will give the second threshold. By replacing two or three observations with the unknown word/phrase observation, the third and fourth thresholds are defined: v_1, ..., v_{i-1}, v, v, v_{i+2}, ..., v_n (if $i = n-2$ then $v_{i+2} = v_n$) and v_1, ..., v_{i-1}, v, v, v, v_{i+3}, ..., v_n (if $i = n-2$ then the 3rd unknown observation v will be v_n). We usually select $i = n-4$ unless $n \le 4$ where i is remodified so that most of the sentences and therefore thresholds can be defined. If $n < 3$ then some thresholds are not considered. However, in practice this does not happen, since there are always training sentences with more words/phrases. Some preliminary experiments have shown that the value of i has no significant effect on the resulting network. We intend to carry out further tests with different values of i, and watch how thresholds are affected.

2.5. Iterative procedure

This stage includes steps 2 to 10, as depicted in Figure 1. The longest sentence *S* of the remaining sentences is selected and transformed into an observation sequence (step 5). The Viterbi algorithm 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 (step 6). If it exceeds or it is equal to the appropriate threshold *T* (Case 1), then a new observation can match one of the existing states (classes), and become a member of this class (move to step 9). In cases where the threshold *T* is not exceeded and only a part and not the whole sentence matches an existing path straightforwardly or by shift (Case 2), the candidate matches between new observations and existing states, may be accepted according to some criteria (step 7). These criteria are 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 these 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 model will come closer to the *n*-gram structure. 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.

If the conditions of steps 6 or 8 are satisfied, we move to step 9 to ensure that an observation will not match a class by mistake. The algorithm gets all the subset sentences of *S*, which contain this observation and computes their probabilities using Viterbi. Therefore step 9 also includes the extraction of subset sentences of *S*, although this is not depicted in Figure 1, because it would complicate the flow chart. If the unknown observation matches the same state in the greatest portion of the subset sentences of *S*, which contain this observation, e.g. more than 70% (in Case 1), it is considered that the observation can become a member of the class and the cluster is updated accordingly (step 10). The percentage is set higher (e.g. 75%) in Case 2 (from step 8), because the criterion should be stricter since the threshold *T* is not exceeded. In order to check if the above percentages are exceeded, the observation and state sequences (derived from the Viterbi algorithm) are compared. 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 percentages are set empirically and the definition of their values is still research in progress. Higher values lead to very strict clusters and more states, while lower ones are responsible for loose clusters and fewer states. From step 10 the procedure continues 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 estimating the new HMM parameters (step 3). 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 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) to (4) taking into account the new data and clusters. On the other hand, if neither of cases 1 or 2 has taken place (conditions of steps 6 and 8 are not satisfied) then the algorithm moves to step 2. If no unknown word/phrase matches an existing state for e.g. more than 70% or 75% of the sentences (condition of step 9 is not satisfied) then the procedure moves to step 3 because the subset sentences of *S* have already been extracted in step 9. From step 2, *S* with its subset sentences is used for updating the HMM parameters (step 3). The unknown words/phrases are considered as new states of the HMM. After the new HMM has been constructed, the thresholds are updated (step 4), the longest of the remaining sentences is selected (step 5), and the steps described above are repeated. The procedure stops when there are no other training sentences (condition of step 5 is not satisfied).

In the sentences used for training, we always add the observations *enter* and *exit* as first and last words/phrases respectively so that the resulting network has only one entrance and one exit states. From now on when we give a sequence of words/phrases that forms a sentence, the first and last observations (*enter*, *exit*) will be omitted, that is if the sentence is *enter*, v_1, v_2, ..., v_n, *exit* it will be denoted as v_1, v_2, ..., v_n. To be more analytical let us have some examples, which explain how the algorithm works in the most common cases and where we can see how the criteria of position, number of words, word order and multiple appearance of a word/phrase apply. Suppose that we have the sentences v_1, v_2, …, v_{n-2}, v_{n-1}, v_n (*S1*), v_{n-1}, v_n, v_1, v_2, ..., v_{n-2} (*S2*) and v_{n-1}, v_{n+1}, v_1, v_2, ..., v_{n-2} (*S3*). All the sentences have the same number of words/phrases. We select *S1* to train the initial HMM since it is the first one in order. In the WPO case *S2* will not comprise a subset sentence of *S1*, therefore Viterbi will be activated. The path v_1, v_2, ..., v_{n-2} is common for both *S1* and *S2*. However, if we add new states only for v_{n-1}, v_n, the resulting network for *S1* and *S2* is the one depicted in Figure 2a. The observation sequence v_{n-1}, v_n appears twice in the same path, which is not allowed. The network of Figure 2a would be correct if *S2* had the form v_{n-1}, v_{n+1}, v_1, v_2, ..., v_{n-2} or equivalently if we only had sentences *S1* and *S3* since in this case the path v_{n-1}, v_{n+1}, would be different from v_{n-1}, v_n. However, now that we have similar paths new states will be added for all the observations of *S2* v_{n-1}, v_n, v_1, v_2, ..., v_{n-2}. In the next iteration, where *S3* will also be considered, v_{n+1} will match v_n (Figure 2b). In the NWPO case *S1* and *S2* will be used to train the initial HMM and when Viterbi is activated for *S3*, v_{n+1} will match v_n (Figure 2c). In order for v_n and v_{n+1} to become members of the same cluster, this match should appear in more than 70% of the available sentences. Figure 2c shows that NWPO introduces loops and increases perplexity but also leads to the construction of more compact networks and to the prediction of more sentences (generalisation), which is very important especially when the training data is limited. E.g. the network of Figure 2c produces the observation sequence v_1, v_2, …, v_{n-2}, v_{n-1},

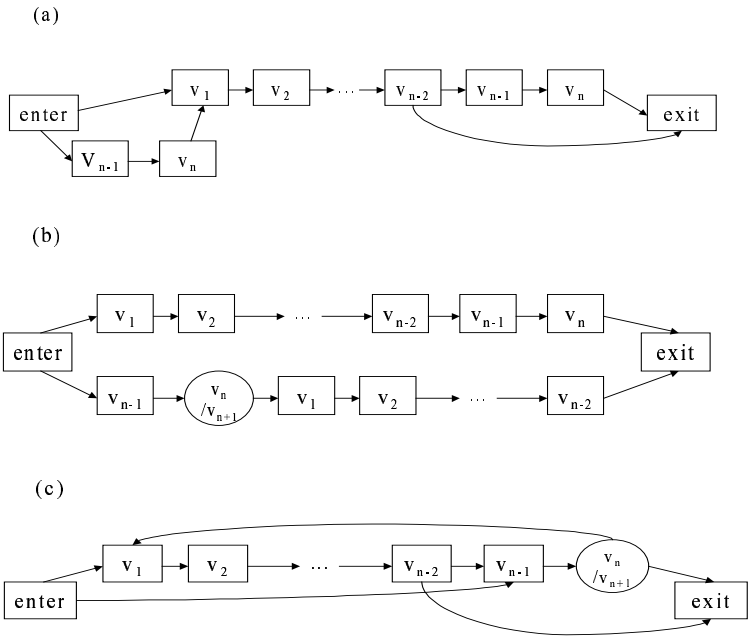

Figure 2. *Example explaining the formulation of classes and how the criteria of word order and multiple appearance of a word/phrase apply.*

v_{n+1}, which is not included in the training sentences. However, if the classes formed are not correct, the error in NWPO will be greater since more paths are involved. In section 3 results are given from experiments carried out with both WPO and NWPO cases. In Figure 2b the complete history is retained whereas in Figure 2c it is not, due to the introduction of loops.

Let us have a more general example now. Suppose that we have the existing HMM path v_1, ..., v_{i-1}, v_i, v_{i+1}, …, v_n (*S1*) and sentence v_{i-m}, …, v_{i-1}, v, v_{i+1}, …, v_{i+p} (*S2*) where $i\text{-}m \geq 1$ and $i+p \leq n$. Then *S2* will be considered to fit in *S1* and v will match with v_i if [($m \geq 2$ or $p \geq 2$) and ($m \neq 1$) and ($p \neq 1$)]. At first we considered 3 neighbour observations instead of 2 but this resulted in failing to match relevant words/phrases. In the same way, if we had sentences v_{i-m}, …, v_{i-1}, v, v, …, v_{i+p+1} (2 unknown observations), v_{i-m}, …, v_{i-1}, v, v, v, …, v_{i+p+2} (3 unknown observations), these sentences would match with *S1* if the previous condition (for 1 unknown observation) is satisfied. Note that the symbol v stands for any unknown observation, which means that 2 or 3 subsequent v do not correspond to 2 or 3 identical words/phrases. If the unknown observations are on the left edge of the sentence v, …, v_{i+m} (1 unknown observation), v, v, …, v_{i+m+1} (2 unknown observations), and v, v, v, …, v_{i+m+2} (3 unknown observations), then it should be $m \geq$

3. Here the constraint is stricter since we have neighbour observations only towards one direction. Finally if the unknown observations are on the right edge of the sentence v_{i-m}, …, v (1 unknown observation), v_{i-m-1}, …, v, v (2 unknown observations), and v_{i-m-2}, …, v, v, v (3 unknown observations), then the constraint should also be stricter, that is $m \geq 3$.

2.6. Additional features

Some additional criteria may also be added so that clusters are correctly formed, e.g. words could be 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. If used, the above criteria can be considered as language-dependent features of the algorithm. These additional constraints (apart from POS) have been taken into account in tests and have resulted in improved performance. Nevertheless, the clusters that the algorithm produces are correctly formed even if these criteria are not considered.

In case we have created our model and new data appears, the HMM is not built from scratch using all the training data (old together with new set of sentences). That is the HMM created with the old set of sentences is used as the initial model in the training procedure. The new data will be used to update this model. In the general case the updated model will not be exactly the same as the model derived from using all the data (old and new). The structure of the two HMMs could be a little different since the final model depends on the available data and their order in the training set. Therefore in the updated model some words/phrases may have not been clustered because the data would not have exceeded the 70% or 75% thresholds. On the other hand, if all the data is used from the beginning, the probability that correct clusters are formed is greater. Thus a model derived from using all the available data is supposed to perform better than the one built in more steps, with 2 or more sets of sentences. However, this is not always the case and depends on the specific data set.

3. Evaluation

3.1. Analysis-recognition stage

In order to test our algorithm we used data from 3 different spoken language systems: ACCeSS (a system for automating the call center services of a car insurance company, EU project LE-11802), IDAS (an Interactive telephone-based Directory Assistance Services system, EU project LE-48315), and a call-routing spoken dialogue system developed by the Greek company Knowledge S.A. We used data from 38 dialogue states of ACCeSS, 7 of IDAS and 4 of the call-routing

system, 49 in total.

Three sets of experiments were carried out. In the first one (Test 1), we consider as training data for our algorithm, the sentences derived from the grammars of the 3 applications. In the second one (Test 2), the training data is derived from the use of the system itself. Finally, in the third experiment (Test 3), the training sentences are data derived from grammars mixed with sentences derived from the use of the system. The total size of the corpora used was 12.3 Mbytes. The number of sentences derived from the grammars varied from 122 to 42609, according to the dialogue state, with an average value of 2642.79. The number of sentences derived from the use of the system varied from 390 to 4165, according to the dialogue state, with an average value of 1834.48. The size of the vocabulary was between 21 and 145 words, again according to the dialogue state, with an average value of 42.79. This high variance in the number of sentences and vocabulary size arises from the fact that the 49 networks are not equivalent in structure. A high number of sentences and/or vocabulary size for a dialogue state does not necessarily entail a high complexity. That is, there could be more difficulties in forming correct clusters and networks in a dialogue state with limited number of sentences and vocabulary size than in another state where the number of sentences and words is high.

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 E1, equations (1) and (3) were used to compute the transition and within class probabilities respectively. In the latter case (E2), we applied equations (2) and (4). As it was mentioned in section 2.2, phrases were manually selected. When we extracted the phrases for our training set, we modified the word-based 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 formed clusters. We define as C the number of correct matches between words or phrases, formed by our method, T the total number of matches formed by using our algorithm, and O the total number of correct matches, which can be derived from the training data. Then:

$$Precision = C / T \quad \text{and} \quad Recall = C / O$$

Let us take an example: if we have words *speak*, *talk*, *communicate* and *contact*, which must be members of the same cluster, then the correct matches we expect to get are *speak-talk*, *speak-communicate* and *speak-contact*. If our algorithm forms matches *speak-talk*, *speak-communicate*, *speak-initiate* and *speak-listen* then T=4, C=2 (*speak-talk speak-communicate*) and O=3 (*speak-talk*, *speak-communicate*, *speak-contact*). Thus the precision is 0.5 and the recall is 0.66. If we considered clusters instead of matches then we would have C=0 since the cluster formed by our algorithm (speak, talk, communicate, initiate, listen) is not correct, T=1 since we have one cluster and O=1, which is (speak, talk, communicate, contact). Thus both the precision and recall would be 0 although some matches formed by the algorithm

are correct. Therefore a definition of precision and recall based on full clusters, as a metric for the algorithm's correctness, would be misleading. We should also state here that we took clusters such as names, car models, cities etc., for granted. That is we were interested only in forming clusters of relevant words/phrases, e.g. *want*, *need*, *desire* etc., since this is the most difficult and useful case. Classes of names, cities etc. do not need such algorithms to be created. They are already formed in databases. 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.

	Test 1		**Test 2**		**Test 3**	
	WPO	**NWPO**	**WPO**	**NWPO**	**WPO**	**NWPO**
	Precision					
W-E1	0.97	0.96	0.93	0.93	0.96	0.96
W-E2	0.98	0.97	0.94	0.93	0.97	0.96
P-E1	0.97	0.97	0.94	0.94	0.97	0.95
P-E2	0.98	0.97	0.95	0.95	0.97	0.96
	Recall					
W-E1	0.77	0.78	0.74	0.75	0.76	0.76
W-E2	0.77	0.77	0.74	0.74	0.75	0.75
P-E1	0.77	0.79	0.75	0.75	0.76	0.76
P-E2	0.76	0.78	0.73	0.74	0.75	0.76

Table 1. *Precision and recall values for the formed classes.*

Test 1 aims at comparing grammar-based networks with our models under the same conditions that is with exactly the same training data. That is, we use as input to the algorithm the sentences, which are derived from the grammar-based networks. The appropriate grammar is loaded according to the dialogue system and the dialogue state. Thus the output of our algorithm is compared with the grammar-based network in each one of the 49 dialogue states.

In Table 1, the precision and recall values are depicted. Computing the average is not a very accurate metric in our case since as it has already been mentioned the 49 networks are not equivalent in structure. However, it is indicative of the efficiency of our method. Sometimes an E1 network can have different precision and recall from the corresponding E2 network. We have observed that often the E2 networks have higher precision but lower recall than the E1 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-based (P) networks generally outperform word-based (W) ones because the use of phrases allows for matches between expressions with the same meaning that would not take place if their constituent words were considered.

The language model can be evaluated through its perplexity on test data, which is given by the following equation:

$$PP = [Pr(v_1 \ldots v_n)]^{-1/N}$$

which is equal to

$$PP = \exp(-\frac{1}{N} \sum_{n=1}^{N} \log Pr(v_n \mid v_1 \ldots v_{n-1}))$$

The grammar-based networks we use are deterministic. The same applies for the models derived from our algorithm (see section 2.1 for details), and bigrams. Thus following the paths will not lead to an ambiguous node selection. Table 2 depicts the average increase (%) in perplexity of our models compared to the grammar-based ones and the average reduction (%) compared to bigrams and class-based bigrams. We have formed class-based bigrams using the clustering algorithm proposed by [KNE 93]. Perplexity in the grammar-based networks is smaller than in the ones derived from our algorithm. This is because the grammar-based networks fit data exactly. However, they do not generalise, and therefore they are not robust against utterances not included in the training data. According to the experiments, E2 networks have lower perplexity than E1 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. The average perplexity reduction (%) of our models compared to bigrams is greater than the one we get by comparing our networks against class-based bigrams.

	Test 1					
	Perplexity Increase vs. grammars (%)		**Perplexity Reduction vs. bigrams (%)**		**Perplexity Reduction vs. class-bigrams (%)**	
	WPO	**NWPO**	**WPO**	**NWPO**	**WPO**	**NWPO**
W-E1	7.34	8.57	17.18	16.07	11.77	10.59
W-E2	7.22	8.25	17.29	16.36	11.88	10.90
P-E1	6.89	8.18	17.86	16.70	12.83	11.60
P-E2	**6.81**	7.92	**17.93**	16.94	**12.91**	11.85

Table 2. *The perplexity (%) in the networks derived from our algorithm compared to grammar-based ones, bigrams and class-based bigrams (Test 1).*

In Test 2, we consider as training sentences data derived from the use of the system itself, to compare our models with bigrams and class-based bigrams. The reason is that the power of bigrams and class-based bigrams arises from the fact that they give reliable estimations when trained with real data. Thus it would not be reasonable to compare our models with bigrams and class-based 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. In our networks, bigrams and class-based bigrams, all based on phrases, the sentences used for testing will first be split into phrases and then perplexity will be estimated. Since the test data may contain events not seen in the training sentences, smoothing techniques should be applied. We use the Witten-Bell discounting scheme [WIT 91; PLA 93] because according to [VAR 99], it was experimentally compared to other classical methods leading to a significant decrease in test-set perplexity. We consider only the occurrences of the specific node and not of the word or phrase associated with it, because the same word/phrase may appear in more than one nodes. For events that have been seen the conditional probability $P(v \mid x)$ of observation v in the context x is:

$$P(v \mid x) = \frac{c_{v|x}}{n_x + r_x}$$

where $c_{v|x}$ is the number of times word/phrase v occurred in context x, n_x is the total number of occurrences of words/phrases in that context

$$n_x = \sum_{v_i} c_{v_i|x}$$

and r_x is the number of different words that occurred in context x. The probability of a previously unseen word/phrase o occurring in that context x is given by:

$$P(o \mid x) = \frac{r_x}{n_x + r_x}$$

Table 3 shows the average perplexity reduction in our models compared to bigrams and class-based bigrams. The perplexity reduction vs. bigrams and class-based bigrams is a little higher in Test 1 compared to Test 2. A reasonable explanation would be that the performance of bigrams and class-based 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 formed matches 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.

In Test 3, we consider as training sentences data derived from grammars mixed with sentences derived from the use of the system. Table 4 shows the perplexity

	Test 2			
	Perplexity Reduction vs. bigrams (%)		Perplexity Reduction vs. class-bigrams (%)	
	WPO	**NWPO**	**WPO**	**NWPO**
W-E1	15.28	13.39	9.61	7.59
W-E2	15.42	13.63	9.76	7.84
P-E1	15.55	14.18	9.88	8.41
P-E2	**15.69**	14.22	**10.03**	8.46

Table 3. *The average perplexity reduction (%) in the networks derived from our algorithm compared to bigrams and class-based bigrams (Test 2).*

	Test 3			
	Perplexity Reduction vs. bigrams (%)		Perplexity Reduction vs. class-bigrams (%)	
	WPO	**NWPO**	**WPO**	**NWPO**
W-E1	15.71	14.20	10.46	8.85
W-E2	15.85	14.44	10.61	9.11
P-E1	16.02	14.57	10.76	9.22
P-E2	**16.15**	14.91	**10.90**	9.58

Table 4. *The average perplexity reduction (%) in the networks derived from our algorithm compared to bigrams and class-based bigrams (Test 3).*

reduction. The perplexity reduction is lower compared to the values of Test 1 but higher compared to Test 2. Again smoothing is applied. Table 1 depicts the average precision and recall values for the formed clusters. 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.

According to the tests, the best language model for the recognition stage is derived from mixing grammatically correct and spontaneous sentences (Test 3). In this case phrase-based networks (WPO-E2) give the lowest perplexity values. Actually the perplexity reduction compared to bigrams and class-based bigrams is greater in Test 1, but in this case the output models would not be robust to spontaneous speech since they are trained only with grammatically correct data.

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 1200 recordings spoken by real users, corresponding to

the system prompt "*Who would you like to speak with?*". The speech recogniser we used was built with the HTK Hidden Markov Models toolkit. In order to train the recogniser we used the SpeechDat-II Greek telephone database.

In Table 5, we can see the recognition accuracy for the grammar-based (G) networks, the ones derived from our algorithm, bigrams (2g) and class-based bigrams (Cl2g). In each cell, the first number shows the % word accuracy and the second the % keyword accuracy. The % word accuracy is given by the following equation:

$$\%\ \text{Word}\quad \text{Accuracy} \quad = \frac{N - D - S - I}{N} \times 100\% \qquad (5)$$

where N is the total numbers of words in the reference transcriptions, D stands for the number of deletions, S for the number of substitutions and I for the number of insertions. The keyword accuracy is the percentage of the sentences where the keyword (*name*) was recognised correctly. In this dialogue state we have only one keyword. However, there are other dialogue nodes with multiple keywords.

	Word/Keyword Recognition Accuracy (%)					
	Test 1		**Test 2**		**Test 3**	
W-G	40.35/75.50					
P-G	40.59/75.67					
	WPO	**NWPO**	**WPO**	**NWPO**	**WPO**	**NWPO**
W-E1	49.02/78.75	52.48/79.33	48.22/78.00	51.36/78.33	52.75/79.58	54.03/80.08
W-E2	51.17/79.17	53.21/79.67	49.55/78.42	52.11/78.92	53.80/79.92	55.29/80.75
P-E1	49.28/79.00	52.84/79.58	48.10/78.08	52.43/78.42	52.91/79.42	56.33/80.17
P-E2	51.61/79.33	**54.09/79.91**	50.46/78.67	**53.67/79.33**	54.10/80.42	**57.64/81.08**
W 2g	45.20/75.25		44.85/76.42		46.31/77.00	
P 2g	46.18/75.33		45.18/76.67		46.88/77.25	
WCl2g	51.13/77.67		50.32/78.42		53.88/79.92	
PCl2g	52.28/77.50		50.69/78.33		54.05/80.00	

Table 5. *Word/keyword recognition accuracy (%).*

The networks derived from our algorithm 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 to spontaneous speech. The NWPO case gives the best recognition accuracy. Therefore the NWPO models are more robust to spontaneous speech since their stochastic features supersede their grammatical structure, as opposed to the WPO case. The columns correspond to the three methods of building the models according to 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, our models, bigrams and class-based bigrams are considered, the word recognition accuracy improvement

for our models compared to grammar-based networks, bigrams and class-based bigrams is 17.05%, 10.76% and 3.59% respectively. In the same way, the gain in keyword recognition performance is 5.41% compared to grammar-based networks, 3.83% compared to bigrams and 1.08% compared to class-based bigrams.

3.2. Generation stage

Evaluating a NLG system is, unfortunately, a difficult task because it is hard to appraise qualitative aspects [GAL 01; HUM 01]. The evaluation of the NLG component in a spoken dialogue system is even more problematic, since it is hard to separate it from other components of the system, especially the text-to-speech engine [OH 00; GAL 01]. Since there is no reference data, we cannot use the Generation String Accuracy (GSA) metric proposed by [BAN 00], which measures the correspondence between the actual models and the desired/target models. GSA is similar to the standard accuracy shown in equation (5). The difference is that it treats the deletion of a token at one location in the string, and the insertion of the same token at another location in the string, as one single movement error. Nevertheless, the results we got from the analysis-recognition stage evaluation could give us some insight about the potential of our method in generating valid system responses.

As in the recognition stage, it is very crucial that the precision is high so that no ill-formed clusters are created, which will result in generating ungrammatical responses. Table 1 shows that E2 networks always have higher precision than E1 ones, the WPO case gives higher precision values than the NWPO case, and phrase-based (P) networks generally outperform word-based (W) ones. Moreover, the networks trained with grammatically correct sentences (Test 1) have the highest precision values. In the same way, if we compare tables 2, 3 and 4, we will find that E2 networks (WPO case), especially phrase-based (P) ones, in Test 1, have the lowest perplexity, which entails that their grammatical structure supersedes their stochastic features.

From the above, it is concluded that, in the NLG process, a phrase-based network (WPO-E2), trained with grammatically correct sentences (Test 1), is expected to give the best results. To prove this point, we carried out a simple test for the dialogue state of *identity card numbers*. We extracted all the sentences from the phrase-based grammar network that we had for this particular dialogue state. The grammatically correct sentences were 202 in total, and they were directed as input to our algorithm (WPO-E2). The identity card number was treated as a single word that is we did not take into account different sequences of letters and numbers, which would result in too many sentences. The produced HMM was ready from Test 1 in section 3.1. Then we extracted all the possible outputs of our model. They were 267 sentences in total, 202 from the training data and 65 new sentences produced due to the generalisation effect. The 65 new sentences were checked manually to see whether they were grammatically correct or not. The result was that 47 of them were valid sentences. Therefore the proportion (%) of the correct sentences to the total

number of sentences, generated by our algorithm, is (202 + 47) * 100 / (202 + 65) = 93.25 %, and the percentage (%) of the correct new sentences to the total new sentences derived from our model, is 47 * 100 / 65 = 72.30 %. Although the precision value is high (0.98), which means that most of the clusters are correctly formed, the percentage of the new correct sentences to the total new sentences is quite lower (72.30%) because an ill-formed cluster is not the only source of errors. Matches may be correct but result in, e.g. associating words with inappropriate prepositions, and thus produce grammatically wrong sentences. The test carried out for this particular node gave promising results but it could not be performed for all the dialogue states, due to the enormous manual effort that would be required for checking the sentences. Therefore for more extensive results we will have to wait until the algorithm is implemented on a real system. So far, our algorithm has been incorporated in the recognition module of our dialogue systems. We intend to incorporate it in the NLG module as well, so that we test its efficiency, as far as NLG is concerned, on real conditions.

5. Conclusions

In this paper, we presented a method for creating SFSNs for language modelling both in the recognition and the generation stages of dialogue systems. Word/phrase classes are created automatically during the construction of a HMM. The resulting HMM incorporates linguistic knowledge, and information provided by statistical estimations, and allows for variable history sizes with no specific upper limit. Adjustment of parameters may lead to a model where stochastic features supersede grammatical structure (recognition stage) and the contrary (generation stage). Our method was tested using data from 3 different spoken dialogue systems. The tests carried out, proved the efficiency of our algorithm regarding precision and recall values for the formed clusters, perplexity, and recognition performance. Moreover, the above values of precision and perplexity make our models promising for the task of generating valid responses without the need of grammars.

Future work will focus on exploring the amount of data necessary for constructing efficient networks. For example, a matter that should be investigated is the analogy of sentences derived from grammars to real data, so that the best networks are produced. Moreover, enhanced smoothing techniques will be investigated, e.g. the delimited smoothing technique [VAR 00].

References

[AXE 00] AXELROD S., “Natural Language Generation in the IBM Flight Information System”, *In Proc. ANLP-NAACL*, Seattle, WA, 2000, pp. 21-26.

[BAN 00] BANGALORE S., RAMBOW O., “Exploiting a probabilistic hierarchical model for generation”, *In Proc. COLING*, Saarbrücken, Germany, 2000.

[BAT 97] BATEMAN J., "Enabling technology for multilingual natural language generation: The KPML development environment", *Natural Language Engineering,* 3:15-55, 1997.

[BEL 98] BELLEGARDA J.R., "Exploiting both Local and Global Constraints for Multi-Span Statistical Language Modeling", *In Proc. ICASSP*, Seattle, WA, 1998.

[BOR 97] BORDEL G., VARONA A., TORRES M. I., "K-TLSS(S) Language Models for Speech Recognition", *In Proc. ICASSP*, Munich, Germany, 1997, Vol. 2, pp. 819-822.

[BRO 92] BROWN P.F., DELLA PIETRA V.J., DESOUZA P.V., LAI J.C., MERCER R.L., "Class-based *n*-gram models of natural language", *Computational Linguistics*, 1992, 18:467-479.

[BUS 98] BUSEMAN S., HORACEK H., "A Flexible Shallow Approach to Text Generation", *In Proc. INLG*, Niagara-on-the-Lake, Canada, 1998, pp. 238-247.

[CHE 99] CHELBA C., JELINEK F., "Recognition Performance of a Structured Language Model", *In Proc. Eurospeech*, Budapest, Hungary, 1999, Vol. 4, pp. 1567-1570.

[ECK 96] ECKERT W., GALLWITZ F., NIEMANN H., "Combining stochastic and linguistic language models for recognition of spontaneous speech", *In Proc. ICASSP*, Atlanta, GA, 1996, pp. 423-426.

[FAW 92] FAWCETT R.P., "The state of the craft in computational linguistics: A generationist's viewpoint", *Technical Report COMMUNAL Working Papers No. 2*, Cardiff Computational Linguistics Unit, University of Wales.

[FON 95] FONG E., WU D., "Learning restricted probabilistic link grammars", *IJCAI Workshop on New Approaches to Learning for Natural Language Processing*, Montreal, Canada, 1995.

[GAL 01] GALLEY M., FOSLER-LUSSIER E., POTAMIANOS A., "Hybrid Natural Language Generation for Spoken Dialogue Systems", *In Proc. Eurospeech*, Aalborg, 2001.

[GAR 01] GARDENT C., THATER S., "Generating with a Grammar Based on Tree Descriptions: a Constraint-Based Approach", *In Proc. of ACL*, Toulouse, France, 2001.

[GEO 00] GEORGILA K., FAKOTAKIS N., KOKKINAKIS G., "Building stochastic language model networks based on simultaneous word/phrase clustering", *In Proc. ICSLP*, Beijing, China, 2000, Vol. 1, pp. 122-125.

[GEO 01] GEORGILA K., FAKOTAKIS N., KOKKINAKIS G., "Efficient Stochastic Finite-State Networks for Language Modelling in Spoken Dialogue Systems", *In Proc. Eurospeech*, Aalborg, Denmark, 2001, Vol. 1, pp. 247-250.

[HOP 79] HOPCROFT E., ULLMAN J.D., Introduction to automata theory, languages and computation, Addison-Wesley 1979.

[HOV 88] HOVY E.H., Generating Natural Language under Pragmatic Constraints, Lawrence Erlbaum, Hillsdale, New Jersey.

[HUM 01] HUMPHREYS K., CALCAGNO M., WEISE D., "Reusing a Statistical Language Model for Generation", *In Proc. ACL-EWNLG*, Toulouse, France, 2001, pp. 86-91.

[JAR 93] JARDINO M., ADDA G., "Automatic Word Classification Using Simulated Annealing", *In Proc. ICASSP*, Minneapolis, MN, 1993, Vol. 2, pp. 41-44.

[JUR 95] 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", *In Proc. ICASSP*, Detroit, MI, 1995, Vol. 1, pp. 189-192.

[KNE 93] KNESER R., NEY H., "Improved clustering techniques for class-based statistical language modelling", *In Proc. Eurospeech*, Berlin, Germany, 1993, pp. 973-976.

[KNI 95] KNIGHT K., HATZIVASSILOGLOU V., "Two-Level, Many-Paths Generation", *In Proc. of ACL*, Boston, Mass., 1995.

[LAN 98] LANGKILDE I., KNIGHT K., "Generation that exploits corpus-based statistical knowledge", *In Proc. COLING-ACL*, Montreal, Canada, 1998.

[LAU 93] LAU R., ROSENFELD R., ROUKOS S., "Trigger-Based Language Models: A Maximum Entropy Approach", *In Proc. ICASSP*, Minneapolis, MN, 1993.

[LEV 97] LEVIN E., PIERACCINI R., "A Stochastic Model of Computer-Human Interaction for Learning Dialogue Strategies", *In Proc. Eurospeech*, Rhodes, Greece, 1997.

[LLO 95] LLOYD-THOMAS H., WRIGHT J.H., JONES G.J.F., "An integrated grammar/bigram language model using path scores", *In Proc. ICASSP*, Detroit, MI, 1995.

[MET 93] METEER M., ROHLICEK J.R., "Statistical language modeling combining *n*-gram and context-free grammars", *In Proc. ICASSP*, Minneapolis, MN, 1993, Vol. 2, pp. 37-40.

[MOH 97] MOHRI M., "Finite-state transducers in language and speech processing", *Computational Linguistics*, 1997, 23:2.

[MOO 93] MOORE J., PARIS C., "Planning text for advisory dialogues", *Computational Linguistics*, 19:651-694, 1993.

[MOO 95] MOORE R., APPELT D., DOWDING J., GAWRON J.M., MORAN D., "Combining linguistic and statistical knowledge sources in natural-language processing for ATIS", *In Proc. of Spoken Language Systems Technology Workshop*, Austin, 1995, pp. 261-264.

[NAS 99] NASR A., ESTÈVE Y., BÉCHET F., SPRIET T., DE MORI R., "A language model combining n-grams and stochastic finite state automata", *In Proc. Eurospeech*, Budapest, Hungary, 1999, pp. 2175-2178.

[OH 00] OH A.H., RUDNICKY A.I., "Stochastic Language Generation for Spoken Dialogue Systems", *In Proc. ANLP-NAACL*, Seattle, WA, 2000, pp. 27-32.

[PER 96] PEREIRA F.C., SINGER Y., TISHBY N., "Beyond word n-Grams", *Computational Linguistics*, 1996, Vol. 22.

[PLA 93] PLACEWAY P., SCHWARTZ R., FUNG P., NGUYEN L., "The estimation of powerful language models from small and large corpora", *In Proc. ICASSP*, Minneapolis, 1993.

[POP 97] POPOVICI C., BAGGIA P., "Language modelling for task-oriented domains", *In Proc. Eurospeech*, Rhodes, Greece, 1997, Vol. 3, pp. 1459-1462.

[RAM 92] RAMBOW O., KORELSKY, T., "Applied Text Generation", *In Proc ANLP*, Trento, Italy, 1992, pp. 40-47.

[RAM 01] RAMBOW O., ROGATI M., WALKER M.A., "Evaluating a Trainable Sentence Planner for a Spoken Dialogue System", *In Proc ACL*, Toulouse, 2001, pp. 434-441.

[RAT 00] RATNAPARKHI A., "Trainable methods for surface natural language generation", *In Proc. ANLP-NAACL*, Seattle, WA, 2000.

[REI 97] REITHINGER N., KLESEN M., "Dialogue Act Classification Using Language Models", *In Proc. Eurospeech*, Rhodes, Greece, 1997.

[REI 00] REITER E., DALE R., Building Natural Language Generation Systems, Cambridge University Press, 2000.

[RIC 96] RICCARDI G., PIERACCINI R., BOCCHIERI E., "Stochastic automata for language modeling", *Computer Speech and Language*, 1996, vol. 10, pp. 265-293.

[RIE 96] RIES K., BUO F.D., WAIBEL A., "Class phrase models for language modeling" *In Proc. ICSLP*, pp. 398-401.

[SMA 96] SMAÏLI K., CHARPILLET F., HATON J.P., "Anew algorithm for word classification based on an improved simulated annealing technique", *In 5th International Conference on the Cognitive Science of Natural Language Processing*, 1996.

[STE 99] STENT A., "Content Planning and Generation in Continuous-Speech Spoken Dialogue Systems", *In Proc. of the KI'99 workshop, "May I speak Freely?"*, Bonn, Germany, 1999.

[STO 93] STOLCKE A., OMOHUNDRO S., "Hidden Markov Model induction by Bayesian model merging", *In Advances in Neural Information Processing Systems 5*, 11-18, San Mateo, CA, 1993, Morgan Kaufmann.

[THO 86] THOMASON M.G., GRANUM E., "Dynamic programming inference of Markov networks from finite set of sample strings", *IEEE Trans. On PAMI*, 1986, 8:491-501.

[TSU 98] TSUKADA H., YAMAMOTO H., TAKEZAWA T., SAGISAKA Y., "Reliable utterance segment recognition by integrating a grammar with statistical language constraints", *Speech Communication*, 1998, Vol. 26, pp. 299-309.

[USZ 96] USZKOREIT H., "Language generation", In R.A. Cole, Ed., *Survey of the State of the Art in Human Language Technology*, URL: http://www.cse.ogi.edu/CSLU/HLTsurvey/, 1996, Chapter 4, pp. 161-188.

[VAR 99] VARONA A., TORRES I., "Using Smoothed K-TSS Language Models in Continuous Speech Recognition", *In Proc. ICASSP*, Phoenix, AZ, 1999, Vol. 2, pp. 729-732.

[VAR 00] VARONA A., TORRES I., "Delimited smoothing technique over pruned and not pruned syntactic language models: perplexity and WER", *In Proc. ITRW ASR, Automatic Speech Recognition:Challenges for the new Millennium*, Paris, France, 2000, pp. 69-76.

[WAL 01] WALKER M.A., RAMBOW O., ROGATI M. "Spot: A trainable sentence planner", *In Proc NAACL*, 2001.

[WIT 91] WITTEN I.H., BELL T.C., "The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression", *IEEE Transactions on Information Theory*, July 1991, 37(4):1085-1094.

[WU 99] WU J., KHUDANPUR S., "Combining nonlocal, syntactic and *N*-gram dependencies in language modeling", *In Proc. Eurospeech*, Budapest, Hungary, 1999, pp. 2179-2182.