Interpolation of Stochastic Grammar and Word Bigram Models in Natural Language Understanding

Sven C. Martin, Andreas Kellner, Thomas Portele

Philips Research Labs Aachen, Weißhausstraße 2, D–52066 Aachen, Germany

ABSTRACT

The paper shows the effects of combining a stochastic grammar with a word bigram language model by log-linear interpolation. It is divided into three main parts: The first part derives the stochastic grammar model and gives a sound theoretical motivation to incorporate word dependencies such as bigrams. The second part describes two different algorithmic approaches to the combination of both models by log-linear interpolation. The third part reports attribute error rate (AER) results measured on the Philips corpus of train time table inquiries that show a reduction of up to 9% relative.

1. STOCHASTIC MODEL OF NATURAL LANGUAGE UNDERSTANDING

The Philips Natural Language Understanding (NLU) module is used in automated inquiry systems (AIS), such as train table enquiries [2], to analyze the word sequence of a user utterance. It does not try to find parse trees that cover the whole word sequence but breaks up the sequence into chunks, where each chunk belongs to a semantically meaningful concept. A stochastic context–free grammar is used to derive the word chunk from a concept. The chunking is useful since the spontaneous speech that occurs in dialogue applications is very ungrammatical. Thus, a robust NLU model concentrates on the useful parts of a user utterance. Other recent works also employ some kind of chunking, e.g. [6, 9].

The stochastic model of the Philips NLU module was developed by H. Aust in [1, p. 81]. Here, we show that this model can be derived from *Bayes' decision rule.* This derivation gives a sound theoretical motivation to incorporate word dependencies such as bigrams. Bayes' decision rule finds the most likely concept sequence $\hat{K} = \hat{k}_1, \ldots, \hat{k}_s$, given the sequence $O = o_1, \ldots, o_t$ of acoustic observations. The derivation of the concept sequence K does not directly depend on the acoustic observations O but on a word sequence $W = w_1, \ldots, w_N$ derived from O as an intermediate result:

$$\begin{aligned} \hat{K} &= \operatorname*{argmax}_K \; p(K|O) \\ &= \operatorname*{argmax}_K \; \sum_W \; p(K, W, O) \\ &\approx \operatorname*{argmax}_K \; \max_W \; p(K, W, O) \\ &= \operatorname*{argmax}_K \; \max_W \; p(O|K, W) \cdot p(W|K) \cdot p(K) \quad . \quad (1) \end{aligned}$$

Thus, three different probability distributions are employed. They are now described in detail.

Acoustic Probability. The acoustic probability $p(O|K, W) \approx p(O|W)$ is computed by the speech recognition module. The speech recognition module delivers a word graph with the **acoustic score** (i.e. log–probability) of each word w in that word graph. This word graph is the input to the NLU module.

Concept Probability. The concept sequence K is derived from a stochastic grammar dividing the word sequence W into chunks, with $\bar{w}_i$ being the ith chunk in sequence, and where each chunk is a sequence of words corresponding to a concept. Each concept is represented by a nonterminal k_i. The sequence $K = k_1^s$ of chunks, represented by the concept nonterminals k_i, is modeled by **concept bigram probabilities** $p(k_i|k_{i-1})$:

$$\begin{aligned} p(K) &= \prod_{i=1}^{s} p(k_i|k_1^{i-1}) \\ &\approx \prod_{i=1}^{s} p(k_i|k_{i-1}) \quad . \quad (2) \end{aligned}$$

Grammar Probability. The chunk $\bar{w}_i$ can be derived from its corresponding concept k_i:

$$\begin{aligned} p(W|K) &= \prod_{i=1}^{s} p(\bar{w}_i|\bar{w}_1^{i-1}, \; k_1^s) \\ &\approx \prod_{i=1}^{s} p(\bar{w}_i|k_i) \quad . \quad (3) \end{aligned}$$

Note that the probability now depends only on the concept k_i and no longer on the predecessor words $\bar{w}_1^{i-1}$ nor on the concept sequence K as a whole. The concepts k_i are a subset of the grammar nonterminals. For each nonterminal k, there exists a set $R(k)$ of rules. Each rule $r \in R(k)$ represents a derivation from k to another sequence of terminals and nonterminals with a **rule probability** $p(r|k)$:

$$\begin{aligned} p(\bar{w}_i|k_i) &= \sum_{r \in R(k_i)} p(\bar{w}_i, r|k_i) \\ &= \sum_{r \in R(k_i)} p(\bar{w}_i|r, k_i) \cdot p(r|k_i) \\ &\approx \max_r p(\bar{w}_i|r, k_i) \cdot p(r|k_i) \quad . \end{aligned}$$

Assume that rule r is of the form $k_i ::= \hat{k}_1 ... \hat{k}_m$ leading to a further segmentation of chunk $\bar{w}_i$ into chunks $\bar{w}_i = \hat{\bar{w}}_1 ... \hat{\bar{w}}_m$, where the phrase $\hat{k}_1^m$ consists of either terminals or nonterminals of the stochastic grammar. The probability $p(\bar{w}_i|r, k_i)$ is further subdivided:

$$\begin{aligned} p(\bar{w}_i|r, k_i) &= p(\hat{\bar{w}}_1^m|\hat{k}_1^m, k_i) \\ &= \prod_{l=1}^{m} p(\hat{\bar{w}}_l|\hat{\bar{w}}_1^{l-1}, \hat{k}_1^m, k_i) \\ &\approx \prod_{l=1}^{m} p(\hat{\bar{w}}_l|\hat{k}_l) \quad . \end{aligned}$$

Again the dependency of the word chunk $\hat{\bar{w}}_l$ on its predecessor words $\hat{\bar{w}}_1^{l-1}$ and on the phrase $\hat{k}_1^m$ as a whole is suppressed, further the dependency on the parent nonterminal k_i. If $\hat{\bar{w}}_l = \hat{k}_l$, i.e. the chunk $\hat{\bar{w}}_l$ is matched by a corresponding terminal sequence, then $p(\hat{\bar{w}}_l|\hat{k}_l) = 1$. Else, the modeling of $p(\bar{w}_i|k_i)$ is applied in a recursive manner to $p(\hat{\bar{w}}_l|\hat{k}_l)$. As a result, the chunk $\bar{w}_i$ is finally derived with a probability that is the product of all involved rule probabilities:

$$p(\bar{w}_i|k_i) \approx \prod_{(k,r) \in \{k_i \stackrel{*}{\Rightarrow} \bar{w}_i\}} p(r|k)^{c(k,r|k_i \stackrel{*}{\Rightarrow} \bar{w}_i)} \quad , \tag{4}$$

where $(k, r) \in \{k_i \stackrel{*}{\Rightarrow} \bar{w}_i\}$ represents the rules r with nonterminal k in that derivation of $\bar{w}_i$ from k_i that maximizes $p(\bar{w}_i|k_i)$, and $c(k, r|k_i \stackrel{*}{\Rightarrow} \bar{w}_i)$ their number of occurrences in that derivation.

Filler Probability. In the stochastic model of Eq. (1), each word belongs to at least one concept. However, in real utterances, there are usually some words that cannot be covered by the grammar and thus remain as "filler" words in–between the chunks. Thus, the word sequence W is subdivided into s concept chunks $\bar{w}_i$ interleaved with q filler chunks $\bar{f}_z$, where $\bar{w}_i$ is the ith concept chunk and $\bar{f}_z$ the zth filler chunk. A concept chunk may be followed by another concept chunk, but a filler chunk is never followed by another filler chunk.

To incorporate fillers into the model of Eq. (1), a special filler concept is introduced from which the filler chunk $\bar{f}_z$ is derived with probability $p(\bar{f}_z)$. Thus, Eq. (3) changes to

$$p(W|K) \approx \prod_{i=1}^{s} p(\bar{w}_i|k_i) \cdot \prod_{z=1}^{q} p(\bar{f}_z) \quad . \tag{5}$$

$p(K)$ ignores the filler concepts, since these may be randomly distributed all over the word sequence W. Since filler words are by definition out–of–grammar, the probability $p(\bar{f}_z)$ is derived by word–based **filler bigram probabilities** $p(f_n|f_{n-1})$ instead of rule–based probabilities:

$$p(\bar{f}_z) = \prod_{(f_{n-1}, f_n) \in \bar{f}_z} p(f_n|f_{n-1}) \quad ,$$

where (f_{n-1}, f_n) are two consecutive words in the filler chunk $\bar{f}_z$.

2. INTERPOLATION OF GRAMMAR AND WORD BIGRAM

The probability of a derivation of a word chunk from a concept in Eq. (4) is the same as normally used for context–free grammars. To get there from Bayes' decision rule, some dependencies are neglected, especially the word context and the parent nonterminals. It is consequent to say that a context–free rule should have a context–free rule probability. In the past years, however, this has turned out to be a weak point of grammar design. Dependency grammars and history–based grammars, for example, re–introduce these dependencies in recent works, e.g. [3, 4, 5]. The point we want to make here is that adding contextual information is not an ad–hoc countermeasure to improve performance but a requirement by the application of Bayes' decision rule to grammars. This is so far very little (if at all) stressed in the literature on stochastic grammars.

As a first step, a simple but efficient method is to interpolate the grammar probabilities with a word–based n–gram probability. We use bigrams since the training corpora for dialogue applications are usually so small that data sparseness problems can be expected for higher–order n–grams. We use log–linear interpolation which is reported to have a good performance [7]. The general idea is to replace the grammar probability $p(W|K)$ as modeled in Eq. (5) by a new grammar probability $p'(W|K)$ that log–linearly combines $p(W|K)$ with the probability $p(W) = \prod_{n=1}^{N} p(w_n|w_{n-1})$:

$$\begin{aligned} &\log p'(W|K) = \\ &= \lambda \cdot \log p(W|K) + (1 - \lambda) \cdot \log p(W) - \log Z(K) \quad , \end{aligned}$$

where $Z(K)$ is a normalizing factor needed to make $p'(W|K)$ a proper probability distribution. However, its computation is very costly. Experience shows that in an application, $Z(K)$ can usually be dropped without much loss in performance, and we do so here. Consequently, we cannot measure any perplexities, since these require proper probability distributions. There are two competing approaches how to implement the interpolation.

n–best Lists. Here, as a rather ad–hoc approach, not the first best concept sequence $\hat{K}$ is derived by the NLU module, but the n best concept sequences, using the algorithm described in [8], along with their respective NLU model log–probabilities. For each of the n concept sequences, the underlying word sequence is recovered from the word graph, and the respective word sequence log–probabilities are computed, based on the word bigram model. The n concept sequences are re–ranked according to the linear interpolation of the respective NLU model and word bigram model log–probabilities, and the new top concept sequence is the NLU result.

This method restricts the concept sequences for re–ranking to the n best concept sequences according to the NLU model. However, the best sequence according to the interpolated model may well fall outside of this range if n is rather small. For large n, however, the computation is very costly.

Integrated Interpolation (II). Thus, an alternative to the n–best list rescoring is necessary. The general idea of the method proposed here is to use the word bigram scores while doing the search operation for the best concept sequence. Thus, Eq. (5) is changed to

$$\log p(W|K) \approx \tag{6}$$

$$\approx \sum_{i=1}^{s} \left(\lambda \cdot \log p(\bar{w}_i|k_i) + (1-\lambda) \cdot \sum_{w_n \in \bar{w}_i} \log p(w_n|w_{n-1}) \right)$$

$$+ \sum_{z=1}^{q} \left(\mu \cdot \log p(\bar{f}_z) + (1-\mu) \cdot \sum_{w_n \in \bar{f}_z} \log p(w_n|w_{n-1}) \right) .$$

Two different interpolation weights are used, λ for concept chunks, and μ for filler chunks. Using two weights may make sense because the concept (grammar) and the filler probabilities rely on different models, whereas the word bigram model is always the same. Further, Eq. (2) is changed to

$$\log p(K) = \lambda \cdot \sum_{i=1}^{s} \log p(k_i|k_{i-1}) \quad . \tag{7}$$

Both terms should be normalized to yield proper probability distributions. However, as stated above, the normalizing can be dropped without much loss in performance for applications.

3. EXPERIMENTS

Corpus & Grammar. For the experiments, we use the Philips TABA corpus that consists of train table inquiries in German over the telephone. This corpus is divided into a training and a testing subcorpus. To see the effect of varying training corpus sizes, a 1k subcorpus of the training corpus is also used for training. A statistics on the training corpora can be found in Table 1. The corpus named “32K” is the full TABA training corpus. The original TABA test corpus comprises 2 hours of speech. The test corpus is subdivided into two corpora named “DEV” and “EVL”, where the “DEV” corpus is used for optimizing (DEVeloping) the interpolation parameters λ and μ, and the “EVL” corpus as the hard test case (for EVaLuation). A statistics on the DEV and EVL corpora can be found in Table 2.

The vocabulary, comprising all words of the training corpus and the testing corpus, consists of 2 545 words, excluding unknown word tag and sentence end marker. The TABA grammar consists of 34 concepts, 96 non–terminals (including concepts), and 1955 rules.

Experimental Setup. From each training corpus, a concept and filler language model is constructed by applying the untrained grammar to the training sentences and

Table 1: TABA training corpora statistics (no. of words excluding sentence end marker).

identifier	no. of turns	no. of words	filler words [%]
1K	1024	4098	6.6
32K	33081	110216	12.6

Table 2: TABA test corpora statistics (GER – graph error rate, WGD – word graph density, no. of words excluding sentence end marker).

identifier	no. of turns	no. of words	filler words [%]	GER [%]	WGD
DEV	1000	3135	13.6	5.7	53.7
EVL	1278	3836	13.5	5.8	53.9

counting the resulting concept bigrams and filler word bigrams, respectively. Also by applying the untrained grammar to the training sentences, the number of applications for each grammar rule is counted. The NLU model probabilities are, basically, derived from these counts by using relative frequencies. A word bigram language model is also constructed from each training corpus.

With each concept, attributes such as time and train station are associated. The measure of quality for the NLU is the attribute error rate (AER), the percentage of wrongly assigned attributes compared to a reference assignment. The reference attribute assignment is constructed from the transcriptions of the DEV and EVL corpora using the untrained grammar. For each training corpus size (1K and 32K), five models are applied to the test corpora:

- The standard NLU model, as described in Section 1. From a modeling point of view, this is equivalent to setting $\lambda = \mu = 1$ in the interpolated model of Eqs. (6) and (7) (the implementation, of course, is different).
- The best path on the word graph is searched according to the word bigram model and the resulting word sequence is handed to the NLU module instead of the complete word graph. From a modeling point of view, this is equivalent to setting $\lambda = \mu = 0$ in the interpolated model of Eqs. (6) and (7).
- The n–best list model as described in Section 2. For the interpolation factor, all values from 0 to 1 with a step size of 0.05 are computed on the DEV corpus, using a list length of $n = 100$. The best–performing value is used for the EVL corpus.
- The integrated interpolation as described in Section 2. The interpolation factors of Eq. (6) are pooled (i.e. $\lambda = \mu$), and all values from 0 to 1 with a step size of 0.05 are computed on the DEV corpus. The best–performing value is used for the EVL corpus.
- The integrated interpolation as described in Section 2. However, the interpolation factors λ and μ are independent now. All values between (0,0) and (1,1) with

a step size of 0.1 are computed. The best–performing value is used for the EVL corpus.

Experimental Results. The AER results can be seen in Table 3, further the relative computing times $t_{\mathtt{rel}}$ with the time for the standard NLU model set to 1.0 as reference:

- The stochastic NLU model works better than the word bigram model, but the gap closes with corpus size. The results show that on small training corpora, the stochastic NLU model is superior due to its knowledge encoded in the grammar. For larger training corpora, however, the word bigram adapts better to the training material and improves its performance faster than the stochastic NLU model. Note, however, that the TABA grammar is very sophisticated and fine–tuned. On other corpora and grammars, it is observed that at some training corpus size the word bigram model performs even better than the stochastic NLU model. Since parsing is only used for the first best word sequence, the word bigram model is far quicker than the stochastic NLU model.
- For n–best lists, there are slight optimum interpolation factors at $\lambda = 0.65$ for the 1K training corpus and $\lambda = 0.35$ for the 32K training corpus on the DEV corpus. Using 100–best lists on the EVL corpus yields a reduction in AER by 9% rel. for the 1K and by 7% rel. for the 32K training corpus. As further experiments show, using larger n does not improve performance but drastically increases computing time. However, on the fine–tuned TABA grammar, a smaller n down to $n = 10$ can be used with almost no loss in performance.
- The results on the EVL corpus for integrated interpolation show a reduction in AER by 8% rel. with the small and by 3% rel. with the large training corpus for pooled interpolation factors. For separate interpolation factors, the gain is 6% rel. on the small and 7% rel. on the large training corpus. In the first case, the interpolated model was overadapted to the training data. Thus, there is a certain risk at using separate interpolation factors. Here, the integrated interpolation does not increase AER performance compared to n–best lists, despite the larger search space. The explanation is that a search space of 100–best concept sequences suffices for the TABA corpus. Further experiments on other corpora show that this is not always the case, e.g. for highly ambiguous grammars that deteriorate the performance of the stochastic NLU model.

Table 3: AER results for interpolation of stochastic NLU model and word bigram model.

		λ	μ	DEV [%]	EVL [%]	$t_{\mathtt{rel}}$
1K	NLU baseline	—	—	16.2	17.6	1.0
	word bigram	—	—	22.0	20.9	0.2
	100–best	0.65	—	16.1	16.1	2.4
	II–pooled	0.65	—	16.1	16.2	2.1
	II–separate	0.8	0.6	15.8	16.5	2.1
32K	NLU baseline	—	—	13.2	15.1	1.0
	word bigram	—	—	14.1	15.5	0.2
	100–best	0.35	—	12.5	14.1	2.4
	II–pooled	0.35	—	12.5	14.6	2.3
	II–separate	0.5	0.4	12.6	14.0	2.3

- For well–estimated stochastic NLU models, there is not much difference in performance between n–best lists and integrated interpolation.

4. CONCLUSIONS

The following conclusions can be drawn:

- Using log–linear interpolation between stochastic NLU and word bigram models yields a notable reduction in AER, in case of the TABA corpus up to 9% rel., compared to the stochastic NLU model, at a roughly doubled computing time.
- Using separate interpolation factors for concept and filler chunks may improve results but can also backfire.

5. REFERENCES

1. H. Aust. *Sprachverstehen und Dialogmodellierung in natürlichsprachlichen Informationssystemen.* Ph. D. Thesis, RWTH Aachen, University of Technology, Aachen, Germany, 1998.
2. H. Aust, M. Oerder, F. Seide, and V Steinbiss. The Philips automatic train timetable information system. *Speech Communication*, 17(3–4):249–262, Nov. 1995.
3. E. Charniak. A maximum–entropy–inspired parser. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics, Seattle, Washington*, pages 132–139, 2000.
4. C. Chelba. *Exploiting Syntactic Structure for Natural Language Modeling.* Ph. D. Thesis, Johns Hopkins University, Baltimore, MA, 2000.
5. M. Collins. Three generative, lexicalized models for statistical parsing. In *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics, Madrid*, pages 16–23, 1997.
6. J. Gillet and W. Ward. A language model combining trigrams and stochastic context–free grammars. In *5th International Conference on Spoken Language Processing, Sydney*, volume 6, pages 2319–2322, 1998.
7. D. Klakow. Log-linear interpolation of language models. In *Proc. ICSLP*, volume 5, pages 1695–1699, Sydney, Australia, December 1998.
8. B.H. Tran, F. Seide, and V. Steinbiss. A word graph based N-best search in continuous speech recognition. In *Proc. ICSLP*, volume 4, pages 2127–2130, Philadelphia, PA, Oct. 1996.
9. Y.-Y. Wang, M. Mahajan, and X. Huang. A unified context–free grammar and n–gram model for spoken language processing. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, Istanbul*, pages 1639–1642, June 2000.