Automatic Learning of Language Model Structure

Kevin Duh and **Katrin Kirchhoff**
Department of Electrical Engineering
University of Washington, Seattle, USA
{duh,katrin}@ee.washington.edu

Abstract

Statistical language modeling remains a challenging task, in particular for morphologically rich languages. Recently, new approaches based on factored language models have been developed to address this problem. These models provide principled ways of including additional conditioning variables other than the preceding words, such as morphological or syntactic features. However, the number of possible choices for model parameters creates a large space of models that cannot be searched exhaustively. This paper presents an entirely data-driven model selection procedure based on genetic search, which is shown to outperform both knowledge-based and random selection procedures on two different language modeling tasks (Arabic and Turkish).

1 Introduction

In spite of novel algorithmic developments and the increased availability of large text corpora, statistical language modeling remains a difficult problem, particularly for languages with rich morphology. Such languages typically exhibit a large number of word types in relation to word tokens in a given text, which leads to high perplexity and a large number of unseen word contexts. As a result, probability estimates are often unreliable, even when using standard smoothing and parameter reduction techniques. Recently, a new language modeling approach, called *factored language models (FLMs)*, has been developed. FLMs are a generalization of standard language models in that they allow a larger set of conditioning variables for predicting the current word. In addition to the preceding words, any number of additional variables can be included, to represent e.g. morphological, syntactic, or semantic word features. Since such features are typically shared across multiple words, they can be used to obtained better smoothed probability estimates when training data is sparse. However, the space of possible models is extremely large, due to many different ways of choosing subsets of conditioning word features, backoff procedures, and discounting methods. Usually, this space cannot be searched exhaustively, and optimizing models by a knowledge-inspired manual search procedure usually leads to suboptimal results, since only a small portion of the search space can be explored. In this paper we investigate the possibility of determining the structure of factored language models (i.e. the set of conditioning variables, the backoff procedure and the discounting parameters) by a data-driven search procedure, viz. Genetic Algorithms (GAs). We apply this technique to two different tasks (language modeling for Arabic and Turkish) and show that GAs lead to better models than either knowledge-inspired manual search or random search. The remainder of this paper is structured as follows: Section 2 describes the details of the factored language modeling approach. The application of GAs to the problem of determining language model structure is explained in Section 3. The corpora used in the present study are described in Section 4 and experiments and results are presented in Section 5. Section 6 compares the present study to related work and Section 7 concludes.

2 Factored Language Models

A standard statistical language model computes the probability of a word sequence $W = w_1, w_2, ..., w_T$ as a product of conditional probabilities of each word w_i given its history, which is typically approximated by just one or two preceding words (leading to bigrams, and trigrams, respectively). Thus, a trigram language model is described by

$$p(w_1, ..., w_T) \approx \prod_{i=3}^{T} p(w_i | w_{i-1}, w_{i-2}) \qquad (1)$$

Even with this limitation, the estimation of the required probabilities is challenging: many word contexts may be observed infrequently or not at all, leading to unreliable probability estimates under maximum likelihood estimation. Several techniques have been developed to address this problem, in particular smoothing techniques (Chen and Goodman, 1998) and class-based language models (Brown and others, 1992). In spite of such parameter reduction techniques, language modeling remains a difficult task, in particular for morphologically rich languages, e.g. Turkish, Russian, or Arabic. Such languages have a large number of word types in relation to the number of word tokens in a given text, as has been demonstrated in a number of previous studies (Geutner, 1995; Kiecza et al., 1999; Hakkani-Tür et al., 2002; K. Kirchhoff et al., 2003). This in turn results in a high perplexity and in a large number of out-of-vocabulary (OOV) words when applying a trained language model to a new unseen text.

2.1 Factored Word Representations

A recently developed approach that addresses this problem is that of *Factored Language Models (FLMs)* (Kirchhoff and others, 2002; Bilmes and Kirchhoff, 2003), whose basic idea is to decompose words into sets of features (or *factors*) instead of viewing them as unanalyzable wholes. Probabilistic language models can then be constructed over (sub)sets of word features instead of, or in addition to, the word variables themselves. For instance, words can be decomposed into stems/lexemes and POS tags indicating their morphological features, as shown below:

Word:	Stock	prices	are	rising
Stem:	Stock	price	be	rise
Tag:	Nsg	N3pl	V3pl	Vpart

Such a representation serves to express lexical and syntactic generalizations, which would otherwise remain obscured. It is comparable to class-based representations employed in class-based models; however, in FLMs several simultaneous class assignments are allowed instead of a single one. In general, we assume that a word is equivalent to a fixed number (K) of factors, i.e. $W \equiv f_{1:K}$. The task then is to produce a statistical model over the resulting representation - using a trigram approximation, the resulting probability model is as follows:

$$p(f_1^{1:K}, f_2^{1:K}, ..., f_T^{1:K}) \approx \prod_{t=3}^{T} p(f_t^{1:K} | f_{t-1}^{1:K}, ..., f_{t-2}^{1:K}) \quad (2)$$

Thus, each word is dependent not only on a single stream of temporally ordered word variables, but also on additional parallel (i.e. simultaneously occurring) features. This factored representation can be used in two different ways to improve over standard LMs: by using a product model or a backoff model. In a product model, Equation 2 can be simplified by finding conditional independence assumptions among subsets of conditioning factors and computing the desired probability as a product of individual models over those subsets. In this paper we only consider the second option, viz. using the factors in a backoff procedure when the word n-gram is not observed in the training data. For instance, a word trigram that is found in an unseen test set may not have any counts in the training set, but its corresponding factors (e.g. stems and morphological tags) may have been observed since they also occur in other words.

2.2 Generalized parallel backoff

Backoff is a common smoothing technique in language modeling. It is applied whenever the count for a given n-gram in the training data falls below a certain threshold τ. In that case, the maximum-likelihood estimate of the n-gram probability is replaced with a probability derived from the probability of the lower-order $(n-1)$-gram and a backoff weight. N-grams whose counts are above the threshold retain their maximum-likelihood estimates, discounted by a factor that re-distributes probability mass to the lower-order distribution:

$$p_{BO}(w_t | w_{t-1}, w_{t-2}) \quad (3)$$
$$= \begin{cases} d_c p_{ML}(w_t | w_{t-1}, w_{t-2}) \text{ if } c > \tau_3 \\ \alpha(w_{t-1}, w_{t-2}) p_{BO}(w_t | w_{t-1}) \text{ otherwise} \end{cases}$$

where c is the count of (w_t, w_{t-1}, w_{t-2}), p_{ML} denotes the maximum-likelihood estimate and d_c is a discounting factor that is applied to the higher-order distribution. The way in which the discounting factor is estimated determines the actual *smoothing* method (e.g. Good-Turing, Kneser-Ney, etc.) The normalization factor $\alpha(w_{t-1}, w_{t-2})$ ensures that the entire distribution sums to one. During standard backoff, the most distant conditioning variable (in this case w_{t-2}) is dropped first, then the second most distant variable etc. until the unigram is reached. This can be visualized as a backoff *path* (Figure 1(a)). If the only variables in the model are

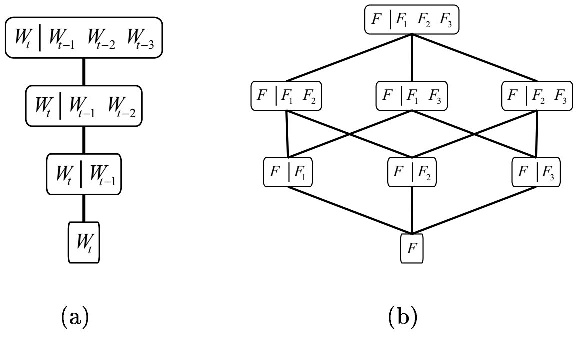


Figure 1: Standard backoff path for a 4-gram language model over words (left) and backoff graph for 4-gram over factors (right).

words, such a backoff procedure is reasonable. However, if variables occur in parallel, i.e. do not form a temporal sequence, it is not immediately obvious in which order they should be dropped. In this case, several backoff paths are possible, which can be summarized in a backoff *graph* (Figure 1(b)). In principle, there are several different ways of choosing among different paths in this graph:

1. Choose a fixed, predetermined backoff path based on linguistic knowledge, e.g. always drop syntactic before morphological variables.
2. Choose the path at run-time based on statistical criteria.
3. Choose multiple paths and combine their probability estimates.

The last option, referred to as *parallel* backoff, is implemented via a new, generalized backoff function (here shown for a 4-gram):

$$p_{GBO}(f|f_1,f_2,f_3) = \begin{cases} d_c p_{ML}(f|f_1,f_2,f_3) \text{ if } c > \tau_4 \\ \alpha(f_1,f_2,f_3) g(f,f_1,f_2,f_3) \text{ otherwise} \end{cases} \quad (4)$$

where c is the count of (f, f_1, f_2, f_3), $p_{ML}(f|f_1,f_2,f_3)$ is the maximum likelihood distribution, τ_4 is the count threshold, and $\alpha(f_1,f_2,f_3)$ is the normalization factor. The function $g(f,f_1,f_2,f_3)$ determines the backoff strategy. In a typical backoff procedure $g(f,f_1,f_2,f_3)$ equals $p_{BO}(f|f_1,f_2)$. In generalized parallel backoff, however, g can be *any non-negative function* of f, f_1, f_2, f_3. In our implementation of FLMs (K. Kirchhoff et al., 2003) we consider several different g functions, including the mean, weighted mean, product, and maximum of the smoothed probability distributions over all subsets of the conditioning factors. In addition to different choices for g, different discounting parameters can be chosen at different levels in the backoff graph. For instance, at the topmost node, Kneser-Ney discounting might be chosen whereas at a lower node Good-Turing might be applied. FLMs have been implemented as an add-on to the widely-used SRILM toolkit[1] and have been used successfully for the purpose of morpheme-based language modeling (Bilmes and Kirchhoff, 2003), multi-speaker language modeling (Ji and Bilmes, 2004), and speech recognition (K. Kirchhoff et al., 2003).

3 Learning FLM Structure

In order to use an FLM, three types of parameters need to specified: the initial conditioning factors, the backoff graph, and the smoothing options. The goal of structure learning is to find the parameter combinations that create FLMs that achieve a low perplexity on unseen test data. The resulting model space is extremely large: given a factored word representation with a total of k factors, there are $\sum_{n=1}^{k} \binom{k}{n}$ possible subsets of initial conditioning factors. For a set of m conditioning factors, there are up to $m!$ backoff paths, each with its own smoothing options. Unless m is very small, exhaustive search is infeasible. Moreover, nonlinear interactions between parameters make it difficult to guide the search into a particular direction, and parameter sets that work well for one corpus cannot necessarily be expected to perform well on another. We therefore need an automatic way of identifying the best model structure. In the following section, we describe the application of genetic-based search to this problem.

3.1 Genetic Algorithms

Genetic Algorithms (GAs) (Holland, 1975) are a class of evolution-inspired search/optimization techniques. They perform particularly well in problems with complex, poorly understood search spaces. The fundamental idea of GAs is to encode problem solutions as (usually binary) strings ('genes), and to evolve and test

[1]We would like to thank Jeff Bilmes for providing and supporting the software.

successive populations of solutions through the use of genetic operators applied to the encoded strings. Solutions are evaluated according to a fitness function which represents the desired optimization criterion. The individual steps are as follows:

```
Initialize: Randomly generate a set (population) of strings.
While fitness improves by a certain threshold:
  Evaluate fitness: calculate each string's fitness
  Apply operators: apply the genetic operators
    to create a new population.
```

The genetic operators include the probabilistic *selection* of strings for the next generation, *crossover* (exchanging subparts of different strings to create new strings), and *mutation* (randomly altering individual elements in strings). Although GAs provide no guarantee of finding the optimal solution, they often find good solutions quickly. By maintaining a population of solutions rather than a single solution, GA search is robust against premature convergence to local optima. Furthermore, solutions are optimized based on a *task-specific* fitness function, and the probabilistic nature of genetic operators helps direct the search towards promising regions of the search space.

3.2 Structure Search Using GA

In order to use GAs for searching over FLM structures (i.e. combinations of conditioning variables, backoff paths, and discounting options), we need to find an appropriate encoding of the problem.

Conditioning factors

The initial set of conditioning factors F are encoded as binary strings. For instance, a trigram for a word representation with three factors (A,B,C) has six conditioning variables: $\{A_{-1}, B_{-1}, C_{-1}, A_{-2}, B_{-2}, C_{-2}\}$ which can be represented as a 6-bit binary string, with a bit set to 1 indicating presence and 0 indicating absence of a factor in F. The string 10011 would correspond to $F = \{A_{-1}, B_{-2}, C_{-2}\}$.

Backoff graph

The encoding of the backoff graph is more difficult because of the large number of possible paths. A direct approach encoding every edge as a bit would result in overly long strings, rendering the search inefficient. Our solution is to encode a binary string in terms of graph grammar rules (similar to (Kitano, 1990)), which can be used to describe common regularities in backoff graphs. For instance, a node with m factors can only back off to children nodes with $m-1$ factors. For $m = 3$, the choices for proceeding to the next-lower level in the backoff graph can thus be described by the following grammar rules:

RULE 1: $\{x_1, x_2, x_3\} \rightarrow \{x_1, x_2\}$
RULE 2: $\{x_1, x_2, x_3\} \rightarrow \{x_1, x_3\}$
RULE 3: $\{x_1, x_2, x_3\} \rightarrow \{x_2, x_3\}$

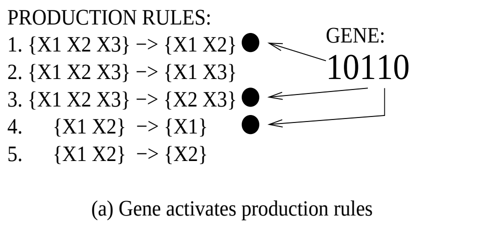
Here x_i corresponds to the factor at the ith position in the parent node. Rule 1 indicates a backoff that drops the third factor, Rule 2 drops the second factor, etc. The choice of rules used to generate the backoff graph is encoded in a binary string, with 1 indicating the use and 0 indicating the non-use of a rule, as shown schematically in Figure 2. The presence of two different rules at the same level in the backoff graph corresponds to parallel backoff; the absence of any rule (strings consisting only of 0 bits) implies that the corresponding backoff graph level is skipped and two conditioning variables are dropped simultaneously. This allows us to encode a graph using few bits but does not represent all possible graphs. We cannot selectively apply different rules to different nodes at the same level – this would essentially require a context-sensitive grammar, which would in turn increase the length of the encoded strings. This is a fundamental tradeoff between the most general representation and an encoding that is tractable. Our experimental results described below confirm, however, that sufficiently good results can be obtained in spite of the above limitation.

Smoothing options

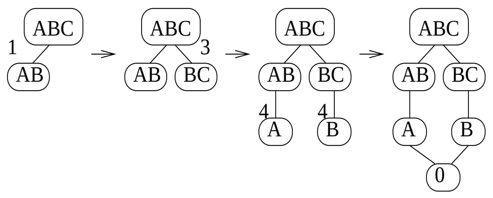
Smoothing options are encoded as tuples of integers. The first integer specifies the discounting method and the second integer specifies the backoff threshold. The integer string consists of successive concatenated tuples, each representing the smoothing option at a node in the graph. The GA operators are applied to concatenations of all three substrings describing the set of factors, backoff graph, and smoothing options, such that all parameters are optimized jointly.

4 Data

We tested our language modeling algorithms on two different data sets from two different lan-



(a) Gene activates production rules



(b) Generation of Backoff Graph by rules 1, 3, and 4

Figure 2: Generation of Backoff Graph from production rules selected by the gene 10110.

guages, Arabic and Turkish.

The Arabic data set was drawn from the CallHome Egyptian Conversational Arabic (ECA) corpus (LDC, 1996). The training, development, and evaluation sets contain approximately 170K, 32K, and 18K words, respectively. The corpus was collected for the purpose of speech recognizer development for conversational Arabic, which is mostly dialectal and does not have a written standard. No additional text material beyond transcriptions is available in this case; it is therefore important to use language models that perform well in sparse data conditions. The factored representation was constructed using linguistic information from the corpus lexicon, in combination with automatic morphological analysis tools. It includes, in addition to the word, the stem, a morphological tag, the root, and the pattern. The latter two are components which when combined form the stem. An example of this factored word representation is shown below:

Word:il+dOr/Morph:noun+masc-sg+article/
Stem:dOr/Root:dwr/Pattern:CCC

For our Turkish experiments we used a morphologically annotated corpus of Turkish (Hakkani-Tür et al., 2000). The annotation was performed by applying a morphological analyzer, followed by automatic morphological disambiguation as described in (Hakkani-Tür et al., 2002). The morphological tags consist of the initial root, followed by a sequence of inflectional groups delimited by derivation boundaries (^DB). A sample annotation (for the word *yararlanmak*, consisting of the root *yarar* plus three inflectional groups) is shown below:

yararmanlak:
yarar+Noun+A3sg+Pnon+Nom
^DB+Verb+Acquire+Pos
^DB+Noun+Inf+A3sg+Pnon+Nom

We removed segmentation marks (for titles and paragraph boundaries) from the corpus but included punctuation. Words may have different numbers of inflectional groups, but the FLM representation requires the same number of factors for each word; we therefore had to map the original morphological tags to a fixed-length factored representation. This was done using linguistic knowledge: according to (Oflazer, 1999), the final inflectional group in each dependent word has a special status since it determines inflectional markings on head words following the dependent word. The final inflectional group was therefore analyzed into separate factors indicating the number (N), case (C), part-of-speech (P) and all other information (O). Additional factors for the word are the root (R) and all remaining information in the original tag not subsumed by the other factors (G). The word itself is used as another factor (W). Thus, the above example would be factorized as follows:

W:yararlanmak/R:yarar/P:NounInf-N:A3sg/
C:Nom/O:Pnon/G:NounA3sgPnonNom+Verb
+Acquire+Pos

Other factorizations are certainly possible; however, our primary goal is not to find the best possible encoding for our data but to demonstrate the effectiveness of the FLM approach, which is largely independent of the choice of factors. For our experiments we used subsets of 400K words for training, 102K words for development and 90K words for evaluation.

5 Experiments and Results

In our application of GAs to language model structure search, the perplexity of models with respect to the development data was used as an optimization criterion. The perplexity of the best models found by the GA were compared to the best models identified by a lengthy manual search procedure using linguistic knowledge about dependencies between the word factors involved, and to a random search procedure which evaluated the same number of strings as the GA. The following GA options gave good results: population size 30-50, crossover probability 0.9, mutation population 0.01, Stochastic Universal Sampling as the selection operator, 2-point crossover. We also experimented with re-initializing the GA search with the best model found in previous runs. This method consistently improved the performance of normal GA search and we used it as the basis for the results reported below. Due to the large number of factors in the Turkish word representation, models were only optimized for conditioning variables and backoff paths, but not for smoothing options. Table 1 compares the best perplexity results for standard word-based models and for FLMs obtained using manual search (Hand), random search (Rand), and GA search (GA). The last column shows the relative change in perplexity for the GA compared to the better of the manual or random search models. For tests on both the development set and evaluation set, GA search gave the lowest perplexity. In the case of Arabic, the GA search was performed over conditioning factors, the backoff graph, and smoothing options. The results

N	Word	Hand	Rand	GA	Δ (%)
	Dev Set				
2	593.8	555.0	556.4	539.2	-2.9
3	534.9	533.5	497.1	444.5	-10.6
4	534.8	549.7	566.5	522.2	-5.0
	Eval Set				
2	609.8	558.7	587.8	548.9	-1.8
3	545.4	583.5	556.5	477.1	-14.3
4	543.9	571.7	574.6	550.7	-3.7

Table 1: Perplexity for Turkish language models. N = n-gram order, Word = word-based models, Hand = manual search, Rand = random search, GA = genetic search.

N	Word	Hand	Rand	GA	Δ (%)
	Dev Set				
2	229.9	229.6	229.9	223.2	-2.8
3	229.3	226.1	230.3	212.6	-6.0
	Eval Set				
2	249.9	230.1	239.2	226.9	-1.4
3	285.4	217.1	224.3	206.2	-5.0

Table 2: Perplexity for Arabic language models (w/o unknown words).

N	Word	Hand	Rand	GA	Δ (%)
	Dev Set				
2	236.0	195.5	198.5	192.2	-1.7
3	237.0	199.0	202.0	188.1	-5.5
	Eval Set				
2	235.2	234.1	247.7	235.8	0.7
3	253.9	229.2	219.0	212.2	-3.1

Table 3: Perplexity for Arabic language models (with unknown words).

in Table 2 were obtained by training and testing without consideration of out-of-vocabulary (OOV) words. Our ultimate goal is to use these language models in a speech recognizer with a fixed vocabulary, which cannot recognize OOV words but requires a low perplexity for other word combinations. In a second experiment, we trained the same FLMs from Table 2 with OOV words included as the unknown word token. Table 3 shows the results. Again, we see that the GA outperforms other search methods in all but one case. The best language models all used parallel backoff and different smoothing options at different backoff graph nodes. The Arabic models made use of all conditioning variables (Word, Stem, Root, Pattern, and Morph) whereas the Turkish models used only the W, P, C, and R variables (see above Section 4).

6 Related Work

Various previous studies have investigated the feasibility of using units other than words for language modeling (e.g. (Geutner, 1995; Çarki et al., 2000; Kiecza et al., 1999)). However, in all of these studies words were decomposed into linear sequences of morphs or morph-like units, using either linguistic knowledge or data-driven techniques. Standard language models were then trained on the decomposed representations. The resulting models essentially express statistical relationships between morphs, such as stems and affixes. For this reason, a context larger than that provided by a trigram is typically required, which quickly leads to data-sparsity. In contrast to these approaches, factored language models encode morphological knowledge not by altering the linear segmentation of words but by encoding words as parallel bundles of features.

The general possibility of using multiple conditioning variables (including variables other than words) has also been investigated by

(Dupont and Rosenfeld, 1997; Gildea, 2001; Wang, 2003; Zitouni et al., 2003). Mostly, the additional variables were general word classes derived by data-driven clustering procedures, which were then arranged in a backoff lattice or graph similar to the present procedure. All of these studies assume a fixed path through the graph, which is usually obtained by an ordering from more specific probability distributions to more general distributions. Some schemes also allow two or more paths to be combined by weighted interpolation. FLMs, by contrast, allow different paths to be chosen at run-time, they support a wider range of combination methods for probability estimates from different paths, and they offer a choice of different discounting options at every node in the backoff graph. Most importantly, however, the present study is to our knowledge the first to describe an entirely data-driven procedure for identifying the best combination of parameter choices. The success of this method will facilitate the rapid development of FLMs for different tasks in the future.

7 Conclusions

We have presented a data-driven approach to the selection of parameters determining the structure and performance of factored language models, a class of models which generalizes standard language models by including additional conditioning variables in a principled way. In addition to reductions in perplexity obtained by FLMs vs. standard language models, the data-driven model section method further improved perplexity and outperformed both knowledge-based manual search and random search.

Acknowledgments

We would like to thank Sonia Parandekar for the initial version of the GA code. This material is based upon work supported by the NSF and the CIA under NSF Grant No. IIS-0326276. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of these agencies.

References

Jeff A. Bilmes and Katrin Kirchhoff. 2003. Factored language models and generalized parallel backoff. In *Proceedings of HLT/NACCL*, pages 4–6.

P.F. Brown et al. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

K. Çarki, P. Geutner, and T. Schultz. 2000. Turkish LVCSR: towards better speech recognition for agglutinative languages. In *Proceedings of ICASSP*.

S. F. Chen and J. Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report Tr-10-98, Center for Research in Computing Technology, Harvard University.

P. Dupont and R. Rosenfeld. 1997. Lattice based language models. Technical Report CMU-CS-97-173, Department of Computer Science, CMU.

P. Geutner. 1995. Using morphology towards better large-vocabulary speech recognition systems. In *Proceedings of ICASSP*, pages 445–448.

D. Gildea. 2001. *Statistical Language Understanding Using Frame Semantics*. Ph.D. thesis, University of California, Berkeley.

D. Hakkani-Tür, K. Oflazer, and Gökhan Tür. 2000. Statistical morphological disambiguation for agglutinative languages. In *Proceedings of COLING*.

D. Hakkani-Tür, K. Oflazer, and Gökhan Tür. 2002. Statistical morphological disambiguation for agglutinative languages. *Journal of Computers and Humanities*, 36(4).

J.H. Holland. 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

Gand Ji and Jeff Bilmes. 2004. Multi-speaker language modeling. In *Proceedings of HLT/NAACL*, pages 137–140.

D. Kiecza, T. Schultz, and A. Waibel. 1999. Data-driven determination of appropriate dictionary units for Korean LVCSR. In *Proceedings of ICASSP*, pages 323–327.

K. Kirchhoff et al. 2002. Novel speech recognition models for Arabic. Technical report, Johns Hopkins University.

Hiroaki Kitano. 1990. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, pages 461–476.

LDC. 1996. http://www.ldc.upenn.edu/Catalog/-LDC99L22.html.

K. Kirchhoff et al. 2003. Novel approaches to Arabic speech recognition: Report from 2002 Johns-Hopkins summer workshop. In *Proceedings of ICASSP*, pages I–344–I–347.

K. Oflazer. 1999. Dependency parsing with an extended finite state approach. In *Proceedings of the 37th ACL*.

W. Wang. 2003. Factorization of language models through backing off lattices. Computation and Language E-print Archive, oai:arXiv.org/cs/0305041.

I. Zitouni, O. Siohan, and C.-H. Lee. 2003. Hierarchical class n-gram language models: towards better estimation of unseen events in speech recognition. In *Proceedings of Eurospeech - Interspeech*, pages 237–240.