Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Speech and Language 19 (2005) 321–343
COMPUTER SPEECH AND LANGUAGE
www.elsevier.com/locate/csl

Using semantic analysis to improve speech recognition performance

Hakan Erdogan [a,*,1], Ruhi Sarikaya [b], Stanley F. Chen [b], Yuqing Gao [b], Michael Picheny [b]

[a] *Faculty of Engineering and Natural Sciences, Sabanci University, Orhanli Tuzla, 34956 Istanbul, Turkey*
[b] *IBM TJ Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA*

Received 13 January 2003; accepted 14 October 2004
Available online 23 November 2004

Abstract

Although syntactic structure has been used in recent work in language modeling, there has not been much effort in using semantic analysis for language models. In this study, we propose three new language modeling techniques that use semantic analysis for spoken dialog systems. We call these methods *concept sequence modeling*, *two-level semantic-lexical modeling*, and *joint semantic-lexical modeling*. These models combine lexical information with varying amounts of semantic information, using annotation supplied by either a shallow semantic parser or full hierarchical parser. These models also differ in how the lexical and semantic information is combined, ranging from simple interpolation to tight integration using maximum entropy modeling. We obtain improvements in recognition accuracy over word and class N-gram language models in three different task domains. Interpolation of the proposed models with class N-gram language models provides additional improvement in the air travel reservation domain. We show that as we increase the semantic information utilized and as we increase the tightness of integration between lexical and semantic items, we obtain improved performance when interpolating with class language models, indicating that the two types of models become more complementary in nature.

* Corresponding author. Tel.: +90 2164839607; fax: +90 2164839550.
E-mail address: haerdogan@sabanciuniv.edu (H. Erdogan).
[1] The author is currently with Sabanci University, Istanbul, Turkey. The work was carried out when the author was at IBM TJ Watson Research Center.

0885-2308/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.csl.2004.10.002

1. Introduction

Language modeling for speech recognition attempts to model the probability $P(W)$ of observing a word sequence W in natural speech. Until recently, simple N-gram language modeling has been the dominant method of modeling natural language for speech recognition.

The purpose of language modeling is to bias a speech recognizer towards sentences appropriate to a particular task or domain and away from meaningless sequences of words. N-gram language models use a moving window of N words to compute the probability of a sentence. It has long been argued that N-gram language models are suboptimal, and one could do better by considering longer range dependencies between words. However, it has been surprisingly difficult to beat the performance of N-gram language models consistently. Recently, there have been studies investigating the use of syntactic structure in language modeling (Jelinek and Chelba, 1999; Chelba and Jelinek, 1999; Khudanpur and Wu, 2000). This method is named structured language modeling (SLM). In SLM, one uses a bottom-up syntactic parser to find the most likely syntactic parses of a partial sentence. These partial parses are used to enhance the language model for speech recognition. One advantage of using parse information is that one can condition word prediction on a longer history. For example, the phrasal head words that are exposed by the parser before the current word are used as additional members of the history. The associated non-terminal labels can also be used in the history. Charniak (2001) and Roark (2001) also explore the use of syntactic structure in language modeling, but using a top-down syntactic parser. With SLM's, researchers have achieved consistent perplexity reductions and about 2–3% relative improvement in word error rate over N-gram models across varied tasks such as Switchboard and Wall Street Journal (Wu and Khudanpur, 2000).

Syntactic parsers analyze sentences in terms of a grammar and parts of speech. This analysis does not attempt to identify constituents that represent similar or related meanings. Another analysis method, *semantic analysis*, attempts to analyze sentences based on constituents that represent concepts or meaning. Semantic analysis is currently used in limited-domain, task-specific spoken dialog systems. Generic semantic analyzers for broad domains are hard to develop due to the many (or possibly unlimited) number of concepts one can refer to in an unconstrained conversation. However, semantic analyzers have been successfully developed for spoken dialog systems for domains such as air travel reservation, e.g., ATIS and DARPA Communicator, and mutual fund transactions. Examples of semantic analyzers include the SOUP parser (Gavalda, 2000; Langley et al., 2002) and the statistical semantic classer and parser developed at IBM (Davies et al., 1999; Ward, 2000).

In this work, we employ the information provided by a semantic analyzer to enhance the language model used in a spoken dialogue system (Erdogan et al., 2002). By using semantic information, we can bias the recognizer towards sentences that are more meaningful within our domain. Our statistical model estimates the joint probability of a sentence and its most likely semantic parse. Sentences that have parses with high probability will then be preferred in recognition.

In this paper, we introduce three new ways to use semantic information in language modeling. The three approaches differ in the amount of semantic information used and the tightness of integration. We hypothesize that as we increase the amount of semantic information and the tightness of integration, language models are further improved. The first method requires a shallow semantic parser to extract a concept sequence from a sentence. The concept sequence can be extracted

using a special weighted finite-state transducer (WFST) as we describe in Section 2.1 or using a statistical classer trained using annotated examples. In our first approach, the concept sequence is used to obtain a concept LM score and is interpolated with the word LM score to obtain the total LM score. The second and third methods can use either a shallow or full semantic parse. The second approach combines semantic parse information with the information in a word sequence using a two-level statistical model. The third approach uses the semantic information in parses by constructing a single probability model that models the probability of a word sequence and its semantic parse. To compute the joint probability of a sentence and its parse, we use features such as parent label name and last completed constituent label name as well as regular N-gram history features for predicting both words and parser labels. We use maximum entropy (ME) models to combine these features into an effective language model for both the two-level and joint methods.

In this research, we do not discuss the real-time building of the parses or the incorporation of the parser into a speech recognition decoder. We assume that we need only compute LM scores for complete sentences; this makes things easier for us, as it is straightforward to condition on elements such as the current parent node label for each word. In Chelba and Jelinek (1999, 2000), this information could not be used since the parse is built bottom-up and the parent label is not available to the decoder when the current word is being decoded. Charniak (2001) and Roark (2001) avoid this problem by using a top-down parser that has the parent label information available.

There has been very limited work in using semantic information in language modeling. One related research area is topic-dependent language modeling. If the topic of the utterance to be decoded can be predetermined to some measure, we can use this information to adapt or choose a specific language model. For example, a separate LM for each topic can be built and then interpolated with a generic LM (Iyer and Ostendorf, 1996). In spoken dialog systems, the current state of the dialog as provided by a dialog manager (Visweswariah and Printz, 2001) can be used to adapt an LM; this is similar in spirit to topic-dependent language modeling. In our work, we condition on local semantic information present within sentences, as opposed to coarse-grained topic information.

For spoken dialog systems, it is possible to design phrasal context free grammars (CFGs) that accept word sequences representing concepts (such as date, time, place, etc.) relevant to an application domain (Hacioglu and Ward, 2001b; Langley et al., 2002). One can combine these concept CFGs into a sentencial stochastic context free grammar (SCFG) that is designed to accept whole sentences. This sentencial SCFG is equivalent to a stochastic recursive transition network (SRTN), which in turn can be used as a search graph for a speech recognition decoder (Hacioglu and Ward, 2001b) which may output semantic concepts along with the words. A similar approach has also been taken in Monkowski (2001). In this type of approach, the decoder automatically parses the sentence under analysis during the recognition process. The language model score is determined using a concept N-gram model along with a word sequence model based on SCFG probabilities. This type of approach has been shown to improve speech recognition performance over word N-gram models (Hacioglu and Ward, 2001a,b). This joint decoding and analysis paradigm enables the use of new LM techniques that employ semantic analysis. Our three new methods are all different from the approach used in Hacioglu and Ward (2001a) in that our models are not SCFGs (though CFGs can be used in a preliminary step to produce semantic parses). Instead,

our first method uses interpolation to loosely combine lexical and semantic information, while our last two methods employ maximum entropy modeling to tightly integrate lexical and semantic features to form a unified semantic language model.

The paper is organized as follows. In Section 2, we present two different semantic analysis methods that we use in this paper, namely a WFST-based method and the use of a statistical analyzer. In Section 3, after a brief introduction to N-gram language modeling, we introduce our first method, N-gram concept sequence modeling. We describe our second approach in Section 4. This approach is a two-level semantic-lexical model that models the parser label sequence probability first and, based on the parser label sequence, estimates the word sequence probabilities. The third approach, also described in Section 4, is a joint semantic-lexical probability computation of the word sequence and parser probabilities. The experimental results are presented in Section 5.

2. Semantic analysis

Semantic analysis refers to analyzing a natural language sentence using semantic tags and groupings. A hierarchical grouping structure can be constructed from word tags to the sentence level, similar to syntactic analysis. By *shallow* semantic parsing, we mean a single (non-trivial) level of hierarchy being present in the semantic parse (in addition to the words themselves). Shallow semantic parsers are sometimes called *chunkers* or *classers*. A full semantic parse refers to multiple levels of hierarchy. The tags and labels in semantic analysis refer to the meanings or concepts that the constituent words represent. Figs. 2 and 3 are examples of a shallow and full semantic parse, respectively. We assume that the assignment of a semantic tag to each word is not obligatory, or alternatively, one can assume that the semantic tag for some words is equal to the word itself. This is the case for our WFST-based shallow semantic parser.

In this paper, we represent semantic parses using a bracket notation. In this notation, each word is immediately followed by its tag separated with a "_" and the constituents are represented by opening and closing tokens, `[LABEL` and `LABEL]`, respectively. Whenever tags are not used, we omit the tags in the representation and just use open and close label tokens only. The token sequence for the semantic parse in Fig. 2 is shown below:

```
[!S! i_word want_word to_word book_word a_word [RT-OW round_rt-ow
trip_rt-ow RT-OW] ticket_word to_word [LOC Denver_city Colorado_state
LOC] for_word [DATE september_month fifteenth_date DATE] !S!]
```

2.1. Shallow semantic analysis using weighted finite state transducers

It is much easier to write phrasal concept grammars than grammars which accept whole sentences. For example, one can come up with a grammar for all expressions that refer to a date. Given a list of locations that we are interested in, we can write a CFG that accepts all of these location expressions. Similarly, many other concepts like these can be represented by phrasal CFGs.

When the phrasal CFGs represent regular languages (which is usually the case in practice), they can be converted into finite-state acceptors. Furthermore, parsing with these CFGs can be implemented by creating finite-state transducers (FSTs) that accept a word sequence $w_1 w_2 \ldots w_N$ as input and output a bracketed expression `[LABEL` $w_1 w_2 \ldots w_N$ `LABEL]`, where `LABEL` is the head

label or concept for the phrasal CFG. These concept FST's can be combined with a trivial FST that maps each word to itself to perform the shallow parsing of whole sentences, by placing each of these FSTs in parallel, assigning a cost to each FST, and adding a loop-back arc, as illustrated in Fig. 1. The transducer formed in this manner is a weighted finite-state transducer (WFST) as we have introduced costs. We refer to a shallow semantic parser of this type as a *WFST classer*.

It is obvious that a sentence may have multiple parses, as there can be overlap between concept CFGs; e.g., "two twenty" can represent a time, apartment number, or flight number. Thus, we assign probabilities, or costs, to each phrasal FST so that we may prefer some parses over others, and train these probabilities to maximize the likelihood of training data. We initialize the costs of phrasal FSTs uniformly and assign the trivial FST a slightly higher cost, so parses containing fewer phrases will be preferred. We then train FST costs using the Viterbi version of Expectation Maximization training. To further improve performance, we can select the Viterbi parse for an iteration by rescoring all possible parses using N-gram statistics for concept sequences that are obtained from the top parses from the previous iteration. Hopefully, we can distinguish between time expressions, apartment numbers and flight numbers using N-gram context information.

The sentences that are accepted by the WFST shown in Fig. 1 can be trivially represented by a (regular) SCFG. An SCFG is equivalent to a stochastic recursive transition network (SRTN), which is similar to a weighted finite-state acceptor but treats each non-terminal arc as a subroutine which can be expanded and which can contain recursive arcs within itself (Jurafsky and Martin, 2000, p. 391).

This kind of shallow parsing has been used in spoken dialog systems (Hacioglu and Ward, 2001a) and speech-to-speech translation (Langley et al., 2002). The SRTN can be used as the search graph on which decoding is performed (Hacioglu and Ward, 2001b). The decoder may

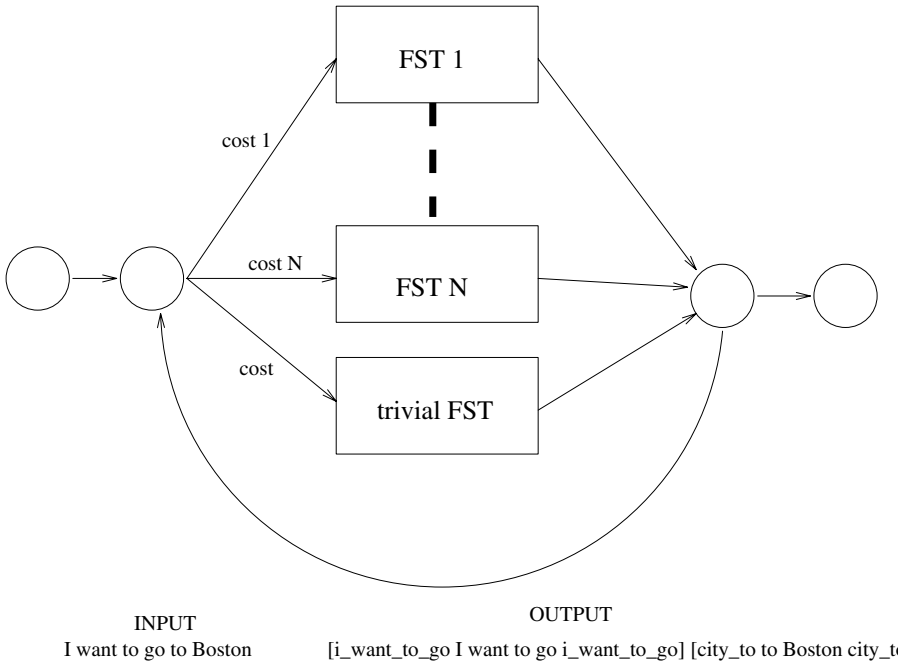

Fig. 1. An illustration of the WFST that implements a semantic classer for concept extraction.

output the concepts (i.e., CFG labels) as an end result of the decoding process. In Langley et al. (2002), the SOUP parser (Gavalda, 2000) is used for parsing arguments into the interlingua that is used in speech-to-speech translation. The IBM ViaVoice decoder uses a similar technology (termed *embedded grammars*) that also enables the use of grammars that can change on the fly from utterance to utterance (Monkowski, 2001). To implement our parser, we have used an in-house finite-state machine (FSM) toolkit (Chen, 2000).

2.2. Statistical semantic analysis

Full statistical semantic parsing is used at IBM for natural language understanding for spoken dialog systems (Ward, 2000; Davies et al., 1999). Statistical parsing requires an annotated training corpus of in-domain sentences which is used to train parser probabilities. Our statistical parser is based on predicting parser actions (tagging a word, extending a parse by connecting constituents to parent labels, etc.) using a decision tree to estimate the probability of parser actions based on the current context (Magerman, 1994). The search is performed left-to-right and bottom-up.

The annotation of training data is the most time-intensive operation required for statistical semantic parsing. However, this can be done in an efficient manner using low-cost labor (Ward, 2000). We perform semantic analysis in two steps. The first step is done by a shallow semantic parser which is also termed the *classer*. Essentially, this groups together words representing the same concept. The function of the classer is similar to that of the WFST-based method described in Section 2.1, but no explicit definition of phrasal grammars is required. These grammars are implicitly inferred from the annotated training data. In addition, the classer tags each word with an appropriate semantic tag. A tree representation of the output of the semantic classer for a sentence in the air travel domain is shown in Fig. 2.

The full semantic parser takes the output of the classer and builds a full hierarchical parse of semantic constituents. An example is shown in Fig. 3. The decision-tree based statistical classer and parser were developed at IBM (Ward, 2000), and used in the NLU module of our DARPA Communicator system for parsing air travel domain utterances. The analysis process is separated into two parts because training becomes easier and it simplifies the overall model. However, it is also possible to define a full semantic parser that will perform the whole analysis in one step.

Our air travel domain statistical classer achieves 95% exact match accuracy and the statistical parser achieves 75% exact match accuracy on a test set of 1173 sentences. We have not done any accuracy tests on the WFST classer; however, the accuracy is close to that of the statistical classer.

We used both the statistical semantic classer output and the full parser output in our language modeling experiments; how we do this is described in the following section.

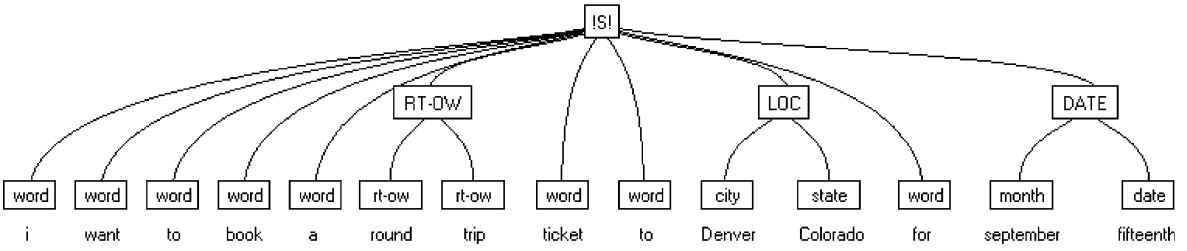

Fig. 2. An example of semantic classer output.

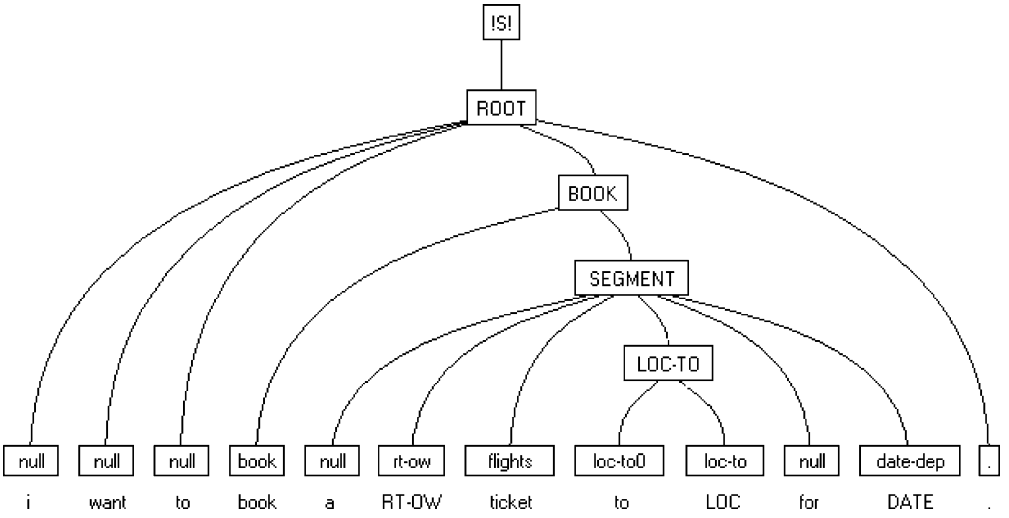

Fig. 3. The full parser output of the same sentence that is classed in Fig. 2.

3. N-gram based language modeling

3.1. Word and class N-gram language models

In N-gram language modeling, we observe that the probability of a word sequence $W = w_1 \ldots w_n$ can be decomposed as follows:

$$P(W) = \prod_{i=1}^{n} P(w_i|w_1, \ldots, w_{i-1}).$$

Note that this way of writing the probability is amenable to real-time speech recognition, since the probability of a word is conditioned only on past words. To reduce data sparseness, one can introduce a Markov assumption and limit the history size for each word to $N-1$. We define equivalence classes on histories where each history is mapped to an equivalence class corresponding to its last $N-1$ words. Thus, we make the following approximation:

$$P(w_i|w_1, \ldots, w_{i-1}) \approx P(w_i|w_{i-N+1}, \ldots, w_{i-1}).$$

Probabilities can be estimated from training data using maximum likelihood estimation. Training is relatively fast and only requires accumulating counts for each N-gram that occurs in the training data. Experimentally, trigrams ($N = 3$) have given the best trade-off in terms of performance and computational requirements. One can sometimes get minor improvements by using 4-grams but aggressive smoothing should be employed.

As with many maximum likelihood techniques, researchers have found the need for smoothing to avoid over-training and to counteract training data sparseness. Thus, there have been many effective methods introduced for smoothing N-gram language models (Chen and Goodman, 1998).

In this paper, we use held-out interpolation to smooth N-gram language models. In held-out interpolation, one interpolates trigram, bigram and unigram probabilities (for $N = 3$). The

interpolation parameters are determined by clustering N-gram histories according to history counts and minimizing the perplexity of heldout data by choosing the appropriate interpolation weights for each cluster of histories (Jelinek and Mercer, 1980).

An extension of word N-gram LM's is class-based N-grams (Brown et al., 1992). Words are partitioned into classes either automatically or manually and probabilities are shared among words in a class. Let $c(w_i)$ be the class for w_i. LM scores are computed as

$$P(W) = \sum_{i=1}^{N} P(w_i|c(w_i))P(c(w_i)|c(w_{i-N+1}), \ldots, c(w_{i-1})).$$

The class-based language model built at IBM for the DARPA Communicator (DC) task (Erdogan, 2001; Gao et al., 2001) is very rich in terms of word classes. It has a total of 41 manually contructed classes, including commonly used classes such as cities, months, and days of the week. For some classes, the distribution $P(w_i|c(w_i))$ is computed using prior information and word unigram statistics; for others, it is simply taken to be uniform. While our class LM has been extensively optimized for this task, in this work we would like to focus on modeling long-span information that complements the information in the class LM.

In our earlier work, we have shown that class-based N-gram modeling improves performance considerably for DARPA Communicator (Erdogan, 2001; Gao et al., 2001). Thus, in all of our semantic language modeling experiments, our semantic language models estimate class probabilities rather than word probabilities. To compute the final word probabilities, we multiple class probabilities by the terms $P(w_i|c(w_i))$. In the following discussion, we do not always explicitly re-iterate this point; however, the reader should be aware that whenever we use a word entity w_i in an equation, it is straightforward to replace it with the word class $c(w_i)$ and to multiply the resulting probability with $P(w_i|c(w_i))$ to obtain a word probability.

3.2. Concept sequence modeling

We define the concept sequence for a sentence to be the sequence of semantic chunks obtained through a semantic statistical classer or by using a WFST-based semantic classer. Words that are not covered by other concepts are considered to be trivial concepts and are also included in the concept sequence, e.g., the words with tag `word` in Fig. 2. A concept sequence for a sentence provides clues about the semantic clarity of the sentence. Thus, the probability of the concept sequence of a sentence can be used to improve language modeling. In addition, a concept N-gram may carry more long-range information than a word N-gram of the same length, since a single concept may be multiple words and may be determined using information outside of the N-gram. Consequently, one hopes that combining the probabilities of a word sequence with its concept sequence should help increase speech recognition performance.

There have been studies that use concepts for language modeling (Hacioglu and Ward, 2001b). In Hacioglu and Ward (2001b), each concept is written as an SCFG and compiled into a stochastic recursive transition network. Sentence probabilities are found by multiplying the concept sequence probability $P(C)$ and the probability that each concept grammar generates its corresponding word sequence $P(W|C)$.

In our approach, we use the following approximations:

$$P(W) = \sum_C P(W|C)P(C), \tag{1}$$

$$P(W) \approx \max_C P(W|C)P(C) = P(W|C_W)P(C_W), \tag{2}$$

$$P(W) \approx P(W)P(C_W)^{\lambda}. \tag{3}$$

Here, C_W is the best concept sequence as returned by our classer and λ is a scalar. In this approximation, we assume that the word sequence probability is independent of the concept sequence, which is not true in practice. However, this enables us to train word and concept sequence probabilities separately and avoid training SCFG weights. Also, we combine the word LM score and the concept LM score using log-linear interpolation. A similar approach is also taken when combining acoustic and language models in speech recognition. The reason for the scalar λ is to enable the combination of probabilities which have differing ranges.

The concept LM score can be computed using trigram probabilities, $P_{\text{N-gram}}(C_W) = \prod_i P(C_i|C_{i-2},C_{i-1})$, where C_i is the ith concept obtained from the semantic parse of the sentence. To obtain the overall language model score in log space, we interpolate the concept N-gram log probability and the word N-gram log probability

$$\text{LM score} = \log P_{\text{N-gram}}(W) + \lambda \log P_{\text{N-gram}}(C_W).$$

This score is not a good estimate of the (log) frequency of the sentence, $\log P(W)$, but just a score to compare the relative merit of sentences. [2]

In Hacioglu and Ward (2001b), the probability $P(W|C)$ is found using a SCFG and is independent of the context around each phrase. Hence, for example, the probability of the initial word in a phrase is conditioned only on the parent label for the word and ignores preceding lexical context. However, in our language models, we would like to use preceding lexical context in computing the probability of a word sequence; this is achieved by using a word N-gram model. This may result in a better language model since cross-concept lexical dependencies can be captured. Furthermore, while N-gram model smoothing has been extensively studied, estimating SCFG weights in the presence of sparse data is less well understood. Since concept and lexical sequences are separated in the model, we can also use different training data for each part as an added advantage. In our joint models, we combine both lexical and semantic contexts into a joint model which also avoids using SCFG weights.

In our experiments, the concept vocabulary we used was different for the statistical and WFST-based semantic analyzers. The statistical classer was designed for the DARPA Communicator task and only marks 20 relevant dialog concepts. While this yielded adequate performance, the

[2] This score can be seen as a generalized approximation of the joint probability $\log P(W,C_W)$ assuming independence of concept and word sequences.

coverage of sentences with non-trivial concepts was not great. We designed the WFST classer to have more coverage and determined 23 phrasal CFG's that have larger coverage than our classer concepts. Each CFG contains a variety of ways of expressing a concept while constraining grammar phrases to be semantically coherent. Words that are not covered by concepts are assumed to be trivial concepts.

Here are some example air travel sentences as parsed with the WFST classer:

- [**book_flight** please book me on **book_flight**] [**numflt** flight twenty one **numflt**]
- [**i_want_to_go** i would like to fly **i_want_to_go**] [**city_from** from philadelphia **city_from**] [**city_to** to dallas **city_to**]
- [**request1** could you please list the **request1**] flights [**city_from** from boston **city_from**] [**city_to** to denver **city_to**] on [**date** july twenty eighth **date**]

In our experiments, the class-based trigram language model was trained using 137 K sentences and smoothed using held-out interpolation on a held-out set of 18 K sentences. The concept trigram model $P(C)$ was trained on a 100 K sentence subset of the training data and also smoothed using held-out interpolation with the same held-out set. We evaluate our concept-based language models in Section 5.

4. Maximum entropy based semantic language models

4.1. Maximum entropy estimation

Maximum entropy is an effective method for combining multiple information sources in statistical modeling. Maximum entropy modeling produces a probability model that matches empirical feature expectations exactly, but which is otherwise as "uniform" as possible. This can be interpreted as making as few assumptions as possible in the model.

For this work, we use maximum entropy modeling to incorporate lexical and semantic information sources for language modeling. ME models have been used in language modeling before, in the context of N-gram models (Chen and Rosenfeld, 2000), whole sentence models (Rosenfeld et al., 2001) and syntactic structural language models (Wu and Khudanpur, 1999). Khudanpur and Wu (2000) also added topic information as another feature to include in a maximum entropy model. However, there has not been any past work that uses higher-level sentence-based semantic features, such as information obtained through a semantic parser. Such semantic features could be very important, especially in domain-specific conversational dialog systems or domain-dependent speech-to-speech translation applications.

We propose to use high-level semantic features in domain-specific language models. To produce this semantic information, we use the semantic parsers introduced in Section 2.

We use maximum entropy models to rescore N-best lists, so the parser need not be bottom-up like the one used in Chelba and Jelinek (1999). For many telephony applications, it is feasible to perform an N-best rescoring step after an N-gram-based decoding is done, since the time needed to rescore N-best lists is relatively small.

We can combine multiple features in a maximum entropy model in the following way:

$$P(o|h) = \frac{e^{\sum_i \lambda_i f_i(o,h)}}{\sum_{o'} e^{\sum_i \lambda_i f_i(o',h)}},$$

where o is the outcome (e.g., the current word), h represents the history or context and f_i are indicator functions, or *features*, which are "activated" when certain outcomes occur in certain contexts. In particular, we restrict $f_i(o,h)$ to have the form

$$f_i(o,h) = \begin{cases} 1 & \text{if } o = o_i \text{ and } q_i(h) = 1, \\ 0 & \text{otherwise,} \end{cases}$$

where o_i is the outcome associated with feature i and $q_i(h)$ is an indicator function on histories; i.e., we constrain features to be active for exactly one outcome.

For example, a bigram feature f_i representing the presence of the word sequence `IN THE` would have o_i = `THE` and $q_i(h)$ would be 1 only if the word `IN` is the immediate predecessor of the current word in context h.

4.1.1. Feature selection

Maximum entropy models suffer from overtraining just as other maximum likelihood models do. To avoid overfitting training data, one would like to avoid including irrelevant features in a model. One method to implement feature selection involves choosing features incrementally in batches according to the criteria that they result in the largest increase in training data likelihood (Berger et al., 1996). This approach is quite computationally expensive.

Another method is simply to throw away features that occurred infrequently in the training data. Typically, features with a count of less than a threshold (e.g., 3) are thrown out. We experimented with feature selection for our ME models based on count thresholding, but observed that keeping all features active in the training data (i.e., using a threshold of 1) and smoothing performed better in our experiments.

4.1.2. ME smoothing

There are several methods to smooth ME models to avoid overtraining (Chen and Rosenfeld, 2000). The most effective smoothing method, as shown in Chen and Rosenfeld (2000), is an instance of Fuzzy ME smoothing. This type of smoothing amounts to adding a Gaussian prior for each parameter with mean zero. The only smoothing parameters to be determined are variance terms for each Gaussian. In our experiments, we used the same variance value for all model parameters. This fixed value was optimized on a held-out set using Powell's algorithm. Using Fuzzy ME smoothing improved the performance more than simple feature selection methods.

4.2. A two-level semantic-lexical language model

In language modeling, we are interested in estimating the probability $P(W)$ of a sentence W. To use semantic parser information in language modeling, we can obtain the most likely parse of a

hypothesized sentence and recompute its LM score using the information in the parse tree. Formally, we can compute $P(W)$ by summing over all possible semantic parses C:

$$P(W) = \sum_C P(W, C).$$

However, we make the Viterbi approximation and use only the most likely parse to simplify computation. Furthermore, we assume that our semantic parser will output a parse that approximates the most likely parse well (according to $P(W,C)$), giving us

$$P(W) \approx \max_C P(W, C) \approx P(W, C_W),$$

where C_W is the parse returned by our parser.

In our two-level model (TLM), we decompose our joint model $P(W,C)$ as follows:

$$P(W, C) = P(W|C)P(C).$$

This is a generative model where the word sequence W is assumed to be generated from the semantic parse C.

Although our statistical parser returns an estimate of $P(C|W)$, we do not use this information to compute $P(C)$. Instead, we first simplify the parse information to be what we call the parent label sequence $L = L_1 \ldots L_N$. This is a sequence that has the same length as the word sequence and which consists of the parent label names in the parse for each word. The parent label sequence for the first part of the sentence in Fig. 2 is shown in Table 1.

Then, the parse probability can be written as

$$P(C) \equiv P(L) = \prod_{i=1}^{n} P(L_i|L_1, \ldots, L_{i-1}).$$

We parametrize $P(L_i|L_1,\ldots,L_{i-1})$ as a maximum entropy model. The model is trained using the parent label sequences extracted from semantic parses of the in-domain training data. We use the following context question types $q(h)$ to estimate $P(L_i|L_1,\ldots,L_{i-1})$:

- Previous label: L_{i-1}.
- Two previous unique labels: L_{i-1} and L_j where j is the largest value that satisfies $j < i-1$ and $L_j \neq L_{i-1}$.
- Previous label and how many times it was repeated: L_{i-1} and $M_{i-1} = i-1-j$, where j is defined as above.

Note that we assume that we can ask questions only about the label sequence, since in this generative model, we generate the word sequence only after the label sequence is complete. However, for the word sequence probability, $P(W|C)$, we can ask context questions about the parent label sequence.

Table 1
Parent label sequence and word sequence

L→	!S!	!S!	!S!	!S!	!S!	RT-OW	RT-OW	!S!	!S!	LOC	LOC
W→	I	want	to	book	a	round	trip	ticket	to	Denver	Colorado

Maximum entropy modeling is used for the probability $P(W|C)$ as well. We have $P(W|C) = \prod_{i=1}^{n} P(w_i|w_1, \ldots, w_{i-1}, C)$, and we use the following context question types $q(h)$ in estimating $P(w_i|w_1, \ldots, w_{i-1}, C)$:

- previous word: w_{i-1};
- previous two words: w_{i-2}, w_{i-1};
- parent label: L_i;
- parent label L_i and number of tokens M_i to the left since this label started;
- L_i, w_{i-1};
- L_i, w_{i-2}, w_{i-1}.

These context question types are used due to our belief that they would be useful in estimating the probability of the current word given the context. We assume here that the current parent label and the previous two words have the most influence on the word probability. Thus, we do not use the surrounding labels in the questions list.

We applied this method, the two-level method, to only semantic classer output. We present our results using this method in Section 5. We did not explore this method in detail since we found that the joint model approach, which we explore next, proved to be more beneficial.

4.3. Joint semantic-lexical probability model

In the previous section, we decomposed the probability computation into two parts. However, it is possible to jointly calculate the probability of the word sequence and the parse within a single model. We are interested in computing the probability of the word sequence $P(W)$ using the semantic features present in the parse C_W as returned by our statistical parser. As before, we take

$$P(W) \approx P(W, C_W).$$

To facilitate the computation of the joint probability, we use the bracket notation introduced earlier to express the semantic parse. Here is the bracket notation representation of the semantic classer output shown in Fig. 2:

- `[!S! i_word want_word to_word book_word a_word [RT-OW round_rt-ow trip_rt-ow RT-OW] ticket_word to_word [LOC Denver_city Colorado_state LOC] for_word [DATE september_month fifteenth_date DATE] !S!].`

This representation completely defines the sentence and its parse. This representation makes it easy to define a joint statistical model since it enables us to compute the probability of both word and label tokens using similar context information. As mentioned before, we use word classes from the class-based LM as our lexical items and multiply the resultant probabilities with $P(w_i|c(w_i))$ to obtain word probabilities. Since our LM classes are very rich and semantic tags have only limited information in addition to the LM classes, we decided to ignore word tags in our formulation. Constituent labels, however, include more long-range information, so we focused on using features that involve labels and words.

We consider every token in the bracket representation as an outcome of the joint model. Thus, our outcome vocabulary is $\mathscr{T} = \mathscr{W} \cup \mathscr{B}_1 \cup \mathscr{E}_1$, where $\mathscr{W}$ is the word (or class) vocabulary and $\mathscr{B}_1$ and $\mathscr{E}_1$ are the begin and end label vocabularies, containing entries of the type `[LABEL` and `LABEL]`, respectively. Then, we represent the joint probability $P(W,C)$ as

$$P(W,C) = \sum_{i=1}^{M} P(t_i|t_1,\ldots,t_{i-1}),$$

where $t_i \in \mathscr{T}$ are individual tokens in the textual representation presented above.

A regular token N-gram model can be built based on this representation. Instead, we built an ME model that contains only N-gram features. We call this model MELM1. MELM1 uses the following question types $q(h)$ to estimate $P(t_i|t_1,\ldots,t_{i-1})$:

1. Unigram history (empty history).
2. Bigram history: t_{i-1}.
3. Trigram history: t_{i-2},t_{i-1}.
4. 4-Gram history: t_{i-3},t_{i-2},t_{i-1}.

Additionally, it is possible to use more intelligent features that capture more long-range and high-level information. Taking data sparsity and computation requirements into account, we came up with the following set of context question types:

- Unigram history (empty history).
- Previous word w_i^{-1} (for bigram features, skipping label tokens).
- Two previous words, w_i^{-2} and w_i^{-1} (for trigram features, skipping label tokens).
- Immediate parent label for the token L_i.
- L_i and N_i, the number of words to the left since starting the current constituent.
- L_i, N_i and w_i^{-1}.
- O_i, the most recently completed constituent and M_i, the number of words to the left since completing O_i.

We call this model MELM2. In this model, the context h_i can be represented by the six-tuple $(w_i^{-1}, w_i^{-2}, L_i, N_i, O_i, M_i)$ introduced above and LM probabilities can be computed by

$$P(t_i|h_i) = e^{\sum_j \lambda_j f_j(t_i,h_i)}/Z(h_i),$$

where each f_j is associated with one outcome token and one question that belongs to one of the types listed above, and $Z(h_i)$ is the normalization term.

While we selected this particular question set, there are many other possible features that we could use that utilize additional information, such as word tags, grandparent labels, etc. Which feature types are best can depend on the domain or the type of semantic parsing employed. The maximum entropy framework enables one to incorporate any type of computable features in a unified manner.

In our final model, we combine the semantic classer and parser output to obtain a full semantic parse for each sentence. The full semantic parse contains all derivations starting from the word

level to the sentence symbol including the classer and parser derivations together. The full semantic parse of a sentence is shown in Fig. 4. We constructed a new statistical model called MELM3 using features from the full semantic parse. The features we used for MELM3 are:

- unigram history (empty history);
- previous word w_i^{-1};
- two previous words, w_i^{-2} and w_i^{-1};
- L_i and N_i;
- L_i, and the grandparent label G_i;
- O_i and M_i.

Thus, we add the grandparent label to the set of elements that we can ask questions about. We observed that using full semantic parse information helped improve performance.

A potential problem with our MELM models is that they assign nonzero probabilities to parses that are invalid. For example the following invalid parse:

- `!S!] yes_word [!S!,`

will receive a nonzero score. However, this is not an issue for us in practice since in our experiments, we apply our models only in rescoring valid parses.

We evaluated the MELMs by rescoring N-best hypotheses on three different task domains and test sets. The results are presented in the following section.

5. Experiments and results

We evaluated the proposed methods on three tasks by rescoring N-best hypotheses from a conventional speech recognizer. These tasks are the air travel reservation (Erdogan et al., 2002), military

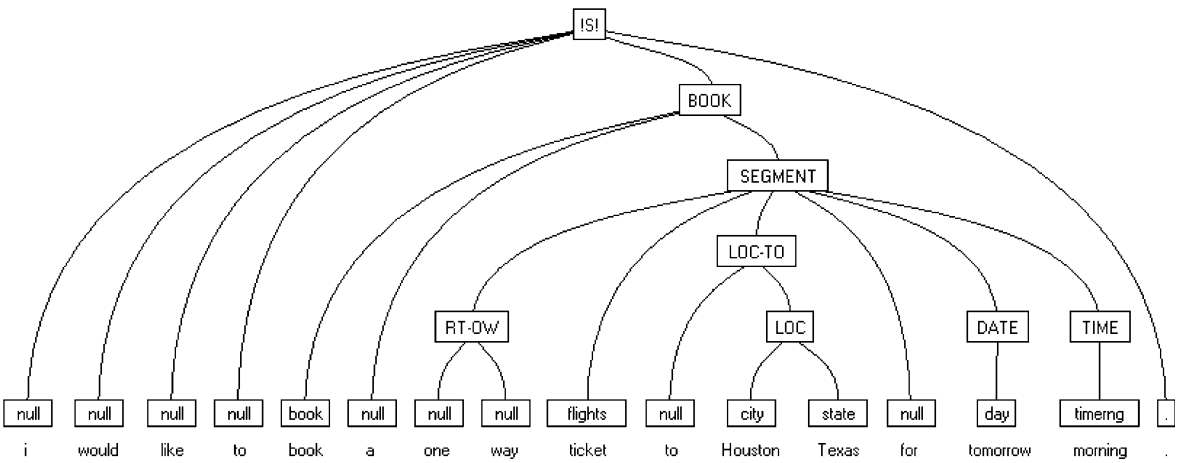

Fig. 4. Full semantic parse of a sentence.

force-protection, and financial transaction domains (Sarikaya et al., 2004). We first report results for an extensive set of experiments on the air travel reservation task, then report results for a more limited set of experiments on the other two domains. One issue that arose is that it makes sense to apply MELM3 only when parse trees of depth larger than two are available. However, for the force-protection and financial transaction domains, training data annotation was largely or exclusively of depth two or less, so we did not apply MELM3 in these domains.

Perplexity is extensively used in comparing language models (Jurafsky and Martin, 2000). In this paper, we do not present perplexity results since our joint semantic-lexical models compute the joint probability $P(W,C)$ rather than $P(W)$ and the number of distinct tokens is different among models. Thus, it is not possible to directly compare perplexities between models. In any case, word-error rates are a more relevant measure for comparing language models and we present word-error rates for three different domains in the following discussion.

5.1. Air travel reservation domain

The first set of experiments was performed in the air travel reservation domain. The acoustic models were trained using air travel and generic telephony data (Gao et al., 2001). The language model training data consisted of about 137 K sentences in the air travel domain. A held-out set of 18 K sentences was used for smoothing. The class-based trigram and concept N-gram language models were smoothed using held-out interpolation. For the maximum entropy models, all candidate features active in the training data were kept as model features. The candidate feature set is created by taking the cross product of all possible context questions $q(h)$ and all possible word/label outcomes. The ME models are trained using the improved iterative scaling algorithm (Pietra et al., 1997) with fuzzy ME smoothing (Chen and Rosenfeld, 2000) using a single universal variance parameter of 3 for MELM1 and MELM2 and 1.5 for MELM3. The held-out data set was used to determine the optimum value of this parameter. The test set has 1173 utterances from calls received by IBM in the DARPA Communicator evaluation in June 2000.

We have already spent much effort to improve language model performance and have achieved good performance with class-based and compound-word language models and rule-based semantic tokenization of such structures as time, date, and flight number (Erdogan, 2001). These techniques model the semantic and syntactic structure of sentences to some extent and are included in the baseline class-based trigram language model that we used.

To evaluate the error rate performance of the new LM's, a lattice with low oracle error rate was generated by a Viterbi decoder using a class trigram language model. From the lattice, we generated at most 300 sentences for each utterance to form an N-best list. We rescored these utterances using the proposed language models and our baseline word and class trigram language models. The results are presented in Table 2.

In Table 2, the first entry is the oracle error rate of the N-best lists, i.e., the error rate yielded if we choose the lowest WER hypothesis in the N-best list for each utterance. This represents a lower bound on rescoring performance. The second and third rows show the error rate for the word and class trigram LM's (Erdogan, 2001; Gao et al., 2001). The error rate for the word trigram LM would have been higher if decoding was done directly with this LM, but looks artificially better due to N-best list rescoring. The next two entries correspond to the concept sequence LM which is a log-linear interpolation of the concept trigram and the class trigram; the entries differ in which

Table 2
Word error rates obtained using various language modeling methods

Method	WER (%)
N-best oracle	8.8
Word trigram	18.9
Class trigram	17.7
Concept Seq-WFST + class trigram	17.3
Concept Seq-STC + class trigram	17.5
TLM-STC	18.9
MELM1-STC	17.7
MELM2-STC	17.4
MELM3-STP	17.4
MELM1-WFST	17.6
MELM2-WFST	17.7

shallow parser was used. The interpolation weight was chosen to be 0.2. This approach reduces the error rate about 2–3% relative over the class trigram model. The two-level semantic-lexical model (TLM) does not achieve a reduction in WER; it only achieves the same performance as the word trigram model. We attribute this bad performance to the crudeness of the model. Thus, we do not investigate this method further, since joint models result in better performance.

The following entries are the error rates for the joint semantic-lexical maximum entropy language models. MELM-STC entries use the statistical classer (STC) to obtain the semantic features, whereas MELM-WFST entries use the WFST classer. The results show that MELM1-STC can achieve the same error rate as the class trigram LM. MELM2-STC performs slightly better probably since it uses longer range features in the probability computation. WFST-based MELM1 and MELM2 also achieve similar results as compared to the statistical classer-based models. Finally, MELM3-STP, which uses the full parser, achieves a 17.4% error rate, which is the same as for MELM2-STC.

Further improvement is possible when multiple language models are interpolated. There are two basic ways of interpolating language models: linear and log-linear interpolation. In linear interpolation, the language model score is found by interpolating the individual scores, $\log(\lambda P_1 + (1-\lambda)P_2)$, where P_i is the probability assigned by the ith model. The interpolation weight $\lambda > 0$ can be estimated by optimizing the perplexity of a held-out data set using the EM algorithm. We used a 2000 sentence subset of the held-out data to optimize the interpolation parameter used in interpolating each MELM-STC model with the class trigram model. The MELM weights were 0.41 for MELM1, 0.64 for MELM2, and 0.40 for MELM3. The results are shown in the left-hand column of Table 3. We did not observe any improvement in WER's for linearly interpolated models over the individual MELM-based methods.

Some researchers argue that log-linear combination of LM probabilities leads to better discrimination (Beyerlein, 1997). Thus, we also tried to interpolate the language models in the log domain. In log-linear interpolation, the language model scores are combined as $\lambda \log P_1 + (1-\lambda)\log P_2$. This results in unnormalized LM "probabilities", but still provides an LM score with which to compare sentences. For log-linear interpolation, we chose the simplest

Table 3
Word error rates (%) obtained using linear and log-linear interpolation of MELMs and the class trigram

Method	Linear-interpolation	Log-linear interpolation
Class trigram	17.7	17.7
MELM1-STC + class trigram	17.5	17.5
MELM2-STC + class trigram	17.5	17.3
MELM3-STP + class trigram	17.4	16.8

approach and used the same weight for each LM; thus, the language model score is computed as $\log P_1 + \log P_2$, where P_i is the probability assigned by the ith model.

The results of log-linear interpolation are shown in the right-hand column of Table 3. Here, we observe that we can achieve a significant reduction in the word error rate using the MELM3 and class trigram mixture over MELM3 alone. We conjecture that the LM scores provided by the class trigram and by MELM3 are complementary to each other. Log-linear interpolation achieves a high score when both language models receive a high score. This is unlike linear interpolation, where one high score is enough to guarantee a high LM score. Thus, log-linear interpolation works better between MELM3 and the class trigram, yielding a 0.9% absolute drop in WER over our class trigram baseline.

5.2. Military domain

We note that the DARPA Communicator project was a multi-year effort for which we spent much effort to improve language model performance over several years. However, rapid deployment of spoken dialog systems in new domains rules out domain-specific tuning of the language modeling to a great extent.

The second set of data we used was collected for a speech-to-speech translation project in a military force-protection domain where soldiers interact with civilians. Unlike DARPA Communicator, there is no clear structure for the dialogs. The dialogs can be about a number of topics. For example, a dialog can be about a search for a specific vehicle. A soldier can ask a civilian if they saw such a vehicle and if so, when it was seen and in which direction it was proceeding. A dialog could also be about entering a military base. If a civilian wants to enter a base, a soldier can ask questions about who they want to see, whether they are carrying any weapons, and so on. Therefore, building class-based language models is not straightforward as compared to the air travel reservation task. As such, we had to adopt an annotation strategy that used a mixture of semantic and syntactic concepts. For example, in the following bracketed sentence some words are annotated using semantic concepts, while others are annotated using syntactic concepts.

```
[!S! [WHQ what=WHQ time=WHQ WHQ] [AUX did=AUX AUX] [PRON you=PRON-SUB
PRON] [VERB see=VERB VERB] [PERSON the=ART person=PERSON PERSON] !S!]
```

We had access to only a small amount of data in this domain. The data available for training contains 5882 sentences. This data is split into two parts for language model training: 500 sentences as heldout and 5382 sentences as the training set. The vocabulary size is 1887. We built a word trigram language model as our baseline. The test set contained 718 sentences. The N-best lists used were generated by using the baseline language model with $N = 300$. The N-best lists had

an oracle WER of 9.6%. We adapted a set of generic acoustic models trained for domain-independent tasks using MLLR + MAP (Leggetter and Woodland, 1995; Gauvain and Lee, 1994). The speech data used for adaptation were from the 5882 sentences used for language modeling.

The experimental results are presented in Table 4. The baseline language model obtained a WER of 14.1%. We built only the MELM2-STC model since most of the annotated sentences had parse trees of depth two. The MELM2-STC model achieved a WER of 12.5%, giving a relative improvement of 11.3%. We note that we achieved a much greater improvement over the baseline result as compared to the air-travel domain. In the DARPA Communicator task, the semantic and syntactic relationships are modeled to a great extent by compound words and a carefully designed class-based language model. However in the military domain, the baseline language model does not contain any additional semantic/syntactic information. The proposed method is clearly superior to the word N-gram model. We conjecture that the proposed method represents the semantic and syntactic relationships between words in a sentence better than a word trigram model does.

As shown in Table 4, we also performed interpolation of the trigram and MELM2-STC model scores. Since we did not have any held-out data for interpolation weight training, we used equal weights for each model in both linear and log-linear interpolations. We observe that when we log-linearly interpolate MELM2-STC and trigram model scores, we obtain a further 7% reduction in the word error rate, down to 11.6%. This result confirms our previous conjecture that the LM scores provided by the trigram and MELM models are complementary to each other. We also notice that linear interpolation does not improve the WER, similar to the case for the air travel domain.

5.3. Financial transaction domain

The third set of experiments were conducted on a financial transaction task where people called a system to perform transactions such as balance inquiries and fund transfers on their retirement accounts.

Unlike the previous two tasks, we did not manually annotate data to build MELM-based language models. However, a grammar and conversational system expert wrote rule-based grammars to define such concepts as AMOUNT, LOANTYPE, SHARES, etc. We used 17 grammars altogether to represent semantic concepts that exist in this domain. We treated these grammars as constituents and constructed classer trees. Using these grammars, we can build parse trees of depth two, e.g.

Table 4
Word error rates obtained using various language modeling methods for the military force-protection domain

Method	WER (%)
N-best oracle	9.6
Word trigram	14.1
MELM2-STC	12.5
MELM2-STC + word trigram (linear interpolation)	12.5
MELM2-STC + word trigram (log-linear interpolation)	11.6

```
Sentence: balance of equity index
Tokenized Sentence: <BAL-REF> balance </BAL-REF> of <FUND> equity index
</FUND>
Classer tree: [!S! [BAL-REF balance BAL-REF] OF [FUND equity index FUND]
!S!]
```

We trained an MELM2-STC model using 28.3 K semantically annotated sentences (105 K words). The test data consisted of 3152 sentences, amounting to 11.4 K words. The speech recognition acoustic models were trained using generic telephony data. A dialog-state-based word trigram language model (DS-3gr) is used to obtain the baseline WER and to generate N-best lists. The baseline model DS-3gr used a separate 194 K sentences from the financial domain as training and an additional 10 K sentences as held-out data.

The N-best lists contained an average of 34 alternative hypotheses per sentence with an oracle WER of 16.2%. Here, MELM2-STC uses dialog state as an additional feature. The MELM2-STC model was used to rescore the N-best lists. Table 5 shows the results for DS-3gr and MELM2-STC. We see that using MELM2-STC resulted in a 15.7% relative improvement. These improvements are due to the inclusion of semantic information that was not modeled in the baseline LM.

In this domain, linear interpolation of the trigram and MELM2-STC model scores resulted in worse performance than the MELM2-STC model alone. This is probably due to the bad performance of the word trigram model. However, log-linear interpolation with equal weights improves the WER by 0.2% absolute. This result once again confirms our previous theory that the LM scores provided by the trigram and MELM models are complementary to each other.

Overall, if the baseline language model is of high quality and incorporates some semantic and syntactic knowledge as in the case of DARPA Communicator, we can achieve about 3–6% relative reduction in error rate after log-linearly interpolating our MELMs with a class trigram language model. These gains are similar to the gains that were achieved with structured LM's over plain word trigram models in other domains (Chelba and Jelinek, 1999; Wu and Khudanpur, 1999).

In our work, if the baseline language model is a conventional word trigram, there is substantial value in using MELM-based methods. For a military domain, the improvement over the baseline word trigram language model is 11.3%. For a financial domain, we achieved a 15.7% improvement over the baseline dialog-state-based word trigram language model. The size of the improvement for the financial domain did not surprise us, since the grammar design step took several months to complete and was optimized for this domain to provide accurate semantic analysis

Table 5
Word error rates obtained using various language modeling methods for the financial transaction domain

Method	WER (%)
N-best oracle	16.2
Dialog state based word trigram (DS-3gr)	28.1
MELM2-STC	23.7
MELM2-STC + word trigram (linear interpolation)	25.77
MELM2-STC + word trigram (log-linear interpolation)	23.51

of the sentences. Consequently, MELM2-STC jointly modeled the word sequence and grammar-based concepts.

6. Conclusion

We introduced several new language models that use high-level long-range semantic information to improve language modeling for speech recognition. All of these models use semantic analyzers to aid in language modeling.

We evaluated MELM-based methods across several different domains using different amounts of training data and different annotation schemes (i.e., grammar-based and manual annotation). The improvement over word trigram language models was significant and consistent across tasks. When the baseline language model is a well-developed and hand-crafted class-based LM, we obtain a small 1–2% relative improvement with the new models and 5% relative improvement after log-linear interpolation with the baseline model. However, when we use an out-of-the-box word trigram model as a baseline, we obtain a much higher 10–15% relative improvement when using the models and up to 7% additional improvement after log-linear interpolation with the baseline model.

We observe that incorporating semantic features improves language modeling in general. Furthermore, in our air travel reservation domain experiment, we compare different semantic language modeling techniques and features. We observe that how the semantic information is integrated makes a difference as well. A tight integration of semantic and lexical information using joint maximum entropy modeling results in more complementary information with respect to N-gram language models as compared to the concept sequence or two-level methods, which can be viewed as loose integration models. A substantial improvement is obtained when scores from an N-gram model and a tightly-integrating MELM model (MELM3) are interpolated in a log-linear fashion, indicating the complementarity of the two models. We verify that maximum entropy modeling is an effective tool for unifying many sources of information.

In designing dialog systems, one usually builds a semantic analyzer for the natural language understanding module of the system. This analyzer can be easily used in rescoring N-best lists generated by a speech recognizer using the semantic-lexical language modeling framework proposed in this paper.

In the future, we plan to utilize both syntactic and semantic information together to improve language modeling. This should not be difficult as maximum entropy provides a framework for incorporating additional types of features in our models in a straightforward manner.

Acknowledgements

The authors thank Adwait Ratnaparkhi for the use of his code implementing maximum entropy training and testing algorithms and Mike Monkowski for designing grammars in the financial domain.

References

Berger, Adam, Pietra, Stephen Della, Pietra, Vincent Della, 1996. A maximum entropy approach to natural language processing. Computational Linguistics 22.

Beyerlein, P., 1997. Discriminative model combination. In: Automatic Speech Recognition and Understanding Workshop, pp. 238–245.

Brown, Peter F., Pietra, Stephen A. Della, Pietra, Vincent J. Della, Lai, J.C., Mercer, Robert L., 1992. Class-based n-gram models of natural language. Computational Linguistics 18 (4).

Charniak, Eugene, 2001. Immediate-head parsing for language models. In: Proceedings of the 39th Annual Meeting of the ACL, pp. 116–123.

Chelba, Ciprian, Jelinek, Frederick, 1999. Recognition performance of a structured language model. In: Eurospeech.

Chelba, Ciprian, Jelinek, Frederick, 2000. Structured language modeling. Computer Speech and Language 14 (4), 283–332.

Chen, Stanley F., 2000. The IBM finite-state machine toolkit version 1.0. Technical Report, IBM, February.

Chen, Stanley F., Goodman, Joshua, 1998. An empirical study of smoothing techniques for language modeling. Technical Report 1098, Center for Research in Computing Technology, Harward University, August.

Chen, Stanley F., Rosenfeld, Ronald, 2000. A survey of smoothing techniques for ME models. IEEE Transactions on Speech and Audio Processing 8 (1), 37–50.

Davies, Ken, et al., 1999. The IBM conversational telephony system for financial applications. In: Eurospeech.

Erdogan, Hakan, 2001. Speech recognition for a travel reservation system. In: International Conference on Artificial Intelligence.

Erdogan, Hakan, Sarikaya, Ruhi, Gao, Yuqing, Picheny, Michael, 2002. Semantic structured language models. In: ICSLP.

Gao, Yuqing, Erdogan, Hakan, Li, Yongxin, Goel, Vaibhava, Picheny, Michael, 2001. Recent advances in speech recognition system for IBM Darpa communicator. In: Eurospeech, pp. 503–506.

Gauvain, Jean-Luc, Lee, Chin-Hui, 1994. Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains. IEEE Transactions on Speech and Audio Processing 2 (2), 291–299.

Gavalda, M., 2000. SOUP: a parser for real world spontaneous speech. In: Proceedings of the IWPT.

Hacioglu, Kadri, Ward, Wayne, 2001a. Dialog context dependent language modeling combining n-grams and stochastic context-free grammars. In: Proceedings of the IEEE Conference on Acoustics Speech and Signal Processing.

Hacioglu, Kadri, Ward, Wayne, 2001b. A word graph interface for a flexible concept based speech understanding framework. In: Eurospeech.

Iyer, R., Ostendorf, Mari, 1996. Modeling long distance dependence in language: topic mixtures vs. dynamic cache models. In: ICSLP, pp. 236–239.

Jelinek, Frederick, Chelba, Ciprian, 1999. Putting language into language modeling. In: Eurospeech.

Jelinek, Frederick, Mercer, Robert L., 1980. Interpolated estimation of Markov source parameters from sparse data. In: Workshop on Pattern Recognition in Practice.

Jurafsky, Daniel, Martin, James H., 2000. Speech and Language Processing. Prentice Hall, Englewood Cliffs, NJ.

Khudanpur, Sanjeev, Wu, Jun, 2000. Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling. Computer Speech and Language (October), 355–372.

Langley, Chad, Lavie, Alon, Levin, Lori, Wallace, Dorcas, Gates, Donna, Peterson, Kay, 2002. Spoken language parsing using phrase-level grammars and trainable classifiers. In: ACL-02 Speech to Speech Translation Workshop, pp. 15–22.

Leggetter, C.J., Woodland, P.C., 1995. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. Computer Speech and Language 9, 171–185.

Magerman, David M., 1994. Natural language parsing as statistical pattern recognition. Ph.D. Thesis, Stanford University, Palo Alto, CA, February.

Monkowski, M., 2001. Embedded grammar objects within language models. Technical Report, IBM.

Pietra, Stephen Della, Pietra, Vincent Della, Lafferty, John, 1997. Inducing features of random fields. IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (4), 380–393.

Roark, Brian, 2001. Probabilistic top-down parsing and language modeling. Computational Linguistics 27 (2), 2–28.
Rosenfeld, Ronald, Chen, Stanley F., Zhu, Xiaojin, 2001. Whole sentence exponential language models: a vehicle for linguistic-statistical integration. Computer Speech and Language 15 (1).
Sarikaya, Ruhi, Gao, Yuqing, Picheny, Michael, 2004. A comparison of rule-based and statistical methods for semantic language modeling and confidence measurement. In: HLT/NAACL Conference.
Visweswariah, Karthik, Printz, Harry, 2001. Language models conditioned on dialog state. In: Eurospeech.
Ward, Todd, 2000. How long until a high school student can build a language understanding system. In: ICSLP.
Wu, Jun, Khudanpur, Sanjeev, 1999. Combining nonlocal syntactic and n-gram dependencies in language modeling. In: Eurospeech.
Wu, Jun, Khudanpur, Sanjeev, 2000. Syntactic heads in statistical language modeling. In: Proceedings of the IEEE Conference on Acoustics Speech and Signal Processing.