

IS WORD ERROR RATE A GOOD INDICATOR FOR SPOKEN LANGUAGE UNDERSTANDING ACCURACY

Ye-Yi Wang, Alex Acero and Ciprian Chelba
Speech Technology Group, Microsoft Research

ABSTRACT

It is a conventional wisdom in the speech community that better speech recognition accuracy is a good indicator for better spoken language understanding accuracy, given a fixed understanding component. The findings in this work reveal that this is not always the case. More important than word error rate reduction, the language model for recognition should be trained to match the optimization objective for understanding. In this work, we applied a spoken language understanding model as the language model in speech recognition. The model was obtained with an example-based learning algorithm that optimized the understanding accuracy. Although the speech recognition word error rate is 46% higher than the trigram model, the overall slot understanding error can be reduced by as much as 17%.

1. INTRODUCTION

Speech recognition technology has made tremendous progress over the past decades. Accompanying its maturity and its potentials for commercial applications, extensive research has been devoted to the learning technologies that can ease the development of a speech understanding system [1]. Researchers have been investigating example-based grammar learning, ranging from unsupervised grammar induction [2], to the semi-supervised grammar learning [3], to the supervised acquisition of statistical understanding model [4], and to the “learning-by-doing” paradigm for grammar development [5]. Most (if not all) of the approaches treat understanding as a separate problem, independent of speech recognition. A two pass approach is often adopted, in which a domain-specific n-gram language model is constructed and used for speech recognition in the first pass, and the understanding model obtained with various learning algorithms is applied in the second pass to “understand” the output from the speech recognizer. While this is a practical solution, we believe it is suboptimal due to the following two reasons: first, the objective function being optimized when building an n-gram language model is the reduction of the test data perplexity, which is related to the reduction of the speech

recognition word error rate, although that is not always the case. It does not necessarily imply the reduction of overall understanding error rate. Secondly, a large amount of training data is rarely available for the developments of many speech applications. An n-gram trained on a small amount of data often yields poor accuracy. It is thus desirable to include prior knowledge (e.g., domain knowledge and grammar models for domain-independent concepts) in a language model whenever this is possible. Constrained domains, such as the air travel information system (ATIS) [6], may allow the use of prior knowledge to compensate for the lack of language modeling training data.

In the past couple of years, we have developed SGStudio, an example-based grammar learning/development tool [7]. The goal is to help developers create a high quality model for text-based understanding. Different from many pure data-driven studies, it combines example-based learning with prior knowledge. The prior knowledge includes manually developed reusable grammars for domain independent concepts, such as date, time, credit card number, etc.; as well as the domain knowledge that can be obtained from the application database, including the application schema and the domain specific concepts, e.g. the airport names in the ATIS domain. Given an input sentence, SGStudio “guesses” its meaning and represents it in a structure according to the schema of the domain. Grammar developers will either acknowledge the guess or make necessary corrections, such that the tool can modify the underlying model to increase the likelihood of the new example. Over the course of the investigation, we have come up with several different underlying understanding models. The last and the best one is a statistical model that is a composition of HMM and CFGs, which had around 50% error reduction over a manually developed grammar. Unlike its predecessors, this model does not depend on a robust parser for robust understanding. Instead, the robustness feature is built-in in the model itself, which allows us to use it as a language model for speech recognition.

This paper investigates the new language model’s impact on word error rate and language understanding error rate.

The next section reviews the new statistical models adopted by SGStudio for language understanding. Following that we introduce its context-free grammar representation that can be accepted by speech-recognizers. Finally we discuss the experimental setting and results.

2. SEMANTIC UNDERSTANDING MODEL

The semantic understanding model uses an HMM to encode the structural information of the application schema, and uses a CFG to model the emissions of some HMM states. Here we use an example to illustrate the topology of the model. Assume that we are interested in the ATIS domain, which has the following (simplified) application schema:

```
<task name="ShowFlight">
  <slot type="City" name="ACity"/>
  <slot type="City" name="DCity"/>
</task>
<task name="GroundTransport">
  <slot type="City" name="City"/>
  <slot type="Transport_Type" name="TType"/>
</task>
```

The schema simply states that the application supports two types of information queries: those for flight information (the ShowFlight task) and those for ground transportation information (the GroundTransport task). To get flight information, a user has to provide information about the arrival city (ACity) and/or the departure city (DCity) slots, so the system can search for the information according to the user's specification. The type of a slot specifies the requirement for its "fillers". For both ACity and DCity slots, the filler must be an expression modeled in the grammar library that refers to an object of the type "City".

The semantic constraints in the schema are incorporated in the understanding grammar with the HMM illustrated in Figure 1. The top level HMM has two branches to the ShowFlight and GroundTransport sub-networks. The transition weights on the branches are the probabilities for the two tasks. The ShowFlight network in the bottom models the linguistic expressions that users may use to issue a ShowFlight command. It starts with a command part (e.g., "Show me the flight"), followed by the expressions for the slots. Each slot is bracketed by a preamble and a post-amble, which serve as the linguistic context for the slot. For example, the word "from" is a preamble for the DCity slot. It signals that the city following it is likely to be a departure city. The slots are inter-connected. The connections are weighted with the bigram probability for slot transitions, which is estimated from the training data.

In the network, the command, preambles and post-ambles are modeled with statistical n-gram models. The slot fillers are modeled with probabilistic CFG rules from a grammar library. The probabilities for the rules in the grammar library are tuned with the domain specific data and smoothed properly. Because of the inclusion of the CFG library, the model is a composition of HMM and CFG.

The n-grams in the model are trained with partially labeled training data. An example of the labeled data is illustrated in Figure 2. It labels the task and slot information in a training sentence. The alignment between the rest of the words in the sentence and the model states (commands, preambles and post-ambles) is not provided. An EM algorithm was used to train the n-grams in the network [8], where the alignments are treated as hidden variables. The training will result in a model that maximizes the likelihood of the observed data --- the semantic structures in the annotated training data.

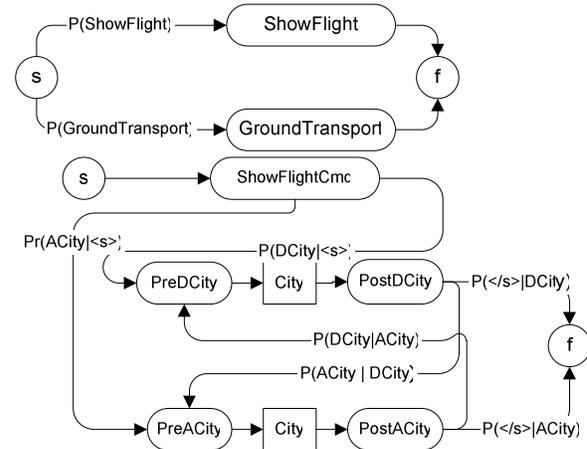


Figure 1. The HMM structure created according to the semantic schema. The upper network is the top level grammar that has two sub-networks. The lower network shows the details of the ShowFlight model. The probabilities are estimated from the training data. The rectangular blocks are modeled with CFG rules; the rounded rectangular blocks are modeled with n-grams.

```
<ShowFlight text="show me the flight from Seattle to Boston">
  <DCity text="Seattle"/>
  <ACity text="Boston"/>
</ShowFlight>
```

Figure 2. A labeled training data sample.

A dynamic-programming algorithm [7] was introduced to find the best semantic interpretation of an input sentence. The model achieved 32% error reduction over our previous CFG model/robust parsing technology.

The overall structure of our model is very similar to that of the Hidden Understanding Model [4]. The “indicators” in HUM functions similarly as our preambles. Both were modeled with n-grams. The major difference is that our model does not try to learn everything from data. Instead we take advantage of grammar library. Because of that, the semantic structure exposed to the user is much simpler. For example, it is up to the library grammar to “understand” what type of Date the word “Friday” is, while the HUM requires developers explicitly annotate it as DayOfWeek. For the same reason, our model requires much less data to get satisfactory accuracy. On the other hand, since there is no guarantee that a third party library grammar is finite state, our model has to be a composite of HMM and CFG, which requires a more complicated decoder; while the HUM is purely an HMM. The inclusion of post-ambles in our model makes it more precise --- a preamble-only model will not account for the words appearing after the last slot. Modeling in finer granularity also makes the model generalize better.

3. SGSTUDIO GRAMMAR AS THE UNIFIED LANGUAGE MODEL

Unlike our previous robust understanding technology, which relied on a robust parser to skip the words not covered by a grammar, the use of n-gram models for the pre-terminals in a grammar makes the model robust in itself. This offers a new opportunity for using the HMM/CFG composite model in speech recognition. Since the optimization objective of the training algorithm is to maximize the likelihood of the observed semantic structures in the annotated training data instead of reducing the perplexity (in other words, maximizing the likelihood of the training sentences,) we can overcome the sub-optimality problem we discussed previously. The model uses prior knowledge. Therefore it can potentially generalize better.

To use the model for speech recognition, we have to convert it into a format that the recognizers can accept as a language model. Although in our previous unified language model work [9] we have implemented a decoder that supports an n-gram language with embedded CFG rules, there is no decoder that supports a language model with multiple n-grams inside a CFG. Our solution to this problem is to convert the n-gram sub-models into probabilistic finite state automata. The converted n-grams and the top-level HMM structure, together with the rules in the library grammar, form a PCFG language model. The n-gram to CFG conversion is similar to the algorithm described in [10], with a minor modification to make the model more compact: since the n-grams in the HMM/CFG

model are HMM state specific, the training data for each n-gram is very sparse. Many words in the vocabulary are unseen in the EM training for a specific n-gram. Every unseen word results in a self loop over the back-off state due to the smoothing with a uniform LM. Instead of adding a loop for each unseen word, we made an approximation by adding a single loop that refers to a shared uniform distribution (Figure 3).

The resulting automata are represented in the SAPI [11] binary CFG format as well as the HTK [12] Standard Lattice Format for the use in the experiments.

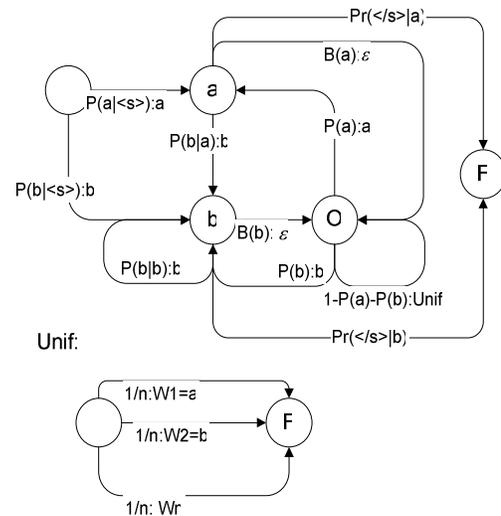


Figure 3. Finite state representation of a bigram language model with two observed words (a,b). The label on an arc shows its weight and output symbol. I represents the initial state, O represents the back-off state, and F represents the final state. Instead of looping over the back-off state for every unseen word, the model is smoothed approximately with a self loop labeled with the uniform distribution over the back-off state.

4. EXPERIMENTAL RESULTS

The experiments were conducted in the ATIS domain. We constructed the semantic schema for ATIS by abstracting the CMU Phoenix grammar for ATIS. 5798 sentences from the class A (utterances that can be understood without context) of the ATIS2 and ATIS3 training data were used to train a trigram language model. The vocabulary of the model contained 780 words. The HMM/CFG model covered the same vocabulary, although it was trained with only ~1700 sentences from class A of the ATIS3 training data. The sentences were annotated in a format similar to the example in Figure 2. The 469 sentences from the class A of the 1993 ATIS3 test data

were also annotated. It was used as the reference semantic structures in the language understanding evaluation. The commands, preambles and post-ambles in the HMM/CFG model were modeled with bigrams. Two HMM/CFG models were trained. In the first model the bigrams were not smoothed. In the second model, the bigrams were smoothed with the uniform distribution with deleted interpolation [13]. The test data perplexity of the trigram model is 15.4. For the smoothed HMM/CFG model, the test data perplexity is 16.2 when the likelihood of a sentence is summed over all possible paths in the network. We call it the *Baum-Welch perplexity*. When the likelihood of a sentence is only calculated over the Viterbi path, the resulting *Viterbi perplexity* is 24.1.

We used the three language models for speech recognition. The recognizer was the commercial product that came with Microsoft SAPI 5. The outputs from the recognizer were sent to the HMM/CFG decoder for language understanding. The outputs were then compared to the manual annotation. The statistics of task classification (henceforth task ID) and slot identification (henceforth slot ID) error rate were collected. Task ID performance was measured by comparing the top-level task (ShowFlight, GroundTransport, etc.) found by the model with the manual label. There were six top-level tasks in the ATIS domain. In slot ID evaluation, slots were extracted by listing all the paths from the root to the pre-terminals in the semantic parse tree, and the resulting list was compared with that from the manually annotated semantic tree. Hence a task ID error will make all the slots in a parse tree being counted as errors in the slot ID evaluation. The total insertion-deletion-substitution error rates are reported for slot ID. Table 1 shows the result.

	n-gram LM	HMM/CFG (US)	HMM/CFG (S)	Transcription
WER	8.2%	12.3%	12.0%	---
Task ID	7.9%	7.1%	5.6%	2.3%
Slot ID	11.6%	11.1%	9.8%	5.1%

Table 1. Recognition word error rate, task classification error rate and slot identification error rate of the trigram model, the unsmoothed HMM/CFG model and the smoothed HMM/CFG model. The results were obtained with the commercial recognizer in SAPI 5. The mismatched acoustic model and aggressive pruning attributed to the high word error rate.

Even though the word error rate is over 46% higher than the trigram model, the HMM/CFG model achieved the task classification error rate that is almost 30% lower than the trigram model, and the slot identification error rate 17% lower. We noticed that the understanding error rate

reduction was even bigger as the word error rates for all the three models became higher when a larger vocabulary was used.

The recognition error for the HMM/CFG model often occurs in the command, preamble and post-amble part. Naturally this is due to the split of training data over many different pre-terminals. The sparseness of the training data for a pre-terminal makes the recognition of words underneath the pre-terminal less accurate. However, since the understanding model is robust, a word error inside this pre-terminal doesn't matter too much as long as it will not flip to another pre-terminal. An example of this is given below:

Reference	find me a flight that flies from Memphis to Tacoma
Trigram	find me a flight that flies from Memphis to Tacoma
HMM/CFG	find me a flight the flights from Memphis to Tacoma

Here although "that flies" was misrecognized as "the flights" with the HMM/CFG model, it did not change its status as the preamble of a flight slot. The meaning was not affected at all.

On the other hand, the trigram model lacks the stricter constraints imposed by the rules in CFG library, therefore the content of a slot often get recognized incorrectly. This will cause slot ID errors. Since the task ID also depends on the correct slot information, this may adversely affect the task ID accuracy too. Below is the example of this case.

Reference	list the originating cities for Alaska airlines
Trigram	list the originating is the cities for last the airlines
HMM/CFG	list the originating fit cities for Alaska airlines

Compared to the best recognition performance for ATIS, the word error rate in the experiment is a bit too high. We believe it can be attributed to the mismatched acoustic model as well as the aggressive pruning of the commercial recognizer --- The decoder takes only one third of the time used by the HapiVite decoder (see the experiment below) when trigram is used as the language model, and 4% of the time consumed by HapiVite when the smoothed HMM/CFG model is used.

We would like to compare the models for understanding accuracy when the recognition error is lower. So we repeated the experiment using an acoustic model trained

using HTK [12] on ATIS data and the HapiVite decoder. Table 2 shows the results.

The optimal language model weight for the HMM/CFG model is 26, which is much higher than that for the trigram model (16). This is because the language model probability mass is split and distributed over multiple ambiguous paths in the HMM/CFG state space, while with the trigram model a word sequence corresponds to a single language model state sequence. Therefore the language model score in a path in the HMM/CFG state space needs to be boosted.

	n-gram LM	HMM/CFG (US)	HMM/CFG (S)	Transcription
WER	6.0%	9.2%	7.6%	---
Task ID	6.8%	4.9%	3.8%	2.3%
Slot ID	9.0%	10.3%	8.8%	5.1%

Table 2. Recognition word error rate, task classification error rate and slot identification error rate of the trigram model, the unsmoothed HMM/CFG model and the smoothed HMM/CFG model. The results were obtained with the HTK decoder. The matched acoustic model and less aggressive pruning resulted in the better word error rate. It took tremendously longer time to recognize an utterance.

The word error rate of the HMM/CFG model is about 27% higher than the trigram model. However, the task classification error rate is more than 40% lower. The advantage of the HMM/CFG in slot error rate diminished to 2.5% improvement over the trigram model. It appears that the slot error rate, which depends more on the actual text being recognized, is more correlated to the word error rate when the word error rate is low. When the word error rate is higher due to reasons other than the language model, the advantage of the HMM/CFG model is more obvious.

Several slot errors are related to the context free nature of the new language model. For example, “New York City area” was misrecognized as “New York City Arizona,” and “Arizona” was further taken as a slot. The model properly learned that it is likely to have a city name followed by a state name, while it lacked the lexical constraints to restrict Arizona from being recognized as the state where the New York City is.

The decoder using the n-gram model ran much faster than the HMM/CFG language model. While the commercial decoder using trigram as the language model recognized utterances in about 0.5x real-time, it took about 85x real-

times to decode a sentence when the unsmoothed HMM/CFG was used, and 180x real-times when the smoothed model was used. The HTK decoder, which searched much bigger spaces, took 1.5x real-times to decode an utterance with the trigram language model, 215x real-times with the unsmoothed HMM/CFG model, and 1200x real-times with the smoothed HMM/CFG model. We are currently optimizing the model structure to make it work faster with the decoders. We believe that the proper optimization, together with the advances in CFG decoding technology and the continuing growth of computing power, will make this model ready for practical use.

5. DISCUSSION

Researchers from AT&T Labs-Research have noticed the divergence between word accuracy and understanding accuracy in [14]. They interpolated the word n-gram with n-grams containing phrases that were salient for the call-routing task, and observed that a slight word accuracy improvement resulted in a disproportionately substantial improvement in understanding accuracy.

Although the AT&T paper was published in 1998, many researchers in the speech recognition field we have talked to still believe that better word accuracy implies better understanding accuracy. Perhaps the divergence in their paper was less obvious. In this study, we use a language model that is directly optimized for spoken language understanding without interpolating with the word n-gram to retain good word accuracy. The divergence between the word accuracy and the understanding accuracy becomes more drastic: the impact on the word accuracy is very negative; while the overall understanding accuracy improves substantially. We hope that this result will induce more *recognition for understanding* researches in the speech community and more effective models that optimize the ultimate goal of accurate understanding.

Recently researchers from University of Avignon studied “conceptual decoding” for speech understanding [15]. They had the similar idea of encoding domain knowledge in a finite state language model. Although they didn’t compare their results with the n-gram language model, their finding also reveals that word error rate may not be a good indicator for language understanding accuracy: while the word error rate was as high as 38.7%, the sentence interpretation error was only 12%.

6. SUMMARY

The HMM/CFG models, originally trained to optimize spoken language understanding accuracy, have been used

as the language model for speech recognition. Thanks to the use of domain knowledge and grammar library, the models use much less training data than the trigram model; but they do require supervised information such as the labeling of the training data. Although the word error rate is much higher than a trigram model, the understanding accuracy is much better. This demonstrates that model training criteria that matches the optimization objective for understanding is as important as, if not more important than, the reduction of word error rate for speech understanding.

7. ACKNOWLEDGEMENTS

The authors would like to thank Julian Odell, Li Jiang, Mei-Yuh Hwang and the members of the Speech Technology Group for their helps in this work.

8. REFERENCES

- [1] S. Young, "Talking to Machines (Statistically Speaking)." In the Proceedings of ICSLP 2002, Denver, Colorado, 2002.
- [2] A. Stolcke and S. M. Omohundro, "Best-first Model Merging for Hidden Markov Model Induction." International Computer Science Institute, Berkeley, California TR-94-003, 1994 1994.
- [3] C.-C. Wong and H. Meng, "Improvements on a Semi-Automatic Grammar Induction Framework." In the Proceedings of ASRU 2001, Madonna di Campiglio, Italy, 2001.
- [4] S. Miller, R. Bobrow, R. Ingria, and R. Schwartz, "Hidden Understanding Models of Natural Language." In the Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics, New Mexico State University, 1994.
- [5] M. Gavalda, "Growing Semantic Grammars." Ph.D. Thesis. *Language Technology Institute*. Pittsburgh: Carnegie Mellon University, 2000.
- [6] P. Price, "Evaluation of Spoken Language System: the ATIS domain." In the Proceedings of DARPA Speech and Natural Language Workshop, Hidden Valley, PA, 1990.
- [7] Y.-Y. Wang and A. Acero, "Combination of CFG and N-gram Modeling in Semantic Grammar Learning." In the Proceedings of Eurospeech 2003, Geneva, Switzerland, 2003.
- [8] Y.-Y. Wang and A. Acero, "Concept Acquisition in Example-Based Grammar Authoring." In the Proceedings of ICASSP, Hong Kong, China, 2003.
- [9] Y.-Y. Wang, M. Mahajan, and X. Huang, "A Unified Context-Free Grammar and N-Gram Model For Spoken Language Processing." In the Proceedings of ICASSP, Istanbul, Turkey, 2000.
- [10] G. Riccardi, R. Pieraccini, and E. Bocchieri, "Stochastic Automata for Language Modeling." *Computer Speech and Language*, vol. 10, pp. 265-293, 1996.
- [11] Microsoft Corporation, "Speech SDK 5.1 for Windows® applications." <http://www.microsoft.com/speech>
- [12] S. Young, "The HTK hidden Markov model toolkit: design and philosophy." Department of Engineering, Cambridge University, Cambridge, UK TR.153, 1993.
- [13] F. Jelinek and E. L. Mercer, "Interpolated Estimation of Markov Source Parameters from Sparse Data," in *Pattern Recognition in Practice*, D. Gelsema and L. Kanal, Eds.: North-Holland, 1980.
- [14] G. Riccardi and A. L. Gorin, "Stochastic Language Models for Speech Recognition and Understanding." In the Proceedings of ICSLP, Sidney, Australia, 1998.
- [15] Y. Estève, C. Raymond, F. Bechet, and R. De Mori, "Conceptual Decoding for Spoken Dialog Systems." In the Proceedings of Eurospeech 2003, Geneva, Switzerland, 2003.