Automatic Induction of N-Gram Language Models from a Natural Language Grammar[1]

Stephanie Seneff, Chao Wang and Timothy J. Hazen

Spoken Language Systems Group
MIT Laboratory for Computer Science
Cambridge, Massachusetts, 02139, USA
{seneff,wangc,hazen}@sls.lcs.mit.edu

Abstract

This paper details our work in developing a technique which can automatically generate class *n*-gram language models from natural language (NL) grammars in dialogue systems. The procedure eliminates the need for double maintenance of the recognizer language model and NL grammar. The resulting language model adopts the standard class *n*-gram framework for computational efficiency. Moreover, both the *n*-gram classes and training sentences are enhanced with semantic/syntactic tags defined in the NL grammar, such that the trained language model preserves the distinctive statistics associated with different word senses. We have applied this approach in several different domains and languages, and have evaluated it on our most mature dialogue systems to assess its competitiveness with pre-existing n-gram language models. The speech recognition performances with the new language model are in fact the best we have achieved in both the JUPITER weather domain and the MERCURY flight reservation domain.

1. Introduction

The Spoken Language Systems group at MIT has been developing spoken conversational systems for well over a decade. In order to process and understand user queries, these systems require a domain-relevant statistical language model (LM) to support the initial recognizer search, as well as a natural language understanding (NLU) component to derive a meaning representation. Typically, the NLU system parses a word graph generated by the recognizer, and chooses the solution that produces the most plausible hypothesis, taking into account both linguistic and acoustic scores, as well as dialogue context.

We have long believed that the natural language (NL) grammar should play a strong role in the specification of the statistical language model used during speech recognition. Such a tight coupling should lead to higher performance, because the system will be more likely to understand the proposed hypotheses. Furthermore, eliminating the need for double maintenance of language models would ease the burden of system development. Perhaps most significantly, the NL grammar could be used to create a functional language model for the recognizer in the absence of any training data, at the critical initial phase of development of a new domain.

In the past, we have explored several options for integration. An early attempt [9] used the full parsing mechanism as the sole language model for the recognizer. While this was feasible for limited domains, the computational cost of parsing is too high for the real-time demands of conversational interaction in practical applications. Another possibility is to fully expand the context-free component of the NL grammar into a recursive transition network (RTN), which could then in theory be incorporated directly into the finite-state transducer (FST) based search space of the recognizer. This too proved to be impractical except for very small domains. A more promising approach was to select a subset of the categories in the NL grammar as classes in a class *n*-gram, and to expand those classes using RTN's, as exemplified by [4, 11]. While this approach gives a performance that compares favorable to that of a standard class *n*-gram, it too suffers computationally, in part because our FST technology for speech recognition has been optimized for performance on traditional class *n*-gram's. An alternative approach, used by the SPEECHBUILDER [2] utility, is to generate both an NL grammar and a class *n*-gram from a set of annotated sentences.

This paper details our work in developing a technique which satisfies all of the required constraints: computational efficiency, ease of maintenance, and high performance. It is similar to the approach in [4, 11], except that the RTN is precompiled into an automatically generated set of multi-word units, such that the resulting *n*-gram is very similar to a standard class *n*-gram. An important difference is that there is the opportunity to define multiple classes for a single word. For example, "to" can be encoded as an infinitive or a preposition, and "first" can be distinguished among several different meanings, as in "first class," "the first flight," and "on March first." Thus the distinctive statistics associated with different word senses can be preserved in the language model. Furthermore, this is achieved via a process that is transparent to the system developer. The developer need only specify which categories in the NL grammar should be considered to be *n*-gram classes; the NLU system can automatically tag words for word sense disambiguation and label the training sentences accordingly.

All our experiments are conducted using the SUMMIT speech recognition system [1], which utilizes FST's to represent its search space. In Section 2, we first describe a reorganization in SUMMIT's FST constraints, namely the introduction of a new component FST to model word reductions and contractions, that facilitated the work described in this paper. In Section 3, we describe in detail the automatic process of generating an enhanced class *n*-gram language model from the NL grammar. Section 4 provides some speech recognition experimental results. In Section 5, we conclude with a summary and point out some additional applications of this technology.

[1]This research is supported by the NSF under subaward number 1120330-133982 and by NTT.

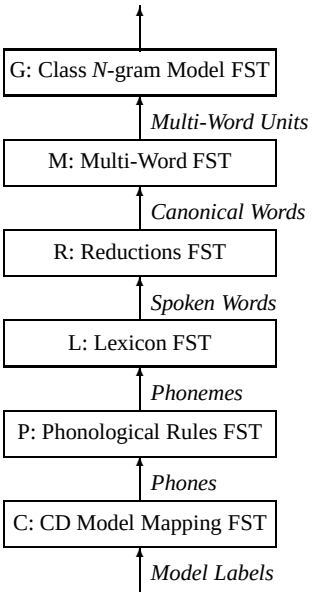

Figure 1: *The set of distinct FST components which are composed to form the full FST search network within the* SUMMIT *recognizer.*

2. SUMMIT

The SUMMIT recognizer utilizes a finite-state transducer representation for its phonological, lexical and language modeling components. The FST representation allows the various hierarchical components of the recognizer's search space to be represented within a single parsimonious network through the use of generic FST operations such as composition, determination and minimization [5]. The full search network used by SUMMIT is illustrated in Figure 1. The figure shows the six primary hierarchical components of the search space: the class *n*-gram language model (G), the multi-word unit constructor (M), the set of word-level rewrite rules for reductions and contractions (R), the lexical pronunciation dictionary (L), the phonological rules (P), and the context-dependent (CD) model mapping (C). Each of these components can be independently created and represented as an FST. By composing the FST's such that the output labels of the lower-level components become the inputs for the higher-level components, a single FST network is created which encodes the constraints of all six individual components. The full network can be represented mathematically with the following expression:

$$N = C \circ P \circ L \circ R \circ M \circ G$$

This hierarchical structure allows the language modeling components M and G to be buffered from issues of pronunciation variation caused by word reductions and contractions. For example, the reductions FST R will map the contraction "*what's*" to its canonical form of "*what is*", allowing the language model to share the statistics of these syntactically and semantically identical forms. Our earlier solution to this problem was to define a vocabulary item "*what_is*" in the lexicon, which could support both the contracted and non-contracted pronunciations. However, this solution relied on the language model to include "*what_is*" in its vocabulary as well, which, as verified by an experiment, hurt the recognition performance.

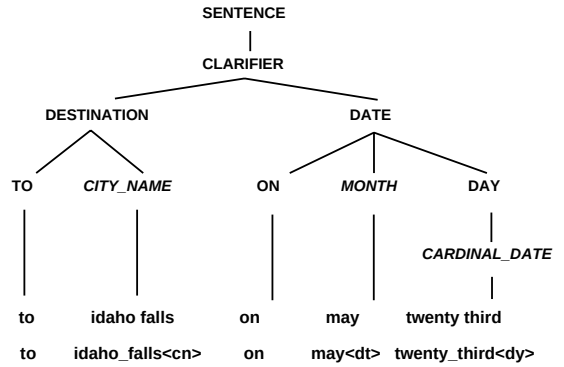

Figure 2: *Example of a parse tree for the sentence "to idaho falls on may twenty third." The grammar categories "*CITY_NAME*", "*MONTH*", and "*CARDINAL_DATE*" (italicized in the parse tree) are selected to be n-gram classes. The words under those selected categories form entries to the n-gram classes, and multiple words are "underbarred" to form multi-word units. Words in the input sentence are automatically tagged with the grammar category information for word sense disambiguation.*

3. Methodology

In this section, we describe the procedure for obtaining the class *n*-gram model using the NL component, which involves automatically creating M and G from a pre-existing context-free grammar and a set of training utterances.

3.1. The TINA Framework

The natural language understanding component of our dialogue systems makes use of the TINA framework [6], which utilizes a set of context-free rules to define allowable utterance patterns within each domain-dependent grammar, along with a feature unification mechanism to enforce agreement constraints and handle movement phenomena. The system supports a probability model which is acquired by tabulating frequency counts on a large corpus of parsed data. The model predicts each word by a product of trigrams progressing up the parse column; each trigram represents the conditional probability of a child given its parent and left sibling in the parse tree.

The context-free grammar used by TINA distinguishes between preterminal and non-preterminal categories. Expansions of the preterminal categories are restricted to vocabulary items, whereas other categories expand only into non-terminals, to specify sentence and phrase patterns. Many grammar categories form natural word classes, for example, a list of city names, months, weekdays, etc. A developer need only select a set of categories to serve as classes in the class *n*-gram. At present, the classes are restricted to be either preterminal or one layer above. We have enhanced the TINA framework to support the automatic creation of the appropriately formatted input files to the SUMMIT *n*-gram tools. The procedure expands the sub-grammar under each category to a class exhaustively to all supported word sequences, to establish the class assignments and the vocabulary of the multi-word units at the output of the multi-word mapping FST, M. TINA is also tasked with producing a corpus of *n*-gram training utterances, with the words appropriately tagged for their class, as determined by parsing.

Figure 2 illustrates how the procedure works. A corpus

of transcribed user utterances is parsed and then relabeled according to the designated n-gram classes, so that different word senses can be disambiguated according to the parse analysis.

3.2. Sentence Tagging through FST Transduction

The sentence tagging procedure outlined above works well for sentences that can be successfully parsed by the NLU system. However, there exist a significant number of training sentences that can not be parsed. Sentences may fail because they contain disfluencies, are incomplete, or are otherwise outside of the grammar's domain. Nevertheless, they are realistic representatives of utterances that the recognizer is expected to encounter. It is important that they be included in the language model training. Otherwise, the language model will be biased towards hypothesizing only well-formed sentences during recognition.

In order to be used for training, the words in the unparsed sentences need to be tagged for their class membership. This could be done by transducing them through the multi-word constructor FST, M, which defines mappings from canonical words to tagged multi-word units. However, such a simplistic solution is inadequate in dealing with ambiguities in word senses, for example, "one" as a numeral or a pronoun. We have devised a procedure that involves a single iteration, where we obtain an initial class *n*-gram model, G_1, from the parsable utterances only, and compose that with M to provide probability support for the disambiguation step. The procedure works as follows. First, the full training corpus is parsed and all parsable sentences are entered into an initial *n*-gram training corpus. An initial version of the language model G_1 is then obtained by processing the resulting tagged sentences through standard *n*-gram creation tools. This language model also includes backoff probabilities for an "<unknown>" word, to support sentences with out-of-vocabulary (OOV) words. The failed sentences are then transduced through $M \circ G_1$, with OOV words pre-mapped to the "<unknown>" tag. The corpus can then simply be augmented with these additional tagged sentences, and the final language model, G, is created from the complete corpus.

This framework can also be utilized to reduce the language model compilation time, which is almost entirely accounted for by the computation involved in parsing a large training corpus. This is especially important when delay becomes an issue, for example, in an interactive environment such as SPEECHBUILDER [2]. An interesting solution is to parse only a small subset of the corpus to bootstrap the mapping transducer, and use FST operations outlined above to label the remaining sentences, i.e., treating them as if they failed to parse. We have conducted an experiment in the JUPITER weather domain to assess the feasibility of this solution, and to evaluate the labeling performance of the FST transduction algorithm. If we parse only 10% of the data to train G_1, and label the rest by transducing the sentences through $M \circ G_1$, then more than 99% of the unparsed sentences are tagged correctly. In contrast, only 60% of the sentences would be tagged correctly if only M were used for the transduction. With this procedure, we reduce the parsing time by a factor of ten, from twenty minutes for the 120,000 training sentences down to just two minutes for 12,000.

3.3. Additional Features

Although one can simply equate the recognizer vocabulary to the NL grammar's vocabulary, there are situations in which one may want to introduce differences, both in terms of adding and removing words. First of all, there may be words in the grammar that are inappropriate for the recognizer. An obvious example is abbreviations, such as "Apr" for "April", or "St" for "street," which are included to cover typed input. We have provided the capability for the system developer to enumerate such words as excluded words.

A second issue is to account for non-speech events, such as laughter, coughing, and various fill words such as "um" and "uh", etc. Our SUMMIT recognizer includes explicit models for a small set of such "words". In fact, they have been carefully transcribed in the training corpus. Since it is difficult to develop grammar rules to cover them in all the situations where they could occur, it is tempting to simply remove them from the hypotheses, prior to parsing. However, they need to be preserved in place for *n*-gram training. Hence, we modified the parsing procedure to support the capability to skip over a selected set of words, as specified by the developer, during parsing, but to retain them in the output tagged sentences.

A third issue is the possibility of defining additional multi-word units, not because of a class membership, but because it may lead to improvements in language model perplexity. Obvious examples would be phrases like "good bye", and "thank you." We designed a data-driven procedure to identify promising candidates based on mutual information measures on a large training corpus [3]. These candidates are pruned based on recognition experiments on development data. Once a set is identified, they are simply entered as a list of extra words, and they will be automatically incorporated into M and G.

Finally, there may be situations where the recognition performance could benefit from the addition of words not supported by the grammar. A case where this idea is well-motivated can be demonstrated by the ORION task delegation system [8]. ORION allows a user to leave a recorded message for later delivery. Such a message is unrestricted in content, but must be embedded in a meaningful carrier phrase, such as "remind me to ...". The NL server pays no attention to the content following the carrier phrase, and therefore, does not necessarily cover those words in its vocabulary. A generic OOV model in the recognizer is able to get through the unrestricted message region; however, explicit support for commonly occurring words in the messages would likely lead to improvements. Such words can simply be added to the "extra words" list.

3.4. Summary of the LM Training Procedure

To summarize, the process of generating the language model from a natural language grammar works as follows:

1. Identify grammar categories appropriate for *n*-gram classes. Files containing extra or excluded words may also be provided.
2. Use TINA to generate formatted files specifying *n*-gram classes, a list of multi-word units, and mappings from multi-word units to canonical words. Also use TINA to parse the training corpus.
3. Use SUMMIT FST tools to generate M from the mapping file.
4. Use SUMMIT LM tools to train an initial class *n*-gram G_1 from the parsed and tagged sentences, which includes support for a catch-all tag "<unknown>" for all unknown words.
5. Map any OOV words in the failed sentences to the "<unknown>" marker. Tag the failed sentences by FST transduction through $M \circ G_1$.

	Baseline	TINA *n*-gram
JUPITER	18.3 %	18.0 %
MERCURY	15.6 %	15.0 %

Table 1: *Comparison of word error rates between a speech recognition system using standard class n-gram (baseline) and a system that uses enhanced class n-gram derived from the NL grammar (*TINA *n-gram), on the* JUPITER *weather domain and on the* MERCURY *flight reservation domain.*

6. Retrain a class *n*-gram G from all training sentences.

4. Evaluation Results

We have implemented the approach outlined above in several different domains and languages, and have evaluated it on our most mature systems to assess its competitiveness with pre-existing *n*-gram language models. The baseline system uses diphone models as well as context-independent segment duration models for acoustic modeling, and standard class *n*-gram models (with hand-crafted word classes) for language modeling. The "TINA *n*-gram" system uses the same configuration, except that the language models use vocabulary and word classes generated from the NL grammar, but are trained on the same set of sentences as used in training the baseline LM models.

Table 1 summarizes the recognition word error rates of the baseline system and the TINA *n*-gram system, on the JUPITER weather domain [12] and the MERCURY flight reservation domain [10]. The baseline JUPITER system has a vocabulary size of 2150 "spoken" words, and the baseline MERCURY system has 1725. The TINA *n*-gram version of the JUPITER recognizer is carefully crafted to match the vocabulary of the baseline system; i.e., given the FST specification of the recognizer search space as depicted in Figure 1, they differ only in the FST's above the canonical words output of R. As indicated in Table 1, the "TINA *n*-gram" system achieves a slightly better performance than the baseline. The comparison in the MERCURY domain is less rigorous – the development of the baseline recognizer and the NL component were out-of-sync, and we did not try to match the vocabularies of the two systems. The NL grammar of the MERCURY domain produced an expanded lexicon of 1850 "spoken" words. This may account for the greater performance gain for the MERCURY domain. The "TINA *n*-gram" performances are in fact the best we have achieved on both the JUPITER and MERCURY domains.

5. Summary and Future Work

In this paper, we have implemented a technique to automatically obtain enhanced class *n*-gram language models from NL grammars. We evaluated this approach in both the JUPITER weather domain and the MERCURY flight reservation domain, and confirmed that the recognition performance is competitive with, and, in fact, slightly better than, the baseline performance. We have migrated many of our dialogue systems, encompassing several different domains and languages, to make use of this new technique for creating recognizer language models. These include the ORION system in both English and Mandarin, a Mandarin weather domain, a city guide and traffic information system, and a flight status domain, in addition to JUPITER and MERCURY. Efforts to integrate this technique for language modeling into the SPEECHBUILDER toolkit [2] are underway. This will allow a developer to regenerate the recognizer's language model after manually adjusting the grammar.

This technique also facilitated our research towards enabling end users to personalize a dialogue system by adding new knowledge to the system during a conversation [7]. A user can speak and spell a new word to the system. The system will deduce the spelling and the pronunciation of the new words, and, after user verification, update the NL grammar and the recognizer lexicon and language model. The NLU system was enhanced to enable *incremental* update of the trained grammar, and the language model can be re-generated accordingly using the procedure described in this paper.

A generalization of this technique can be used to specialize a recognizer to a restricted subset of an NL grammar's domain. It is straightforward to restrict the recognizer vocabulary to just those words that showed up in a domain-restricted training corpus, generalized to all words in any grammar categories that were visited by the corpus. We plan to explore techniques to automatically create a suite of lesson-specific language models that would be used for distinct lesson plans for computer-aided language learning applications. In a language learning scenario, it is absolutely critical to restrict the search space as much as possible in the recognizer, in order to overcome the difficulties of recognition when the speaker is not fluent in the language.

6. References

[1] J. Glass, T. J. Hazen, and I. L. Hetherington, "Real-time telephone-based speech recognition in the JUPITER domain," *Proc. ICASSP*, Phoenix, March, 1999.

[2] J. Glass and E. Weinstein, "SPEECHBUILDER: Facilitating spoken dialogue systems development," *Proc. EUROSPEECH 2001*, pp. 1335–1338, Aalborg, Denmark, Sep. 2001.

[3] Michael K. McCandless and James R. Glass, "Empirical acquisition of word and phrase classes in the ATIS domain," *Proc. EUROSPEECH 1993*, Berlin, Germany, Sep. 1993.

[4] X. Mou, S. Seneff and V. Zue, "Integration of supra-lexical linguistic models with speech recognition using shallow parsing and finite state transducers," *Proc. ICSLP '02*, Vol. II, pp. 1289–1292, Denver, CO, Sep. 2002.

[5] F. Pereira and M. Riley, "Speech recognition by composition of weighted finite automata," in *Finite-State Language Processing* (E. Roche and Y. Schabes, eds.), pp. 431–453, Cambridge, MA, MIT Press, 1997.

[6] S. Seneff, "TINA: A natural language system for spoken language applications," *Computational Linguistics*, Vol. 18, No. 1, pp. 61–86, 1992.

[7] S. Seneff, G. Chung, and C. Wang, "Empowering end users to personalize dialogue systems through spoken interaction," submitted to *EUROSPEECH 2003*.

[8] S. Seneff, C. Chuu, and D. S. Cyphers, "ORION: From on-line interaction to off-line delegation," *Proc. ICSLP '00*, Beijing China, Oct. 2000.

[9] S. Seneff, M. McCandless, and V. Zue, "Integrating natural language into the word graph search for simultaneous speech recognition and understanding," *Proc. EUROSPEECH 1995*, pp. 1781-1784, Sep. 1995.

[10] S. Seneff, "Response Planning and Generation in the MERCURY Flight Reservation System," *Computer Speech and Language*, Vol. 16, pp. 283–312, 2002.

[11] C. Wang, D. S. Cyphers, X. Mou, J. Polifroni, S. Seneff, J. Yi, and V. Zue, "MUXING: A telephone-access Mandarin conversational system," *Proc. ICSLP '00*, Vol. II, pp. 715–718, Beijing, China, Oct. 2000.

[12] V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T. J. Hazen, and L. Hetherington, "JUPITER: A Telephone-Based Conversational Interface for Weather Information," *IEEE Transactions on Speech and Audio Processing*, Vol. 8, No. 1, pp. 85-96, January, 2000.