Unified Language Modeling using Finite-State Transducers with First Applications

J.G.A. Dolfing, P.G. Buckley, D. Horowitz

Vox Generation Ltd.
Manor House, 21 Soho Square, London W1D 3QP, UK
hdolfing@voxgeneration.com

Abstract

In this paper, we investigate a weighted finite-state transducer approach to language modelling for speech recognition applications. We explore a unified framework to conversational speech recognition which combines the benefits of grammars, n-gram and class-based language models, with the flexibility of using dynamic data, and the potential for integrating semantics. Based on a virtual personal assistant application, we show first applications and recognition results of out-of-grammar handling and the integration of class-based, weighted, dynamic data into this framework.

1. Introduction

In the context of the FASiL project, we investigate language modelling in order to enable a conversational speech interface to a virtual personal assistant (VPA) application [1]. The language models consist of an elegant combination of statistical n-gram language models and grammars, structured in a hierarchy of components and potentially combined with semantics [2], that form a unified hierarchical language model. In this paper, we report on first experiments with a part of the unified model realized with finite-state transducers (FST).

The approach combines several strands of language modelling theory and research into one elegant model. A number of researchers have worked on methodologies to combine the semantic processing and natural language understanding with standard n-gram statistical language modelling, starting with [3], and successively with [4, 5], and more recently [6]. Other approaches have been made to exploit prior knowledge about language structure by building a hierarchical model such as [7, 8]. An overview is given in [9].

The approach based on FSTs provides a solid, known basis for the “pure” language modeling \( P(W) \) and also allows us to explore ways to integrate the structured semantics, or parts of it, in the same, unified FST framework ultimately solving not only \( P(W|X) \) but also \( P(S|X) \) where \( S \) denotes a structured semantic interpretation. Although we have begun to integrate a probabilistic semantic model into the FSTs, this paper concentrates on the first step, a structured hierarchical language model which in itself is not new, where the FSTs are functionally equivalent to language models or probabilistic networks, implemented with finite-state transducers, to solve the Bayesian decision problem for \( P(W|X) \).

Various combinations of n-gram language models, grammars, and parsers have been explored in literature. Mostly, these combinations of language models and grammars were costly in terms of software code and development. Some of the grammar and n-gram combinations are even possible with commercial software, e.g., the new extended version of the W3C/SRGS standard for speech recognition grammars which allows n-grams but does not specify a generic format. Therefore, the FST approach in this paper allows us to use much more sophisticated models compared to the plain SRGS level.

First, our goal is to improve robustness and accuracy in the recognition based on a hybrid of language models and grammars. There are two ways to look at the union of grammars and language models. We can use the grammars as principal recognition network and use the language model as a generic “filler” model to catch out-of-grammar (OOG) utterances. Alternatively, we can do conversational speech recognition based on a generic n-gram augmented with a very specialized, small grammar to catch a set of vital utterances.

Second, we build up a hierarchical system which is similar to a class-based language model with the principal difference being that the classes change their content on-the-fly. Similar approaches are investigated by [10, 11], a divide and conquer approach for one-stage recognition with large FST-based language models, and building on that and putting the “late-binding” in a commercial framework in [12], and [7] for computation of probabilities in a more structured way than plain n-grams.

In Section 2, we briefly discuss the general properties of a finite-state transducer-based recognition system and show how this relates to several ways of structuring the language models. We concentrate on parallel “union” and hierarchical networks, and apply them to OOG handling and a conversational speech task employing dynamically changing name lists in Section 4. Section 3 explains the experimental setup and we summarize our findings in Section 5.

This research is supported by the EU Grant FASiL IST 2001-38685, www.fasil.co.uk.
2. Finite-state transducer framework

Recent papers [10, 13] show that finite-state transducers are an attractive alternative for speech decoding. One of the main problems in speech decoding is to minimize recognition errors. Because multi-pass recognizers [14] are potentially more vulnerable for decoding errors, since they require tuning in every pass, one-pass recognizers are attractive [15]. A one-pass recognizer can employ all available knowledge sources at the same time which results in the best possible recognition result, at the price of a more complex decoding process.

Traditionally, recognition and semantic understanding are processed in two passes. However, if it was possible to combine some semantic information, then we could potentially decrease the number of decoding errors. In its simplest form, this is demonstrated by resoring an n-best list, based on semantic equivalence, especially for directed dialogue applications.

Typically, within a finite-state transducer-based recognition system the various constraints such as language model, lexicon, phonological rules, context-dependency, HMM topology, etc., are each represented as a possibly weighted transducer, and these transducers are composed together to form the single transducer to be used for recognition [13].

Although the recognizer typically uses the weighted finite-state transducer $CPLG = \text{opt}(C \circ P \circ L \circ G)$ for recognition, where $\text{opt}()$ is a combined pushing, determinization, and minimization operation, $G$ is an n-gram language model, $L$ the lexicon, $P$ is a set of phonological rules and $C$ adds context-dependent phonetic models (usually diphones), we work mostly on the level of $G$ to design various types of structured language models.

Previously, most work on the language model $G$ has been done with the objective of better parameter estimation which includes interpolation, class-based models, distance bigrams, and more. In addition, work has been done on exploiting hierarchical structures in language models [7], unifying language models with parsers [6], and closer integration of language models with the speech decoder [10, 15].

Here, we describe a first application of a unified framework to a virtual personal assistant application. The basic decoder software remains unchanged which is a big advantage compared to the approach of [6]. The proposed finite-state transducer framework also handles class-based models where the classes are either static or have dynamically changing content. The dynamic data is included via on-the-fly composition as discussed in [10–12].

3. Experimental setup

For the experiments to follow, we study a virtual personal assistant (VPA) application. The VPA is based on a system developed by Vox Generation and extended to multi-modality and conversational speech in the context of FASIL.

During the course of the project, we have collected a variety of corpora for testing the system and doing research. We have several speech corpora specialized to the human-to-machine application, corpora with human-to-human dialogues in three languages, and additional text corpora used for language modeling.

We use two different corpora for our experiments, Task1 with directed dialogue data and Task2 with conversational speech. First, Task1 contains a set of directed dialogue commands for the VPA and about 30% of the utterances are out of grammar. The utterances are relatively simple. Second, the Task2 corpus contains conversational speech on the task of sending emails in a variety of ways to a varying set of people in one utterance. Audio is recorded via a variety of telephone lines, both landlines and mobiles. A part of the recordings is relative high noise recorded on the street.

Both corpora are defined in separate test and training/development sets. The corpus Task1 contains 1480 utterances, which accumulates to 5346 words and about one hour of speech data. The corpus is split 50-50 into development and test set and recorded over a variety of telephone lines. The Task2 corpus with conversational speech consists of 1.5 hours of speech data with 9534 words split 70-30 into development and test sets.

The human-to-human conversational speech corpus is collected in the three languages English, Swedish and Portuguese based on a wizard-of-oz study. This corpus will be released to the speech community after the completion of the project. The textual transcriptions were used and interrelated with a variety of news data and data generated from grammars to provide domain-specific text material for training the n-gram model. The English textual data amounts to about 3 million words of domain-specific data for training. A precise corpus definition will be reported in another paper.

The employed recognizer is the SpeechWorks/ScanSoft OpenSpeech recognizer OSR 2.0, which is based on finite-state transducers, and available in several languages. We use English US acoustic models and a VPA specific vocabulary of about 1000 words including compound words. In the course of our experiments, we precompile various types of finite-state language models in the form of weighted finite-state transducer networks, and direct the recognizer to use these networks via low-level interfaces. The plain n-gram models are compiled into finite-state transducers with the “sgc” grammar compiler.

4. Results

Based on the finite-state transducer algebra, we can build parallel, hierarchical or sequential language models with dynamically changing content. The parallel models like $G = G_1 \cup G_2$ are used to put grammars and n-grams in parallel. The parallel models can also be used to run an n-gram recognition with a filler-phoneme network in parallel. The hierarchical model is built as $G = G_1 \circ G_2$ where the composition is either standard, static composition or an on-the-fly composition. This can be used to modify language model weights as in [10] or to include weighted dynamic data like name lists. Finally, we can build sequential
models like \( G = G_1, G_2 \) using the concatenation operator. This can be used, e.g., to put a small grammar in front of a language model.

In this section, we study two instantiations of the above models. First, we study the behavior of a parallel model where we construct \( n \)-grams with a specialized grammar in parallel \( G_{\text{union}} = G_{\text{grammar}} \cup G_{n-\text{gram}} \) and study the prior weighting of these models. The parallel model uses only static data. Second, we study a more complex, hierarchical model \( G = G_{n-\text{gram}} \circ (G_1 \cup G_2) \) using on-the-fly composition.

### 4.1. Union-type recognition/OOG

Based on the directed dialogue corpus Task1, we do a first experiment with a combination of grammars and language models. This will be a weighted “union” of grammar and language model. The goal of the experiment is to show that the “union” provides a better recognition result with many OOG utterances. Experiments like this are not new which is why support for such “filler” language models is even part of standards like GRXML/SRGS. What we show here is a first experiment in the unified framework. Due to the many OOG utterances, we expect that the “union” combination will have a sweet-spot and show a word error-rate better than the grammar-only solution. This approach is more useful for directed dialogue interactions for which we can write grammars. In the case of conversational speech, the “union” can only be used to ensure improved recognition of certain important commands.

In this experiment, we interpolate \( P(W) = \lambda P_{\text{grammar}}(W) + (1 - \lambda) P_{n-\text{gram}}(W) \). For the simplified experiment below, we just add the prior probability as scaled loglikelihood \(-\alpha \log \lambda \) to the grammar, shown on the \( x \)-axis, with \( \alpha \) being the language model scaling factor of about 70.

![Interpolation of grammars with bigram (2G) and trigram (3G) language models. The baseline grammar performance is on the extreme left-hand side, while the baseline language model performance is on the right-hand side.](image)

![Figure 1: Interpolation of grammars with bigram (2G) and trigram (3G) language models. The baseline grammar performance is on the extreme left-hand side, while the baseline language model performance is on the right-hand side.](image)

The baseline error-rates are relatively high for an application of this complexity, most likely because of a mixture of the quality of the telephone recordings, the lack of large amounts of domain-specific data for the language models, and the OOG data. The experiment shows that both the interpolations with bigram and trigram show a curve with a minimum error-rate. The best error-rate is much better than the grammar-only rate of course, and the minimum error-rate is better than the language model only error-rate. The better the language model, in this case the trigram, the smaller the difference between minimum and language model performance. It is important to note that the minima do not occur at weighting 0 but at 250 and 500, respectively. The position of the minima depends on the amount of OOG data and the specific \( n \)-gram model.

### 4.2. Hierarchical recognition

In this section, we show that the finite-state transducer framework enables a hierarchical model which enables us to use classes with dynamic content and \( n \)-gram weights. To start with, we study the model \( G = G_{n-\text{gram}} \circ (G_1 \cup G_2 \cup G_3) \) using on-the-fly composition where \( G_1 \) is a weighted name list with up to 10,000 names and \( G_2 \) and \( G_3 \) are fixed, static grammars.

Strictly speaking, the composition should be \( G = G_{n-\text{gram}} \circ (G_{\text{filler}} \cup G_1 \cup G_2 \cup G_3) \) where \( G_{\text{filler}} \) is a one-state transducer with the full vocabulary on input and output labels. That is necessary to make the composition work. However, in the OSR2 recognizer framework, that is done automatically. Hence, we denote \( G = G_{n-\text{gram}} \circ (G_1 \cup G_2 \cup G_3) \).

The hierarchical framework based on [12] gives us a flexible basis for application specific tuning. First, we can weight the included grammars with anything from zero-gram to a full \( n \)-gram. Second, due to the dynamic content of the included \( G_n \), we can change the recognizer behavior with every run and tune it based on historical data. For example, we can easily change the weights on a day-to-day basis, apply different smoothing techniques depending on name list properties, and even change the weights depending on other prior knowledge about the application, e.g., key phrases and names from news channels.

As an example, we used corpus Task2 and used a name list with varying size. Starting with a minimal size of typically about 40, which covers all the names in the corpus, to a list with 10,000 names, in a zero-gram and unigram scenario. This is actually testing a part of the VPA scenario.

As expected, the error-rates go up when the name list gets larger. In the unweighted scenario, the system becomes unusable quickly. In the weighted, unigram scenario, the system behaves much better. The experiment in itself is not new. What is new is that this is a unified framework which allows us to quickly change the weighting scheme, e.g., based on time, date, position, etc, to tune the VPA for better performance. The finite-state transducer approach makes the system very flexible and extensible without huge amounts of new code.

### 5. Conclusion

We have shown that finite-state transducers are a very flexible framework for unifying many types of language models. In future, we will explore how to integrate the semantics. Based on the framework, we showed how to build a
recognition system which is more robust for OOG utterances and a hierarchical system, similar to a class-based language model, but with dynamically changing content and weights for the classes. In contrast to other unification approaches, we use only a single, commercially available decoder based on finite-state transducers.

In addition to the current VPA which implements semantics as a GRXML/SRGS page “on top of” the FST model, we are working on an extension to include structured semantics into the transducer $PCLG \otimes S$. 1

6. References


1The authors would like to thank Roberto Pieraccini and Sasha Caskey from SpeechWorks for their support with the recognizers and useful discussions, as well as Kerry Robinson and Partha Lal from Vox Generation.

Figure 2: A comparison of zero (3G-0G) and unigram (3G-1G) weighted dynamic name lists included as part of a hierarchical trigram model. The data consists of utterances with sequences of names embedded in conversational speech.