Dynamic WFST-based Decoder for Automatic Speech Recognition

Student: Danilo de Oliveira Ribeiro e Silva
Company Supervisor: Marc Ferras-Font (SONY)

Automatic Speech Recognition (ASR) systems convert speech from a recorded audio signal to text. Such systems aim to infer the original words given an observable signal, most commonly following a probabilistic approach. We call decoding the process of calculating which sequence of words is most likely to match the acoustic signal [1].

One extensively used way of searching the best word sequence involves the use of Weighted Finite State Transducers (WFST), static graphs encoding the whole word-sequence search space. They contain many levels of information, and define what is allowed (and more probable) in the language. During decoding, the decoder examines them in order to predict the best sequence(s) - the most probable / less costly path(s).

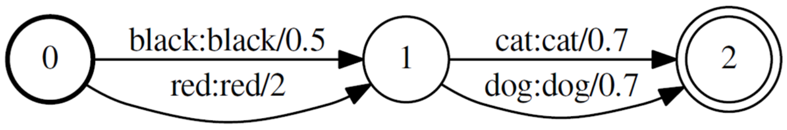

Figure 1: Example grammar WFST. Transitions contain input and output labels and a probability (or a cost) of making such transition. A path from a start to an end state *transduces* one input sequence of labels to an output one

The internship consisted in the investigation and implementation of ways to allow dynamic changes to the decoding graphs and/or decoding process of a WFST-based decoder. The main goals were to create class-based decoders, as well as the use of dynamic approaches for the construction of cascades of WFSTs.

We managed to implement functional decoders that are class based; numbers were kept in a separate graph, and this graph was replaced in a main graph whenever a quantity was expected. This makes it easier for modifications, since OOVs can be more easily inserted into their respective class graphs due to the reduced size of the latter. This technique can be extended to other classes, such as names of people and cities, which are often unknown to the system.

The WFST examined during decoding is a combination (composition) of different graphs with multiple levels of information. These graphs are usually combined before the decoding operation, but this increases significantly the WFST size. We managed to implement this combination during decoding, so the search space is constructed on-the-fly. This significantly reduces memory usage both in graph on-disk storage and RAM resources during decoding.

These two features were combined in a sole decoder that includes on-the-fly class replacement and on-the-fly composition. Table 1 presents the performance of the different decoders that were implemented over the course of this internship. Word Error Rate (WER) expresses the error of the predictions (less is better) and Real Time Factor (RTF) shows the speed (less is better, 1.0 means real time. *Static* is the original case; *Static Replace* adds the replacement of class graphs in the main transducer; *Lazy* implements the on-the-fly combination of the different levels of representation; and *Lazy Replace* uses both techniques in the same decoder.

Granted, the performance of the dynamic decoder is worse than the original non-lazy, non-replaced graph, both in terms of accuracy and speed. Nevertheless, this impact on performance was expected; the original graph, though impractical in terms of size, is further optimized, since it's a single piece. Our implemented graph can also be further optimized so as to get as close a performance as possible to the original, ordinary graph.

Type	WER	RTF
Static	10.0%	0.37
Static Replace	11.2%	0.39
Lazy	13.4%	1.28
Lazy Replace	14.4%	1.48

Table 1: Scores obtained for lazy composition experiments.

References

[1] R.E. Gruhn, W. Minker, and S. Nakamura. *Statistical Pronunciation Modeling for Non-Native Speech Processing.* Signals and Communication Technology. Springer Berlin Heidelberg, 2011.