Weighted Finite State Transducer Components Construction

Xu Xiang
E-mail: chinoiserie811@gmail.com

Abstract

The weighted finite state transducer (WFST) framework provides a general representation for components of speech recognition systems, including acoustic model, context-dependency model, pronounciation model and statistical language model. We investigate several approaches and schemes of constructing WFST components used in IDIAP's Juicer WFST decoder and Josef Novak's Transducersaurus WFST tool sets. We illustrate the model to WFST construction progress to address some more general problems, and attempt to achieve some reasonable results.

Contents

1 Introduction **1**

2 Statistical language model to WFST **1**
2.1 $n-$gram Language Model Format 1
2.2 Auxiliary Labels on Backoff $\epsilon-$Transitions 3
2.3 $n-$gram Language Model Normalisation 4

3 Pronunciation model to WFST **4**
3.1 Basic Representation 4
3.2 Auxiliary Labels 5
3.3 Optimization Issues 5

4 Context-dependency model to WFST **6**
4.1 Sub-word Units 6
4.2 Basic Representation 6
4.3 Silence Modeling 8
4.3.1 Silence Modeling in Juicer 8
4.3.2 Silence Modeling in Transducersaurus 8
4.4 Auxiliary Symbols 9

5 Conclusion **9**

List of Figures

1 A bigram language model constructed by OpenGrm (ngrammake) 2
2 A bigram language model constructed by Juicer (gramgen) 3
3 Compose L* and G using matching filter, without auxiliary labels 3
4 Compose L* and G using sequencing filter, without auxiliary labels 3
5 Compose L* and G, with auxiliary labels 4
6 A simple L WFST 5
7 A transformed L* WFST 5
8 A prototype C WFST 7
9 A simple cross-word type C WFST 7
10 Insertion of `sp` into transitions in C 8
11 An example of L WFST with silence phones 8
12 A simple silence transducer T 9
13 Auxiliary loops on states of C WFST 9

List of Tables

1 Standard composition of $\tilde{\mathrm{L}}$ and $\tilde{\mathrm{G}}$ 6
2 Static lookahead composition of $\tilde{\mathrm{C}}\tilde{\mathrm{L}}$ and $\tilde{\mathrm{G}}$ 6
3 Context dependency model to WFST construction rules 7

1 Introduction

WFST [7, 6] has been widely used in the area of automatic speech recognition (ASR). In ASR system designs, WFST is used to represent models at different stages such as acoustic model (denoted by H), context-dependency model (denoted by C), pronunciation model (denoted by L), statistical language model (denoted by G), etc. There are several rules and conventions to construct WFST from these models, however, it is sometimes tricky to tune the construction progress to make the resulting WFST more efficient and robust.

This technical report examines both Juicer and Transducersaurus, and compares their WFST construction approaches.

This technical report is arranged as follows. Section 2 describes the statistical language model to WFST construction and discusses about normalisation and auxiliary symbols. Section 3 describes the pronunciation model to WFST construction and covers the topic of optimization. Section 4 illustrates a simple example of context-dependency model to WFST construction. Conclusions are drawn in section 5.

2 Statistical language model to WFST

2.1 $n-$gram Language Model Format

The word level grammar G, often a stochastic $n-$gram, can be represented by WFST compactly [2]. The basic idea of constructing WFST from $n-$gram, can be described by the following rules (OpenGrm project NGram Model Format by Michael Riley *et al.*).

An n-gram is a sequence of k symbols: $w_1 \cdots w_k$. Let $\mathbb{N}$ be the set of n-grams in the model.

- There is a *unigram* state in every model, representing the empty string.
- Every proper prefix of every n-gram in $\mathbb{N}$ has an associated state in the model.
- The state associated with an n-gram $w_1 \cdots w_k$ has a backoff transition (labeled with $\langle epsilon \rangle$) to the state associated with its suffix $w_2 \cdots w_k$.
- An n-gram $w_1 \cdots w_k$ is represented as a transition, labeled with w_k, from the state associated with its prefix $w_1 \cdots w_{k-1}$ to a destination state defined as follows:
  - If $w_1 \cdots w_k$ is a proper prefix of an $n-$gram in the model, then the destination of the transition is the state associated with $w_1 \cdots w_k$.
  - Otherwise, the destination of the transition is the state associated with the suffix $w_2 \cdots w_k$.
- Start and end of the sequence are not represented via transitions in the automaton or symbols in the symbol table. Rather
  - The start state of the automaton encodes the "start of sequence" $n-$gram prefix (commonly denoted $\langle s \rangle$).
  - The end of the sequence (often denoted $\langle /s \rangle$) is included in the model through state final weights, i.e., for a state associated with an $n-$gram prefix $w_1 \cdots w_k$, the final weight of that state represents the weight of the $n-$gram $w_1 \cdots w_k \langle /s \rangle$.

There is a small but complete bigram language model in ARPA format. Notice that the probabilities are base 10 logarithms.

```
\data\
ngram 1=4
ngram 2=6

\1-grams:
-99 <s> -0.39794
-0.69897 </s>
-0.39794 foo -0.60206
-0.39794 bar -0.60206

\2-grams:
-0.251812 <s> foo
-0.4436975 <s> bar
-0.6478175 foo foo
-0.139662 foo bar
-0.3716111 bar </s>
-0.3233064 bar foo

\end\
```

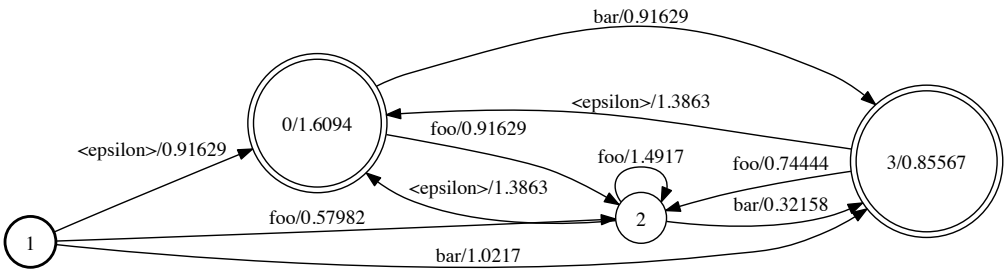


Figure 1: A bigram language model constructed by OpenGrm (ngrammake)

The model format used by Juicer and Transducersaurus is almost the same as OpenGrm NGram Format, except for

- There is a additional "start" state which only has one transition (labeled with $\langle s \rangle$) to the state associated with the "start of sequence" $n-$gram prefix (commonly denoted $\langle s \rangle$)
- There is a additional "end" state (commonly denoted $\langle /s \rangle$). The end of the sequence (denoted $\langle /s \rangle$) is included in the model through a transition (labeled with $\langle /s \rangle$) to the "end" state. The weight of the transition represents the weight of the $n-$gram $w_1 \cdots w_k \langle /s \rangle$. Thus, all successful paths are ended at this unique "end" state.

Figure 2 depicts the G WFST constrcuted by gramgen.

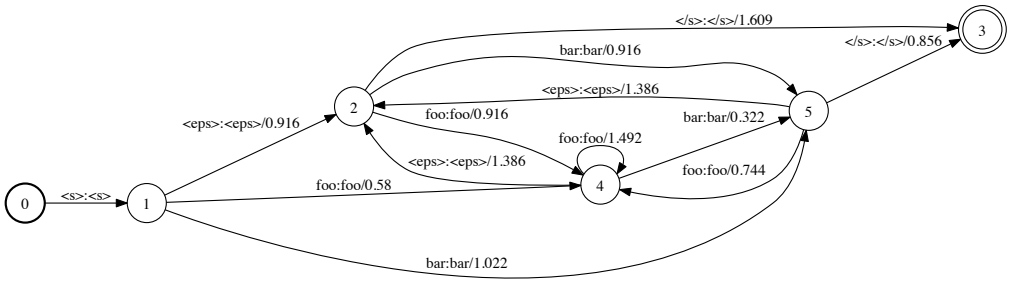


Figure 2: A bigram language model constructed by Juicer (gramgen)

2.2 Auxiliary Labels on Backoff $\epsilon-$Transitions

In the presence of backoff $\epsilon-$transitions, G may have several paths for a given word sequence. In this case, we can disambiguate the model by replacing the input $\epsilon-$label on backoff transitions with the auxiliary label (usually denoted by $\phi-$label). Respectively, We need to apply auxiliary symbols to L properly. By introducing auxiliary labels to G and L, we can ensure the composite WFST of L* (Kleene closure) and G is deterministic.

If we compose L* and G with matching filter without auxiliary symbols, we may get a non-deterministic and larger result as depicted in Figure 3. An easier solution is using sequencing filter instead. Figure 4 shows the deterministic resulting WFST using sequencing filter, which can be optimized further.

Figure 5 shows that the same resulting WFST of composing L* and G by applying auxiliary symbols.

To reduce the size and facilitate the optimization of the composite WFST, it is encouraged to implement auxiliary symbols on L and G.

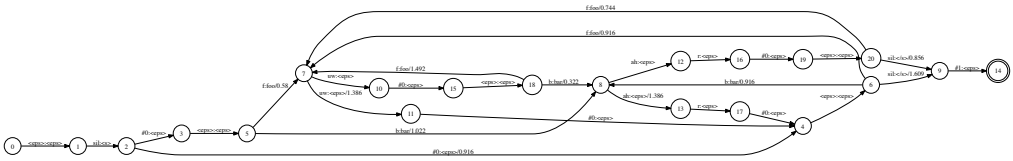


Figure 3: Compose L* and G using matching filter, without auxiliary labels

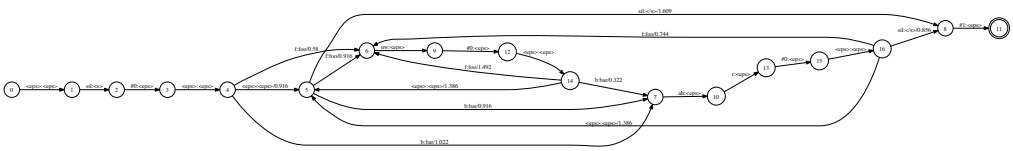


Figure 4: Compose L* and G using sequencing filter, without auxiliary labels

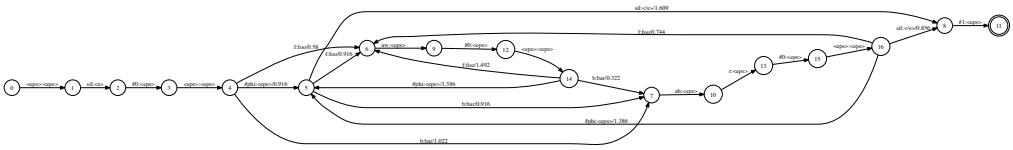


Figure 5: Compose L* and G, with auxiliary labels

2.3 $n-$gram Language Model Normalisation

In practice, $n-$gram language models are trained on very large corpus. An unmodified language model is usually too large for practical use, so there are several pruning techniques to produce a smaller language model. However, some probability (seen $n-$grams) may be "disgarded" after pruning. To keep the stochasticity, the backoff weights need to be re-estimated.

For example, the equation for Katz's backoff model [5] is

$$\hat{P}(w_i|w_{i-n+1}\cdots w_{i-1}) = \begin{cases} d_{w_{i-n+1}\cdots w_i}\dfrac{C(w_{i-n+1}\cdots w_i)}{C(w_{i-n+1}\cdots w_{i-1})} \text{ if } C(w_{i-n+1}\cdots w_i) > k, \\ \alpha_{w_{i-n+1}\cdots w_{i-1}}\hat{P}(w_i|w_{i-n+2}\cdots w_{i-1}) \text{ otherwise.} \end{cases}$$

where,

$C(x)$ is the number of times x appears in training.

w_i is the ith word in the given context.

d is typically the amount of discounting found by Good-Turing estimation.

Thus the backoff weight α is computed as

$$\alpha_{w_{i-n+1}\cdots w_{i-1}} = \frac{1 - \sum_{\{w_i:C(w_{i-n+1}\cdots w_i)>k\}} d_{w_{i-n+1}\cdots w_i}\frac{C(w_{i-n+1}\cdots w_i)}{C(w_{i-n+1}\cdots w_{i-1})}}{\sum_{\{w_i:C(w_{i-n+1}\cdots w_i)\leq k\}} \hat{P}(w_i|w_{i-n+2}\cdots w_{i-1})}$$

3 Pronunciation model to WFST

3.1 Basic Representation

From the simple dictionary below, we can construct an L WFST. Figure 6 depicts the L WFST.

```
<s> sil
</s> sil
bar b ah r
foo f uw
```

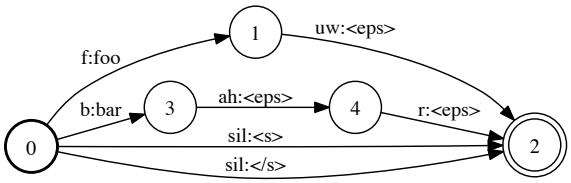


Figure 6: A simple L WFST

3.2 Auxiliary Labels

In general, L is not determinizable in the presence of homophony. To make L determinizable, we need to introduce auxiliary symbols to L. For example

`r eh d #0` read
`r eh d #1` red

Let P denotes the maximum degree of the homophony. Then we add P different auxiliary phones.

We need to add a $\phi-$loop at the start state of L. This $\phi-$loop is both to match the $\phi-$labels in G WFST, and to propagate them to the composite WFST of L^* and G. We also need to add other auxiliary labels (one of #0, #1, ... #P-1) to the end of the phonetic transcription of each word to distinguish homophones. In practice we use the transformed L^* to compose.

Figure 7 depicts the transformed L^* WFST.

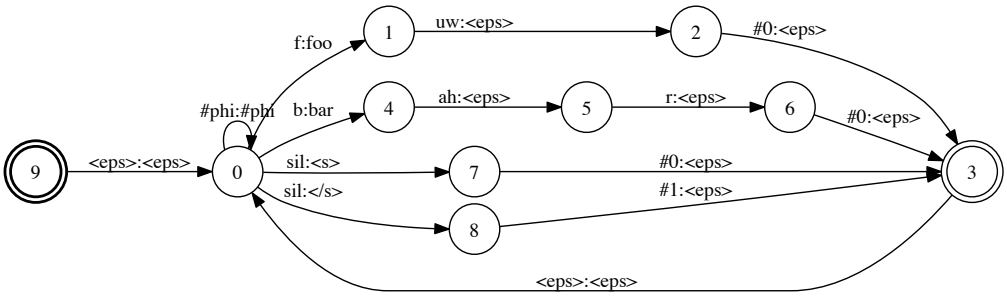


Figure 7: A transformed L^* WFST

3.3 Optimization Issues

We performed experiments on a 4 core Intel Core i3-3220 based machine running at 3.30GHz with 3MB cache and 8GBs of main system memory running GNU/Linux. We evaluated the composing progress using the standard WSJ 5k close NVP bigram language model and the WSJ 5k non-verbalized vocabulary.

To compose L^* and G in a standard way, it's preferable not to optimize the L^* WFST, since it's optimized to leave the output label in place. Table 1 shows the differences of two configurations that optimizes $\tilde{L}$ or not. The $\circ$ denotes to the standard composition.

Composition Configuration	Time	Memory Usage	WFST Size
$\tilde{L} \circ \tilde{G}$	0.542s	47MB	20MB
$\det(\tilde{L}) \circ \tilde{G}$	31.262s	3.7GB	891MB

Table 1: Standard composition of $\tilde{L}$ and $\tilde{G}$

To compose CL WFST and G WFST with static lookahead approach, C and L^* are composed first to produce CL WFST. In this case, it's preferable to determinize L^* first. Table 2 shows the differences of two configurations that optimizes $\tilde{L}$ or not. The $\circ$ denotes to the standard composition, and the . denotes to the lookahead composition.

Composition Configuration	Time	Memory Usage	WFST Size
$\tilde{C} \circ \tilde{L}$	1.601s	121MB	68MB
$(\tilde{C} \circ \tilde{L}).\tilde{G}$	4.163s	315MB	109MB
$\tilde{C} \circ det(\tilde{L})$	0.074s	-MB	1.4MB
$(\tilde{C} \circ det(\tilde{L})).\tilde{G}$	3.321s	183MB	38MB

Table 2: Static lookahead composition of $\tilde{C}\tilde{L}$ and $\tilde{G}$

4 Context-dependency model to WFST

4.1 Sub-word Units

The context dependency model maps sequences of context dependency phones (the sub-word units presented in a GMM based acoustic model) to sequences of monophones (the sub-word units used in a pronunciation model).

There are two variants of context-dependency models using different strategies to cope with *word boundary* information. One is word-internal model, and the other is cross-word model. The example below shows the differences of these two methods.

For simplicity, we will focus on the cross-word context dependency model in subsequent discussions.

Word-internal context dependency:
Word boundaries represent a distinct context.

```
speech task =
/sil s+p s-p+iy p-iy+ch iy-ch
t+ae t-ae+s ae-s+k s-k sil/
```

Cross-word context dependency:
Word boundaries are ignored or used as additional context information.

```
speech task =
/sil sil-s+p s-p+iy p-iy+ch iy-ch+t
ch-t+ae t-ae+s ae-s+k s-k+sil sil/
```

4.2 Basic Representation

Basically, the C WFST is constructed by the rules [3] stated in Table 3.

Type	Source	Destination	Input Symbol	Output Symbol
l-c+r	l,c	c,r	l-c+r	r
c+r	-,c	c,r	c+r	r
l-c	l,c	c,-(Final)	l-c	-
c	-,-(Start)	-,c	-	c
c	-,c	c,-(Final)	c	-

Table 3: Context dependency model to WFST construction rules

Figure 8 shows a prototype C WFST, the context independent phones are x and y only.

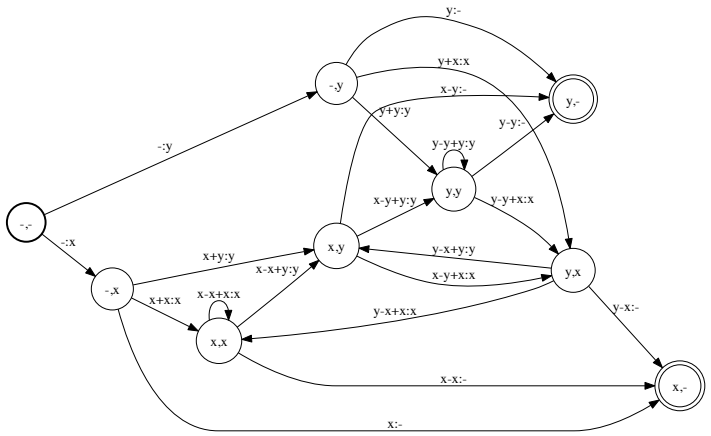


Figure 8: A prototype C WFST

There is a very small emparical C WFST constructed by Juicer (cdgen) depicted in Figure 9. Notice that we assume that all logical triphones are physical triphones. We use only one silence model "sil" and the auxiliary symbols are not illustrated. All triphones in tiedlist are in set $\{sil\} \cup \{f, uw, sil\} \times \{f, uw\} \times \{f, uw, sil\}$.

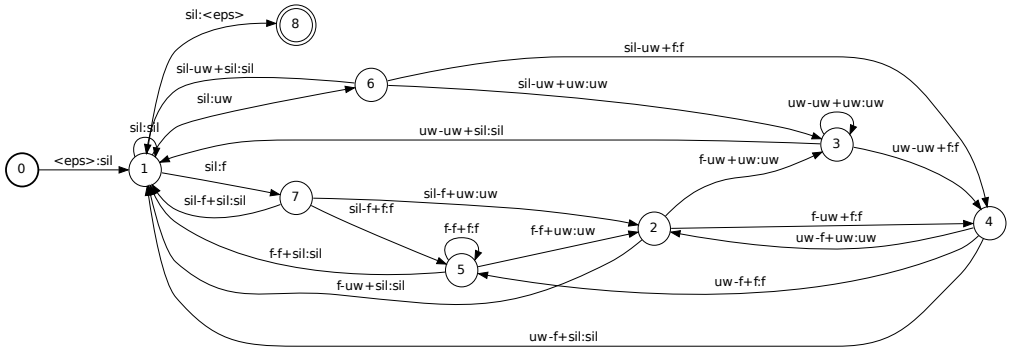


Figure 9: A simple cross-word type C WFST

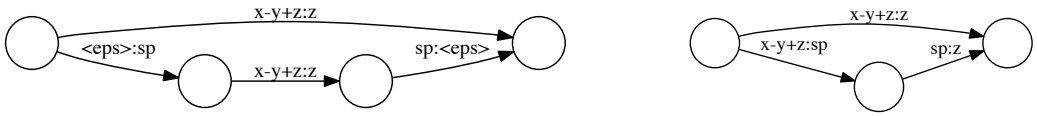
4.3 Silence Modeling

There are two silence models described and advocated to use in acoustic model by Young, *et al* [8].

- A silence model, `sil`, with the same structure as the other phonetic models, acts as a context, but is context dependent.
- A short pause model, `sp`, is tied to the silence model. Short pause model is context free.

4.3.1 Silence Modeling in Juicer

There are two different routines to apply short pause model in C WFST described by Garner *et al* [4]. The insertion of `sp` into transitions in C is illustrated in Figure 10.



Juicer's implementation A more compact form

Figure 10: Insertion of `sp` into transitions in C

To accommodate silence at the lexicon level, the insertion of `sp` into transitions in L is also needed. Figure 11 shows the three pronounciations of a single word `NO`.

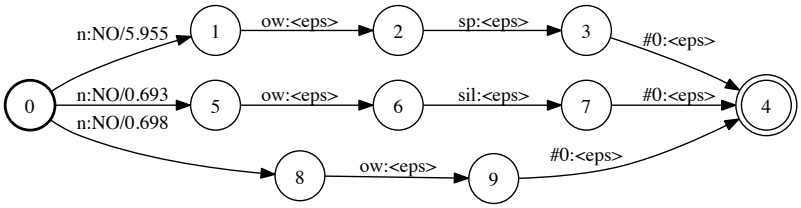


Figure 11: An example of L WFST with silence phones

4.3.2 Silence Modeling in Transducersaurus

In Transducersaurus' implementation, a silence transducer T is introduced. Thus the traditional C∘L∘G composition is modified to C∘L∘G∘T. With this approach, silences are treated as just like other words, as described in [1]. Figure 12 shows the basic representation of a silence transducer T. To allow more complex silence modeling, class-based approach can be applied in transducer T. But currently only the silence transducer in Figure 12 is implemented.

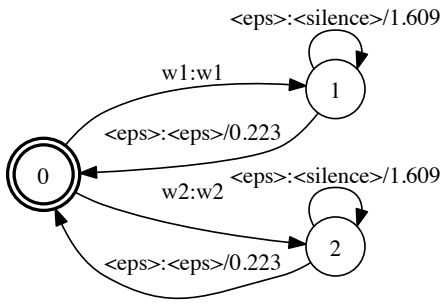


Figure 12: A simple silence transducer T

4.4 Auxiliary Symbols

If the auxiliary labels are left in L∘G, then C WFST must add some auxiliary transitions to match those in L∘G. The easiest way to implement is to add auxiliary loops on all states of C WFST, as illustrated in Figure 13.

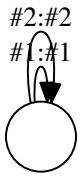


Figure 13: Auxiliary loops on states of C WFST

5 Conclusion

References

[1] Cyril Allauzen, Mehryar Mohri, Michael Riley, and Brian Roark. A generalized construction of integrated speech recognition transducers. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, volume 1, pages I–761. IEEE, 2004.

[2] Cyril Allauzen, Mehryar Mohri, and Brian Roark. Generalized algorithms for constructing statistical language models. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 40–47. Association for Computational Linguistics, 2003.

[3] Paul Dixon and Sadaoki Furui. Introduction to the use of wfsts in speech and language processing.

[4] Philip N Garner. Silence models in weighted finite-state transducers. In *Proceedings of Interspeech*, 2008.

[5] Slava Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 35(3):400–401, 1987.

[6] Mehryar Mohri. Weighted automata algorithms. In *Handbook of weighted automata*, pages 213–254. Springer, 2009.

[7] Mehryar Mohri, Fernando CN Pereira, and Michael Riley. Speech recognition with weighted finite-state transducers. *Handbook of Speech Processing*, pages 559–582, 2008.

[8] Steve J Young, Gunnar Evermann, MJF Gales, D Kershaw, G Moore, JJ Odell, DG Ollason, D Povey, V Valtchev, and PC Woodland. The htk book version 3.4. 2006.