Software Library

**GRM Library**: Grammar Library. General software collection for constructing and modifying weighted automata and transducers representing grammars and statistical language models (Allauzen, Mohri, and Roark, 2005).

Software Libraries

- **OpenGRM Libraries**: open source libraries for constructing and using formal grammars in FST form, using OpenFST as underlying representation.

  - **NGram Library**: create and manipulate n-gram language models encoded as weighted FSTs. (Roark et al., 2012)

  - **Thrax**: compile regular expressions and context-dependent rewrite grammars into weighted FSTs. (Tai, Skut, and Sproat, 2011)

- [http://opengrm.org](http://opengrm.org)
Overview

- **Generality**: to support the representation and use of the various grammars in dynamic speech recognition.

- **Efficiency**: to support competitive large-vocabulary dynamic recognition using automata of several hundred million states and transitions.

- **Reliability**: to serve as a solid foundation for research in statistical language modeling.
Language Modeling Tools

Counts: automata (strings or lattices), merging.

Models:

- Backoff or deleted interpolation smoothing.
- Katz or absolute discounting.
- Kneser-Ney models.

Shrinking: weighted difference or relative entropy.

Class-based modeling: straightforward.
## Corpus

### Input:

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>hello</td>
<td>&lt;eps&gt;</td>
</tr>
<tr>
<td>bye</td>
<td>hello</td>
</tr>
<tr>
<td>hello</td>
<td>bye</td>
</tr>
<tr>
<td>bye bye</td>
<td>&lt;unknown&gt;</td>
</tr>
</tbody>
</table>

### Program:

```bash
farcompilestrings --symbols=labels.txt --keep_symbols corpus.txt > corpus.far
```

or

```bash
cat lattice1.fst ... latticeN.fst > foo.far
```
This Lecture

- Counting
- Model creation, shrinking, and conversion
- Class-based models
Counting

Weights:

- use `fstpush` to remove initial weight and create a probabilistic automaton.
- counting from far files.
- counts produced in log semiring.

Algorithm:

- applies to all probabilistic automata.
- In particular, no cycles with weight zero or less.
Counting Transducers

- $X$ is an automaton representing a string or any other regular expression.
- Alphabet $\Sigma = \{a, b\}$. 
Counting

Program:

ngramcount --order=2 corpus.far > corpus.2.counts.fst
ngrammerge foo.counts.fst bar.counts.fst > foobar.counts.fst

Graphical representation:
This Lecture

- Counting
- Model creation, shrinking, and conversion
- Class-based models
Creating Back-off Model

Program:

```
ngrammake corpus.2.counts.fst > corpus.2.lm.fst
```

Graphical representation:
Shrinking Back-off Model

- **Program:**

\[
\text{ngramshrink --method=relative_entropy --theta=0.02 corpus.2.lm.fst > corpus.2.s.lm.fst}
\]

- **Graphical representation:**

```
1

0/0.81093

ε/1.0986

bye/0.81093

2/0.20479

ε/1.0986

bye/1.0986

3/0.54857

ε/0.27444

hello/0.89794

ε/1.0986

hello/1.5041
```
Merging/Interpolation

- **Program:**

  ngrammerge --normalize --alpha=2 --beta=3 a.lm.fst b.lm.fst > merged.fst

- **Resulting language models are mixed at relative importance corresponding to --alpha and --beta, normalizing the output LM to be a probability distribution**
This Lecture

- Counting
- Model creation, shrinking, and conversion
- Class-based models
Class-Based Models

Simple class-based models:

\[ \Pr[w_i|h] = \Pr[w_i|C_i] \Pr[C_i|h]. \]

Methods in GRM: no special utility needed.

- create transducer mapping strings to classes.
- use `fstcompose` to map from word corpus to classes.
- build and make model over classes.
- use `fstcompose` to map from classes to words.

Generality: classes defined by weighted automata.
Example: BYE = \{bye, bye bye\}.

Graphical representation: mapping from strings to classes.
Class-Based Model - Counts

Original counts.

Class-based counts.
Models

original model.

class-based model.
Final Class-Based Model
References


References


References


