# Language Understanding Systems

*Language Modeling with OpenGRM Tools*

Evgeny A. Stepanov

SISL, DISI, UniTN
`stepanov@disi.unitn.it`

# Outline

1. Language Models

2. OpenGRM for Language Modeling

3. Exercises

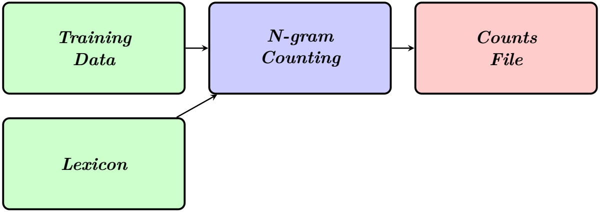
# Section 1

## Language Models

# LM Functionality

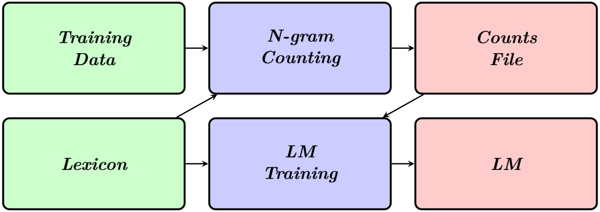
# LM Functionality

- Compute n-gram counts from the corpus



# LM Functionality

- Compute n-gram counts from the corpus
- Train LM from the n-gram counts file



*Training Data*

*N-gram Counting*

*Counts File*

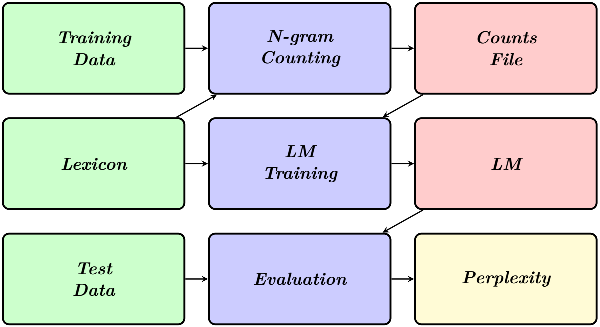
*Lexicon*

*LM Training*

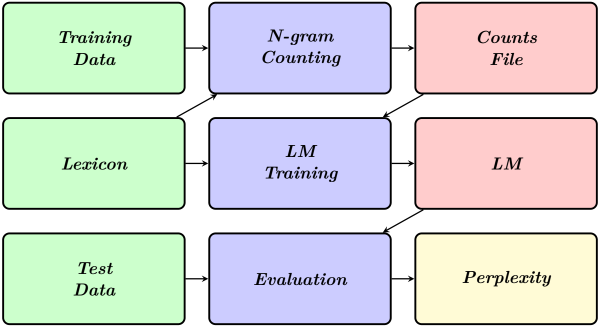
*LM*

# LM Functionality

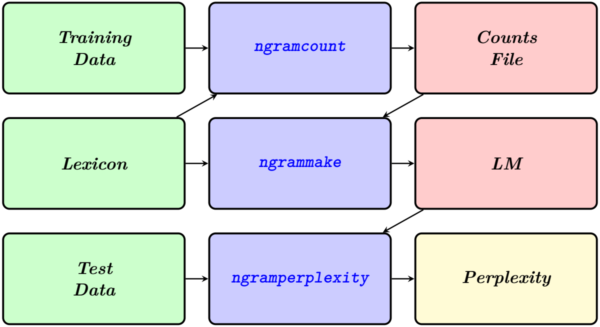
- Compute n-gram counts from the corpus
- Train LM from the n-gram counts file
- Calculate the test data *perplexity* using LM



# Language Modeling



# LM Tools: OpenGRM



# OpenGRM: Discounting

*OpenGRM*

- Absolute (Ney)
- Witten-Bell (default)
- Kneser-Ney
- Katz
- presmoothed
- unsmoothed

Section 2

# OpenGRM for Language Modeling

# OpenGRM Tools

# OpenGRM Tools

- **`ngramcount`**: takes as input a *fst archive* representing a text input and outputs an *fst* representing counts of n-grams

# OpenGRM Tools

- **`ngramcount`**: takes as input a *fst archive* representing a text input and outputs an *fst* representing counts of n-grams
- **`ngrammake`**: takes as input a *counts fst* (the output of `ngramcount`) and outputs an *fst* representing a n-gram back-off stochastic LM

Signals & Interactive Systems

# OpenGRM Tools

- **`ngramcount`**: takes as input a *fst archive* representing a text input and outputs an *fst* representing counts of n-grams
- **`ngrammake`**: takes as input a *counts fst* (the output of `ngramcount`) and outputs an *fst* representing a n-gram back-off stochastic LM
- **`ngrammerge`**: merges two *counts fsts* (the output of `ngramcount`) or two *LMs* (the output of `ngrammake`)

# OpenGRM Tools

- **`ngramcount`**: takes as input a *fst archive* representing a text input and outputs an *fst* representing counts of n-grams
- **`ngrammake`**: takes as input a *counts fst* (the output of `ngramcount`) and outputs an *fst* representing a n-gram back-off stochastic LM
- **`ngrammerge`**: merges two *counts fsts* (the output of `ngramcount`) or two *LMs* (the output of `ngrammake`)
- **`ngramapply`**: intersects n-gram model with *fst archive*;

# OpenGRM Tools

- **`ngramcount`**: takes as input a *fst archive* representing a text input and outputs an *fst* representing counts of n-grams
- **`ngrammake`**: takes as input a *counts fst* (the output of `ngramcount`) and outputs an *fst* representing a n-gram back-off stochastic LM
- **`ngrammerge`**: merges two *counts fsts* (the output of `ngramcount`) or two *LMs* (the output of **`ngrammake`**)
- **`ngramapply`**: intersects n-gram model with *fst archive*;

Other useful tools:

- **`ngramsymbols`**: produces a lexicon from an input text corpus;
- **`ngramread/ngramprint`**: utilities to read/write n-gram models from/to text file;
- others...

# FAR Tools

**FARs** – *weighted finite-state machines archives* – a concatentation of the file representation of one or more finite-state machines.

Commands:

- **`farcompilestrings`** – text to FAR
- **`farprintstrings`** – FAR to txt
- **`farinfo`** – information on contained FSMs
- **`farequal`** – compare two FARs for equality
- **`farcreate`** – create FAR from input FSTs
- **`farextract`** – split FAR into constituent FSTs

Refer to manual for options:

`http://www.openfst.org/twiki/bin/view/FST/FstExtensions#FstArchives`

# Example

Input file: text.txt
Build Language Model:

```
ngramsymbols < text.txt > lex.txt
farcompilestrings --symbols=lex.txt
                   -keep_symbols=1 text.txt > text.far
ngramcount --order=3 text.far > text.cnts
ngrammake --method=witten_bell text.cnts > text.lm
```

Generate random sentence:

```
ngramrandgen text.lm | farprintstrings
```

# Section 3

# Exercises

# Exercises

- Read Tool Manuals (for options)
- Train different language models (LM) using train.txt
  - vary order {1-3}
  - vary smoothing
  - take care of unknown words using lexicon (e.g. frequency cut-off)
  - compute LM perplexity on test.txt
  - report order & smoothing method with lowest perplexity