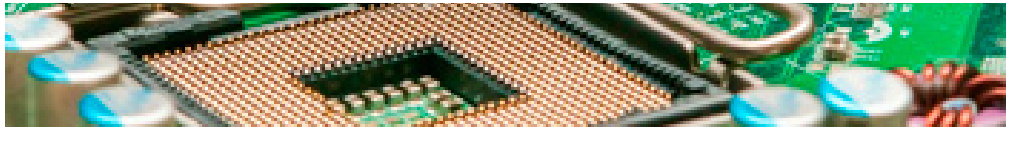
Máster Universitario en Ingeniería Informática

Sistemas Inteligentes

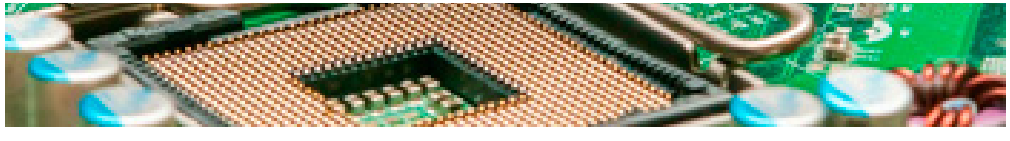
Unit 5. N-grams and FSA - Seminar

2022/2023

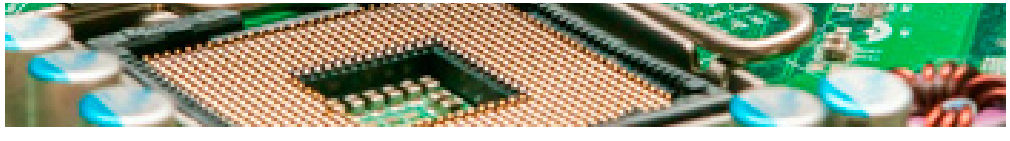


Outline

1. Introduction to LM
2. SRILM Installation
3. Introduction to SRILM
4. Introduction to FSA
5. OpenFst Installation
6. Example. Practical exercise
7. Student Projects



Outline

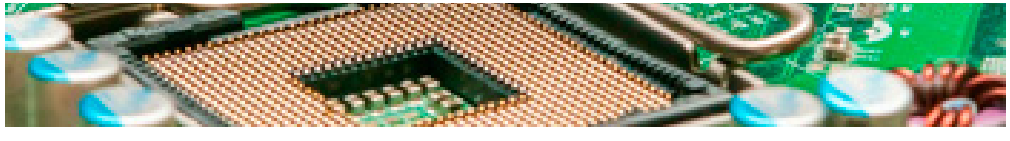


Available N-gram tools:

- **SRILM**: http://www.speech.sri.com/projects/srilm/
- KENLM: https://kheafield.com/code/kenlm/
- CMU: http://www.cs.cmu.edu/~dorcas/toolkit_documentation.html
- IRST: https://github.com/irstlm-team/irstlm
- OpenGrm: http://www.openfst.org/twiki/bin/view/GRM/NGramLibrary

Recommended web pages:

- https://en.wikipedia.org/wiki/N-gram

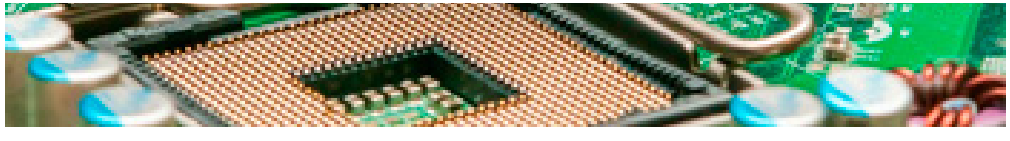


SRILM Toolkit

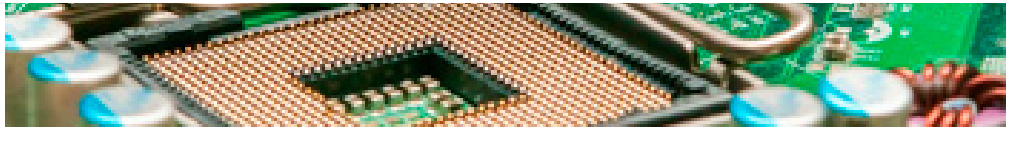
SRILM: The SRI Language Modeling Toolkit

SRILM is a toolkit for building and applying statistical language models (LMs), primarily for use in:

- Speech recognition
- Handwritten text recognition
- Statistical tagging and segmentation
- Machine translation
- Natural language processing



Outline



SRILM Installation

SRILM can be downloaded from its web page

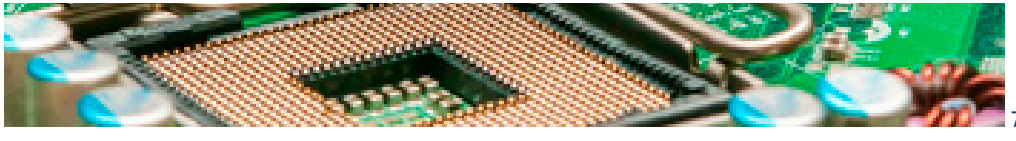
The installation is described in the INSTALL file

The compilation takes some time

The most important binary for these sessions is the `ngram-count` binary, available in PoliLabs

SRILM consists of the following components:

- A set of C++ class libraries implementing language models, supporting data structures and miscellaneous utility functions
- A set of executable programs built on top of these libraries to perform standard tasks such as training LMs and testing them on data, tagging or segmenting text, etc.
- A collection of miscellaneous scripts facilitating minor related tasks



SRILM Installation

SRILM can be downloaded from its web page

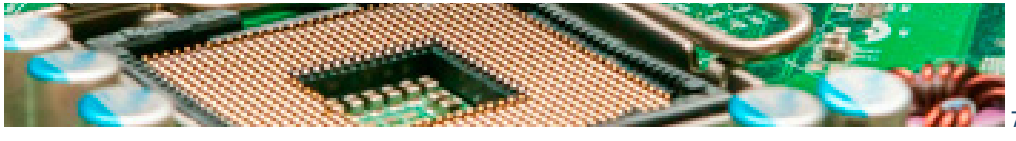
The installation is described in the INSTALL file

The compilation takes some time

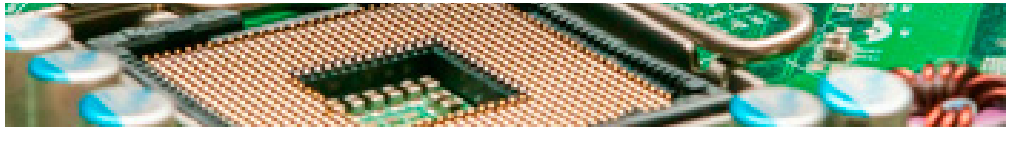
The most important binary for these sessions is the `ngram-count` binary, available in PoliLabs

SRILM consists of the following components:

- A set of C++ class libraries implementing language models, supporting data structures and miscellaneous utility functions
- A set of executable programs built on top of these libraries to perform standard tasks such as training LMs and testing them on data, tagging or segmenting text, etc.
- A collection of miscellaneous scripts facilitating minor related tasks



Outline



Introduction to SRILM

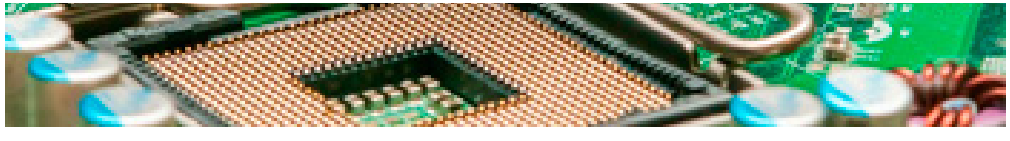
SRILM accepts text files in plain text to build a language model

Punctuation marks or new-line characters are used to split sentences

```
$ cat smallSet
el perro corre rapido
el tren azul corre veloz
el coche azul corre veloz
el perro corre rapido
el tren azul corre veloz
el coche azul corre veloz
```

Create a 2-gram LM

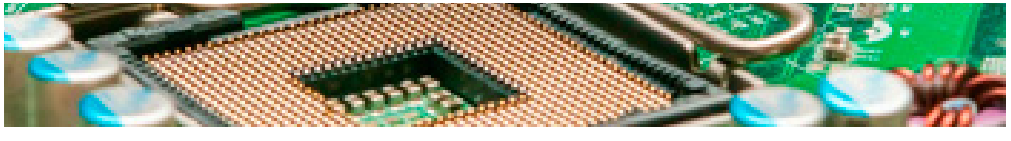
```
$ cat smallSet | ngram-count -text - -lm smallSet-2.darpa -order 2 \
  -write-vocab vocab-smallSet
```

Introduction to SRILM

DARPA format in plain text for representing an *N*-gram model (also binary)

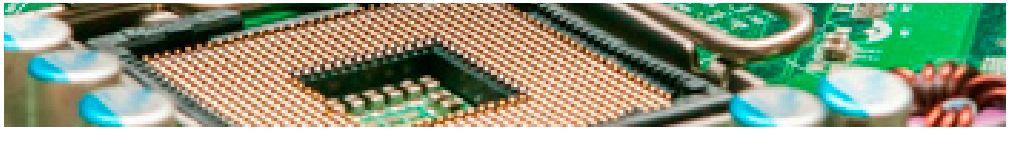
```
$ cat smallSet-2.darpa
\data\
ngram 1=10
ngram 2=12

\1-grams:
-0.7533277          </s>
-99          <s>          -99
-0.9294189          azul          -99
...
-0.9294189          veloz          -99

\2-grams:
0          <s> el
0          azul corre
-0.4771213          corre rapido
...
0          tren azul
0          veloz </s>

\end\
```

Introduction to SRILM

SRILM has a comprehensive help

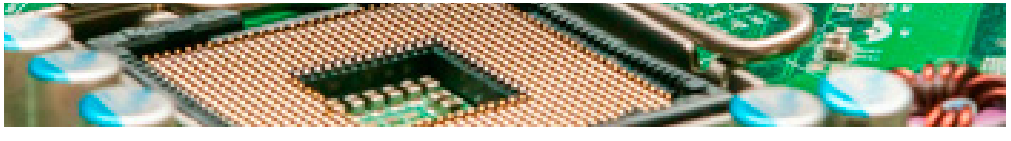
```
$ ngram-count -help
Usage of command "ngram-count"
 -version:                  print version information
 -order:                    max ngram order
                Default value: 3
 -varprune:                 pruning threshold for variable order ngrams
                Default value: 0
 -debug:                    debugging level for LM
                Default value: 0
 -recompute:                recompute lower-order counts by summation
 -sort:                     sort ngrams output
 -write-order:              output ngram counts order
                Default value: 0
 ...
```

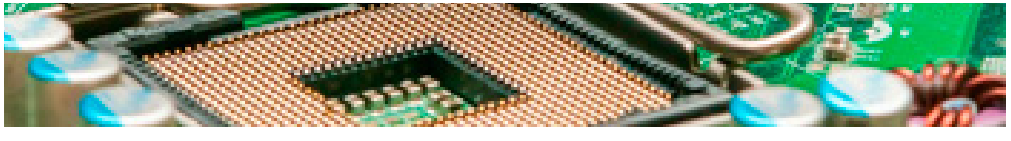
Introduction to SRILM

Relevant options:

- `-order`: order of the max ngram order
- `-text`: text file to read
- `-lm`: file where to write the ngram model
- `-write-binary-lm`: output LM in binary format
- `-write-vocab`: write vocab to file
- `-limit-vocab`: limit count reading to specified vocabulary



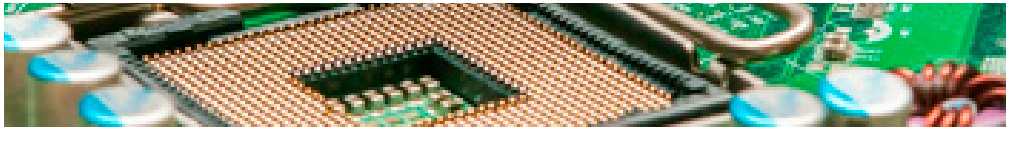
Outline



Available tools for FSA

Available tools for FSA:

- SRILM: http://www.speech.sri.com/projects/srilm/
- HSFT: http://hfst.github.io/
- SFST: http://www.cis.uni-muenchen.de/~schmid/tools/SFST/
- **OpenFst**: http://www.openfst.org/twiki/bin/view/FST/WebHome

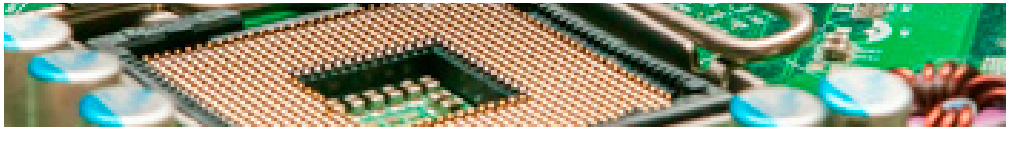


OpenFst Toolkit

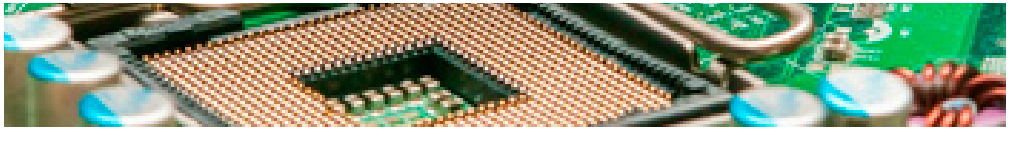
OpenFst: OpenFst is a library for constructing, combining, optimizing, and searching weighted finite-state transducers (FSTs)

FSTs have key applications in:

- Speech recognition and synthesis
- Machine translation
- Optical character recognition
- Pattern matching
- String processing
- Machine learning
- Information extraction
- ...



Outline



OpenFst Installation

OpenFst can be downloaded from its web page

The installation is described in the INSTALL file

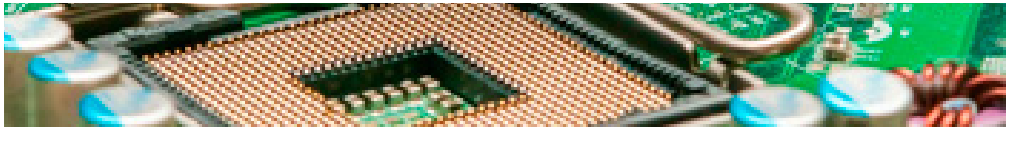
The compilation takes some time (hours!)

The toolkit is installed in PoliLabs

OpenFst consists of the following components:

- A set of C++ class libraries
- A set of executable programs built on top of these libraries to perform standard tasks
- A collection of miscellaneous scripts facilitating minor related tasks

Relevant commands are: `fstcompile`, `fstprint`, `fstrmepsilon`, and `fstdraw`



OpenFst Installation

OpenFst can be downloaded from its web page

The installation is described in the INSTALL file

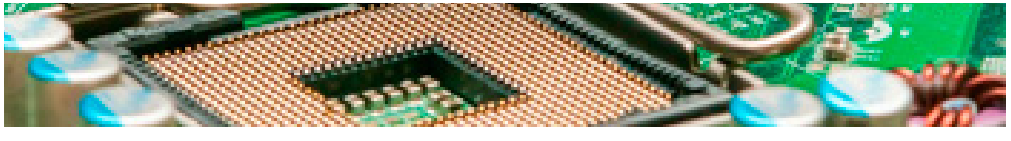
The compilation takes some time (hours!)

The toolkit is installed in PoliLabs

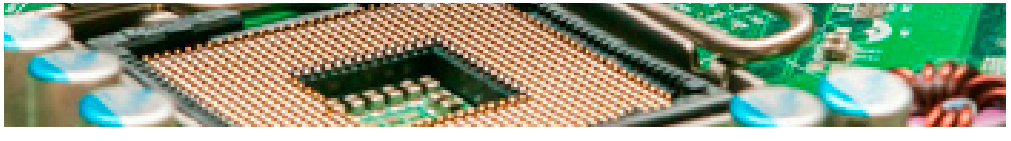
OpenFst consists of the following components:

- A set of C++ class libraries
- A set of executable programs built on top of these libraries to perform standard tasks
- A collection of miscellaneous scripts facilitating minor related tasks

Relevant commands are: `fstcompile`, `fstprint`, `fstrmepsilon`, and `fstdraw`



Outline



Example. Practical exercise

System setup

Create your folders for auxiliar executables and libraries:

```
mkdir $HOME/W/binsin
mkdir $HOME/W/libsin
```

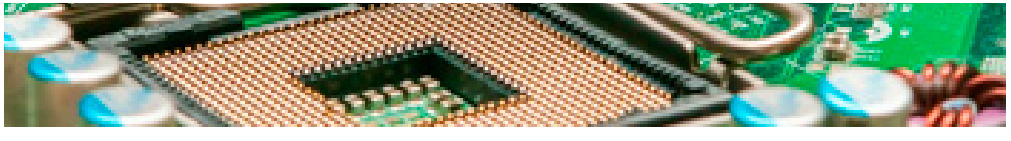
Download from PoliformaT auxiliar binary and script to `$HOME/W/binsin`:
`clean-tweets.sh`, `arpa2fst`, `str2fst.pl`

Download from PoliformaT auxiliar libraries to `$HOME/W/libsin`:
`libfst.so.16`, `libkaldi-base.so`, `libkaldi-lm.so`,
`libkaldi-matrix.so`, `libkaldi-util.so`

Modify paths to executables and libraries:

```
export PATH=$HOME/W/binsin:$PATH
export LD_LIBRARY_PATH=$HOME/W/libsin/:$LD_LIBRARY_PATH
```

It is recommended to put these lines in you `.bashrc` file



Example. Practical exercise

FSA creation

Create a 3-gram LM along with the vocabulary

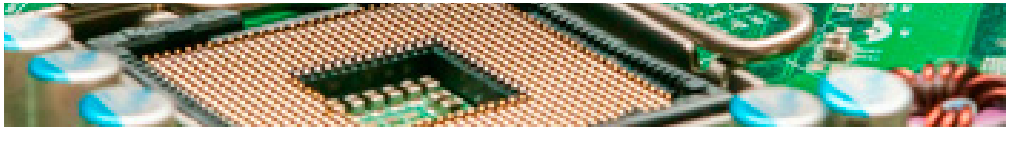
```
cat smallSet | ngram-count -text - -lm smallSet-3.darpa -order 3 \
  -write-vocab vocab-smallSet
```

The <eps> symbol and word indexes are added to the vocabulary file

```
awk 'BEGIN{print "<eps>\t0"}{printf("%s\t%d\n",$1,NR);}' \
    vocab-smallSet > vocab-smallSet.tmp
mv vocab-smallSet.tmp vocab-smallSet
```

The FSA for the LM is created

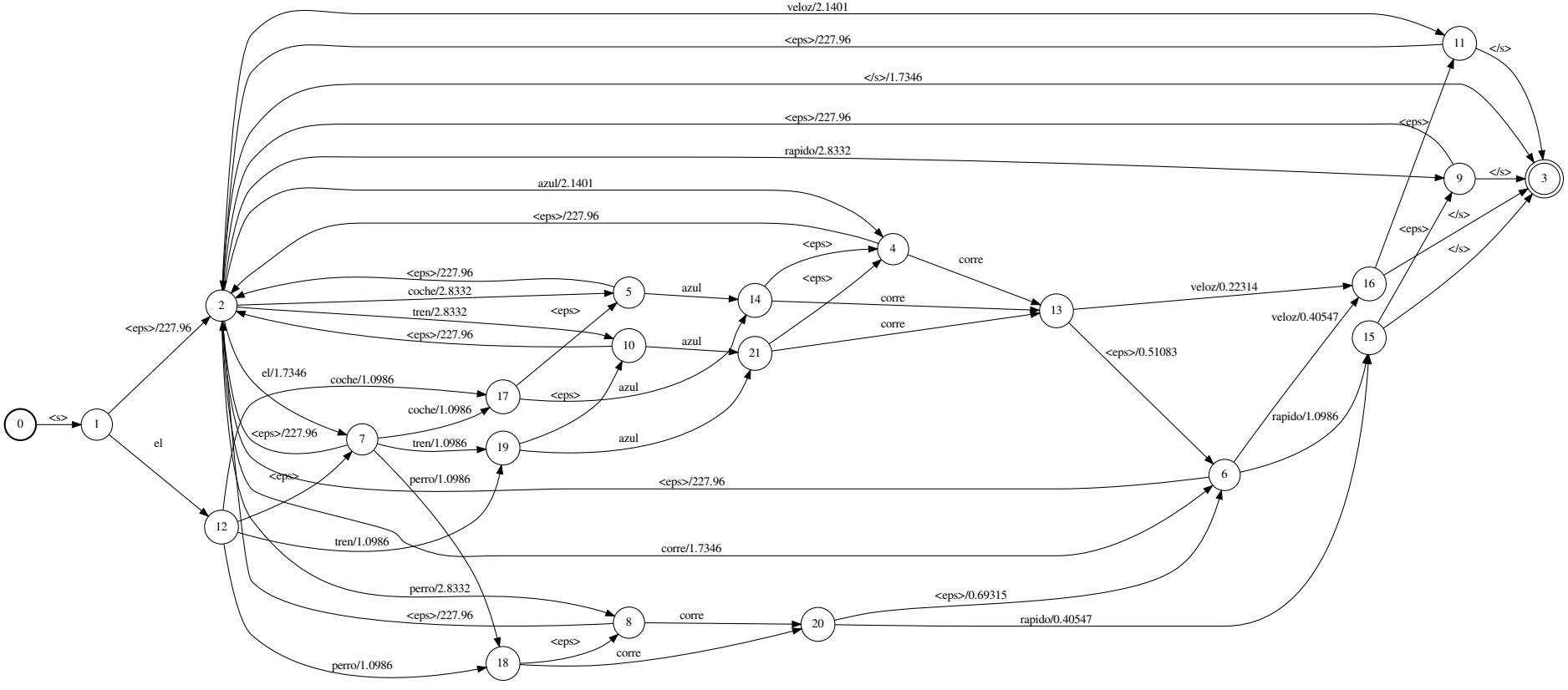
```
cat smallSet-3.darpa | arpa2fst - | fstprint | \
   fstcompile --isymbols=vocab-smallSet --osymbols=vocab-smallSet \
   --keep_isymbols=true --keep_osymbols=true > smallSet-3.fst
```

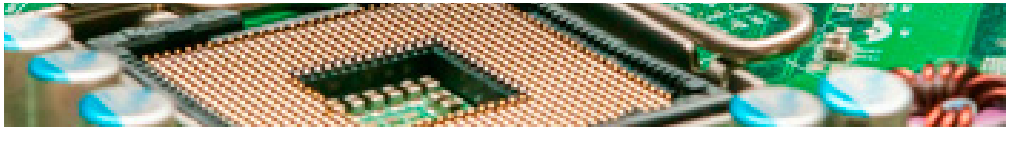



Example. Practical exercise

The final FSA can be plotted

```
fstdraw --isymbols=vocab-smallSet --acceptor \
-portrait smallSet-3.fst | dot -Tpdf > smallSet-3.pdf
```





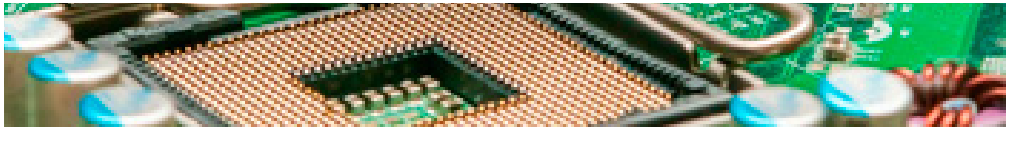
Example. Practical exercise

Practical exercise

1. Replicate the previous example.
2. Use different prune thresholds (`fstprune`) and plot the resulting FSA
3. You can generate sentences with the created model and check the type of sentences that the model is able to generate

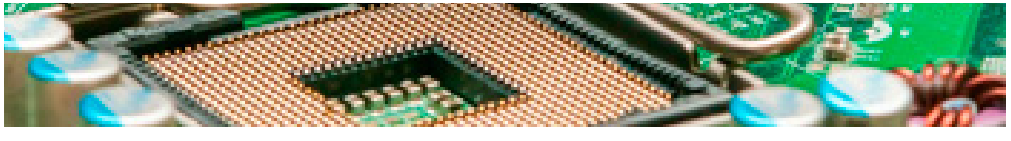
```
fstrandgen --npath=1 --select=log_prob \
  --seed=`echo $RANDOM` smallSet-3.fst | fstprint
```

4. Carry out a similar exercise with a set of sentences defined by you. The number of sentence has to be about 10 without repeated sentences



Outline

1. Introduction to LM
2. SRILM Installation
3. Introduction to SRILM
4. Introduction to FSA
5. OpenFst Installation
6. Example. Practical exercise
7. Student Projects



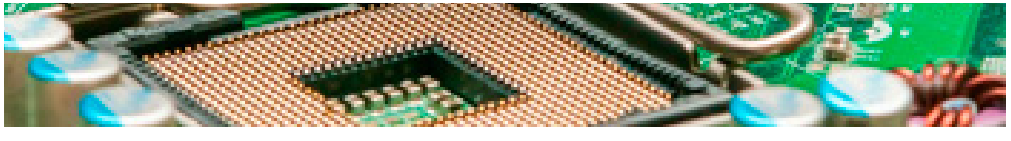
Student Projects

Student Project Main goal: to build text classifier based on FSA to classify the gender of tweets authors

Download `Stance-IberEval2017-training-20170320.zip` from PoliformaT (Unzip password: click the info icon)

Unzip and clean the dataset:

```
unzip Stance-IberEval2017-training-20170320.zip
cd Stance-IberEval2017-training-20170320
clean-tweets.sh training_tweets_es.txt > \
  training_tweets_es_clean.txt
```

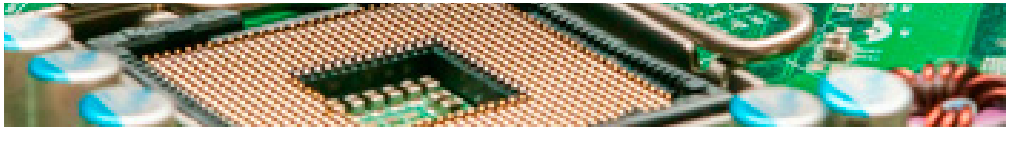



Student Projects

Process dataset for `FEMALE` class:

```
cut -d ":" -f 7 training_truth_es.txt | \
  paste -d " " - training_tweets_es_clean.txt | \
  grep "^FEMALE" | sed "s/^FEMALE //g" > \
  training_tweets_es_clean-FEMALE.txt
cat training_tweets_es_clean-FEMALE.txt | \
  ngram-count -text - -lm female_es.darpa \
  -order 3 -write-vocab vocf_es -unk
awk 'BEGIN{print "<eps>\t0"}{
  printf("%s\t%d\n",$1,NR);
}' vocf_es > vocf_es.tmp
mv vocf_es.tmp vocf_es
```

Repeat the process for the `MALE` class



Create FST for `FEMALE` class:

```
cat female_es.darpa | arpa2fst - | fstprint | \
  fstcompile --isymbols=vocf_es --osymbols=vocf_es \
  --keep_isymbols=true --keep_osymbols=true \
  > female_es.fst
```

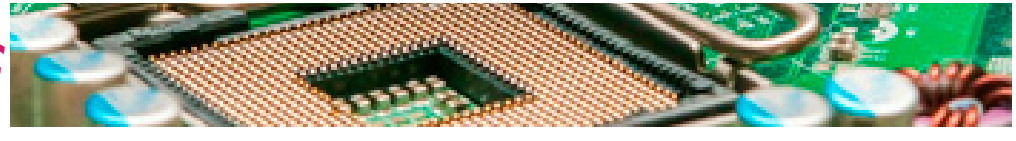
Repeat for the `MALE` class

Test a sentence:

```
ess="<s> Un dia largo por delante #27S </s>"
echo $ess | str2fst.pl vocf_es |
  fstcompile --isymbols=vocf_es --osymbols=vocf_es \
  --keep_isymbols --keep_osymbols > essf.fst
fstcompose essf.fst female_es.fst | fstshortestpath | \
  fstrmepsilon | fstprint | \
  LC_NUMERIC=C awk '{s+=$NF;} END{print(s);}'
```

Repeat with `vocm_es` and `male_es.fst` (use `essm.fst`)

The output with a lower value is that of the assigned class

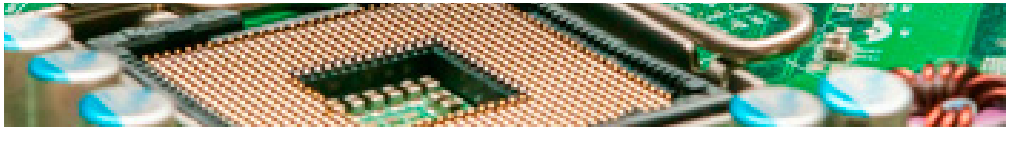


Student Project. Exercise B2.1. (Max mark: 5 out of 35)

Perform the same process for the Catalan part (`training_tweets_ca.txt`):

- Clean tweets
- Extract the `FEMALE` and `MALE` tweets
- Generate the `FEMALE` and `MALE` n-gram and vocabulary
- Generate the `FEMALE` and `MALE` FST

Upload to the PoliformaT task the generated files for vocabulary and FST (`vocf_ca`, `vocm_ca`, `female_ca.fst`, `male_ca.fst`) before next **May 8th at 19:00** for evaluation.



Student Project. Exercise B2.2. (Max mark: 5 out of 35)

Perform a gender classification on a blind Spanish test set that would be available in a PoliformaT task:

- Select two gender-balanced subsets of the current training data to perform experiments with different parameters:
  - *n*-gram degree
  - Different discounts
- Train the final *n*-gram with the best parameters and the whole training dataset
- Perform classification of the given blind test set and send classification result

Upload to the PoliformaT task a text file with a label in each line (`FEMALE` or `MALE`), corresponding to the classification of the tweet in the same line in the test file. Send it before next **May 19th at 19:00** for evaluation.

Grade would be $\min\{5, 8a\}$, where a is the classification accuracy.