Natural Language Processing

Lab 4: Language Modeling

1 Introduction

In this lab, we are going to build and evaluate n-gram models for the task of probabilistic language modeling, or predicting the next word in a text. We will make use of a standard toolkit called SRILM, and we will explore different orders of n as well as different smoothing techniques.

Acknowledgment: Thanks to Emily Bender for letting us reuse and modify an older lab.

2 Data and preprocessing

To train our language models, we will use the same collection of Sherlock Holmes stories as in the previous exercises (`holmes.txt`). To evaluate the models, we will use three different texts:

1. *His Last Bow*, a collection of Sherlock Holmes stories by Conan Doyle (not included in `holmes.txt`)
2. *The Lost World*, a novel by Conan Doyle
3. *Stories by English Authors: London*, short stories written around the same time as the other works

These texts are in the files `his-last-bow.txt`, `lost-world.txt`, and `other-authors`, respectively.

Note on tokenization: The SRILM toolkit requires tokenized text in a different format from the one we have been using so far. Instead of having one token per line and blank lines to separate sentences, we should now have one sentence per line and spaces to separate tokens. To tokenize the texts, you should use `tokenizer6.py`, which performs exactly the same tokenization as `tokenizer5.py` but produces its output in the format required by SRILM.
You should tokenize the three texts as well as `holmes.txt`.

3 Train a language model

To train a language model on `holmes.txt`, we need to run a command like the following:

```
ngram-count -order 2 -text holmes-tok.txt -addsmooth 0.01 -lm bigram-add
```

This commands tells SRILM to count bigrams (`-order 2`) in the training file (`-file holmes-tok.txt`) and to create a language model and save it in a file called `bigram-add` (`-lm bigram-add`) with additive smoothing with (`-addsmooth 0.01`). By varying parameters such as n-gram order and smoothing method, we can create different language models.

4 Evaluate a language model

To evaluate a language model created as above on `his-last-bow.txt`, we run:

```
ngram -order 2 -lm bigram-add -ppl his-last-bow-tok.txt
```

This command tells SRILM to use a bigram model (`-order 2`) based on the previously stored model (`-lm bigram-add`) to estimate the perplexity of the test file (`-ppl his-last-bow-tok.txt`). The output generated by this command should be something like:

```
file his-last-bow-tok.txt: 5080 sentences, 81393 words, 2729 OOVs
0 zeroprobs, logprob= -177261 ppl= 130.829 ppl1= 179.224
```

This tells us that the test file contained 5080 sentences and 81393 words, of which 2729 were out-of-vocabulary (OOV), meaning that they did not occur in the training set. The model did not find any zero probabilities (thanks to the smoothing), the log probability was -177261 and the perplexity (ppl) was 130.829.[1] Remember that perplexity, as well as the related entropy measure, tells us how surprised or confused the model is when seeing the test data, so lower perplexity is better.

[1] The second perplexity measure (ppl1) is computed without end-of-sentence tokens and can be ignored for now.

5 Explore different test sets

Our first test set is drawn from the same author and the same genre as our training set. What happens if we apply the model to the other two test sets? What happens if we apply the model to the training set itself? First think about what you think will happen and discuss with your classmates. Write down your expectations: E.g.: I think perplexity will be lower on text set X than on test set Y. Run the experiments and compare the results with your intuitions. Do the results match your expectations?

6 Explore different language models

Our first language model was a bigram model with simple additive smoothing. What happens if we change the n-gram order? Explore at least unigrams and trigrams on all three test sets with the same smoothing method. Decide which model works best and switch the smoothing method for this model to Kneser-Ney smoothing (a type of backoff smoothing). Make a table with the different results to ease the interpretation of the results.

Note on SRILM options: To change the n-gram order, simply change `-order 2` to `-order` n (for whatever n you want to try). Please note that the n-gram order used for testing cannot be higher than the n-gram order used for training (but it can be lower). To change smoothing method, replace `-addsmooth 0.01` by `-kndiscount` (for Kneser-Ney discounting). If you want to explore further options, consult the SRILM manual at `http://www.speech.sri.com/projects/srilm/manpages/ngram-count.1.html`.