Assignment 1

Seminar on Statistical Language Modeling

Universität des Saarlandes

Jon Dehdari

November 9, 2014

Question 1: Fun with Ken and Tom

This excercise is to get you familiar with using two popular LM toolkits. You *don't* need to understand *yet* the theory of the models. Download and install the KenLM[1] and RNNLM[2] language modeling toolkits. For KenLM, try to download from the Git repository (instead of downloading a tarball), so you can easily grab updates later. KenLM is a great free language modeling toolkit for doing interpolated modified Kneser-Ney-smoothed *n*-gram language models (that's all it does). Yes, it's written by a guy named Ken :-) RNNLM is a free LM toolkit for doing recurrent neural-network language models.

RNNLM requires a held-out development set, in addition to a training set. KenLM only requires a training set. For RNNLM, use 400 hidden nodes. For KenLM, use 4-grams.

1. Download the "English News Crawl (articles from 2007)" corpus from: `http://www.statmt.org/wmt14/training-monolingual-news-crawl/news.2007.en.shuffled.gz`

2. Use only the first 100,000 lines, and tokenize the resulting smaller corpus, using a simple tokenizer found at `http://www.ling.ohio-state.edu/~jonsafari/corpus_tools/tokenize.pl`

3. Convert the text to lowercase. You can use the script `http://www.ling.ohio-state.edu/~jonsafari/corpus_tools/lowercase.pl` . You can use the following Unix commands to do the above instructions:

```
zcat corpus.gz | head -n 100000 | ./tokenize.pl | ./lowercase.pl > corpus.head100000.tok.lc
```

   If you don't understand the above Unix command, ask someone who does.

4. Split the resulting subcorpus into training/development/test sets, with the ratio 18:1:1 respectively, using the script `http://www.ling.ohio-state.edu/~jonsafari/corpus_tools/generate_splits.pl` . Use the `--help` argument for usage info.

[1] Main website: `http://kheafield.com/code/kenlm` ; Github: `https://github.com/kpu/kenlm` ; Git command: `git clone https://github.com/kpu/kenlm.git` . It depends on several Boost headers and libraries.

[2] `http://rnnlm.org` . The current Makefile is a little broken, so compile it using: `make CC=c++`

5. Train KenLM and RNNLM models on the training set. For RNNLM, also give the development (dev) set file as a command-line argument. Report training times and binary model file sizes for each LM software. In Unix environments, you can get time info by prepending "`time `" before a command. FYI, RNNLM might take several hours to train this training set.
6. Report both probabilities and log10 probabilities of each of the first 8 words in the first sentence of the dev set, up to "*candidates*".[3] You can get word-level info for RNNLM by adding the "`-debug 2`" argument. How does each LM software handle unseen (OOV) words, as well as `</s>` ? Briefly discuss.
7. Report perplexity (PP) on the entire test set, including OOV words and not including them. Unfortunately RNNLM only reports PP for non-OOV words, so just report non-OOV PP for RNNLM.

Question 2: Discount on Beer

(From Koehn (2010, p. 215))

Given the training data:

Training Data:

Count	Count of Counts
1	5000
2	1600
3	800
4	500
5	300

The word *beer* occurs as history in three bigrams in the data:

Count	Bigrams
4	beer drinker
4	beer lover
2	beer glass

(a) What are the discounted counts under Good–Turing discounting for the three given bigrams?

(b) The amounts from discounting counts are given to a back-off unigram model. Using such a back-off model, what are the probabilities for the following bigrams?

1. $p(\text{drinker}|\text{beer})$
2. $p(\text{glass}|\text{beer})$
3. $p(\text{mug}|\text{beer})$

Note: $p(\text{mug}) = 0.01$. State any assumptions that you make.

References

Koehn, Philipp. 2010. *Statistical Machine Translation*. Cambridge University Press.

[3] KenLM reports log10 probabilities, in the third field after the word. RNNLM reports probabilities, in the unit interval (which is $[0, 1]$).