Assignment 2

Seminar on Statistical Language Modeling

Universität des Saarlandes

Jon Dehdari

December 1, 2014

$p(\textbf{fun}|\textbf{language model}) = 1$

By now you should have downloaded and installed the KenLM[1] toolkit. Now let's build language model querying software!

You can work in groups of 2–3 people for this. There should be at least one programmer in each group, and preferrably one non-programmer (to evenly distribute these folks).

The software will read one or more UTF-8 sentences, one per line, from stdin on the Unix/Linux command-line, and outputs the $\log_{10}$ probabilities of each word to stdout. The output should be just like the output of KenLM.[2] Your software should allow for arbitrary length *n*-grams. The one required command-line argument of your program is the name of the ARPA file (see below for discussion of this file format). Also allow common command-line args like `--help`, `--version`, etc. Thus the command-line invocation should be:

```
./lm-query lm.arpa < test.txt > test.probs 2> test.pp
```

If you're using a scripting language, then you can use filename suffixes like `lm-query.py`, `lm-query.pl`, etc. Also if you're using a scripting language, include the appropriate information in the shebang. Use minimal dependencies, especially those beyond what are found on a standard Debian/Ubuntu system.

The non-programmer(s) in your team can be experts in the ARPA format and how backoff-weights are applied. These folks can also look into how KenLM handles unigrams, unseen (OOV) words, and end-of-sentence markers, and follow such practices.

Output total perplexity (both for unseen and for seen) to stderr.

Use Git for version control and collaboration. If you don't know how to use Git, well, now's a good time :-) All members of your team (progs and non-progs) should learn how to use the basics of Git. Here are some links to learn how to use it:

- `http://git-scm.com/book/en/v2/Getting-Started-About-Version-Control`

[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]The word ID (first field after the equals sign) can be a dummy number. At the end of the output, KenLM outputs the total perplexity to stdout, but you should follow better practice and output perplexity to stderr.

- http://git-scm.com/doc
- https://en.wikipedia.org/wiki/Git_(software)
- https://training.github.com/kit/downloads/github-git-cheat-sheet.pdf
- https://en.wikibooks.org/wiki/Git/Introduction

In your main project directory, have a `bin/` directory (where all softare executables go) and an optional `src/` directory if applicable (where non-executable source code goes). If applicable have a Makefile in the main project directory. You project should be hosted on Github, unless you have a good reason othewise. You should have a `readme.md` in the main project directory describing the software name, how to install/compile, command-line usage, copyright, etc. If you're not familiar with Markdown, have the non-programmer(s) learn the format (see below). Have the non-programmer(s) write documentation in the `readme.md` file. Here are some links to learn the markdown format:

- https://en.wikipedia.org/wiki/Markdown
- http://daringfireball.net/projects/markdown/basics
- http://daringfireball.net/projects/markdown/syntax

Local usage: `markdown readme.md > readme.html`

Appendix: ARPA Format

ARPA doc: http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html

$$\begin{aligned}
p(w_3|w_1,w_2) &= \text{if trigram exists } \rightarrow p_3(w_1,w_2,w_3)\\
&\quad \text{else if bigram } w_1,w_2 \text{ exists } \rightarrow bo_wt_2(w1,w2)\times p(w_3|w_2)\\
&\quad \text{else } \rightarrow p(w_3|w2)
\end{aligned}$$

$$\begin{aligned}
p(w_2|w_1) &= \text{if bigram exists } \rightarrow p_2(w_1,w_2)\\
&\quad \text{else } \rightarrow bo_wt_1(w_1)\times p_1(w_2)
\end{aligned}$$

Everything before `\data\` is ignored. Afterwards the counts of each n-gram order are listed, such as: Markov order $x+1 = n$-gram order n, for SRILM and KenLM.

```
ngram 1=83321
ngram 2=734706
ngram 3=1510236
ngram 4=1844304
```

Then, the $\log_{10}$ probabilities and backoff weights of each n-gram order are listed, as in:

```
\1-grams:
-0.9775657  <unk>   0
-inf    <s> -0.89692545
-2.2764258  </s>    0
-4.5306624  dave    -0.13577949
-6.348688   aneckstein  -0.107379325
...
\2-grams:
-2.6448765  research </s>   0
-1.9577274  company </s>    0
-4.10923    <s> dave    -0.052287683
-4.464891   said dave   -0.052287683
...
...
\4-grams:
-2.7922783  she said , </s>
-1.1493444  roberts said , </s>
...
\end\
```

Notice that there is no backoff weight for the highest *n*-gram order, since we'll never use these.

See also:
http://kheafield.com/professional/avenue/kenlm_talk.pdf slide 10 "Example Queries" especially
http://www1.icsi.berkeley.edu/Speech/faq/grammarfmts.html