CSE 5525 / Foundations of Speech and Language Processing / Eric Fosler-Lussier
Instructions for LM tutorial

1) SRILM should be installed in your VM (run "which ngram-count")
2) Read the SRILM paper (especially section 3) and the LM intro. You may also want to refer to the man pages.
3) Let's start with a small example. Put the following data into a file "small_data"

A B A C
B A A B
C A B
C C A C
B

a) Compute a "counts file" for bigrams using

ngram-count -order 2 -text small_data -write small_data.2counts

b) Take a look at the counts file. What do you notice about this file? What symbols have been added?
c) How would you compute P(B|A) from this file?
d) Now run this data through ngram-count again, but write out a bigram LM instead of a counts file. You can do this by reading in the counts file, or directly from the text (you can ignore the warnings, they occur because this is a small file):

ngram-count -order 2 -read small_data.2counts -lm small_data.2gram

or

ngram-count -order 2 -text small_data -lm small_data.2gram-a

e) Notice that the output language model format gives you the probabilities in log base 10. Is P(B|A) the same as you calculated? Why or why not?
f) Turn off discounting by doing the following (you are telling the system to use the maximum-likelihood estimates for all counts above 0):

ngram-count -order 2 -gt1max 0 -gt2max 0 -read small_data.2counts -lm small_data.nodisc.2gram

g) Note that it will still compute a backoff (I'm not sure why), but the probabilities should be different. Compare them to each other and your original estimate is.

4) Take a look at the data in the files WSJtrain and WSJtest

5) Now we are going to take a larger example and train from it. Build a bigram grammar using the file WSJtrain. Call this file WSJtrain.2gram

6) You can test the perplexity of the grammar by using ngram on WSJtest

ngram -lm WSJtrain.2gram -ppl WSJtest

7) You can also get per-sentence perplexity by using "–debug 1", or per-word probabilities by using "-debug 2". What words are very unlikely in the first sentence?

8) Now try changing some of the parameters (such as gt1min, gt2min, order, or the discounting type) and see if you can reduce the perplexity of the test set.

9) BONUS: connecting to FSMs. You can think of a grammar as a finite state machine, where the states encode history. See if you can develop a grammar that encodes the probabilities for a bigram grammar constructed from the data in Step 3. You can encode the bigram probabilities as weights on arcs between states. An interesting question is what to do with backoff weights (we can discuss this if you get that far, but think about it for a bit). See if you can compute the probability of a sentence by intersecting the sentence with the language model FSA, and compare this to the answer you get from SRILM.