Russ Ronalds Comp Ling

3/1/19 HW3

Part 1

Question 1

Bi-gram:

# S # It gives this because it sees S commonly as the beginning of a word, and commonly as the end of a word, so it thinks that just the S alone should be great! (Because it can’t see far enough) (This will probably be the case for most one phoneme words it creates)

# JH D # As it doesn’t know that English-speakers enjoy having vowels between most consonants, and the testing has seen these phonemes at the end and beginning of words, it thinks this should be just fine.

# P EH M IH T S T ER T ER S IH K AA AY R D EH K AH D # Similar to above, in this abnormally long word, there exists the quite rare T-S-T triplet. As always, since this model can’t see far enough, T going to S is great and then S going to T, while that would be rare in English. (and presumably this model could get stuck on a TSTSTS loop for a little)

# P AO R AH M AA R IY Z D SH AH S K S F IH G IY D # Just more crazy triplets that would be really hard to say in English. Z D SH?? A S K S F??? Also, this model is bad at telling us what sort of words are likely to be this long.

# CH EY N D AH L AH N S K L Z # The coda of this word might almost be something close to saying icicles really fast. But, as is essentially the same problem as all of the above, the model can’t tell that we have gone way too long without a vowel.

Tri-gram:

This model is doing a much better job in making words that are pronounceable, but the words seem much more… German or Scandinavian than English. Since this model looks at the world in triplets, it loves making consonant vowel consonant vowel runs.

# # N AA M IH S AH N T IY T AH D Z # While there aren’t any forbidden consonant strings, now this sounds foreign in some way, maybe actually Spanish instead of Germanic.

# # W AA B IY AH N D HH IH S G AH L Z # This model also seems to create more words that sound like compound words to me. The N D HH sounds close to ‘handhold’ or something. These end up seeming less likely to be *new* English words, I think, just because they seem to be words that have come down from long ago, as opposed to more modern neologisms(?)

# # EH B Y AH L EY T ER S AA D R EY V AH N S IY Z # Here’s another foreign-sounding word. I can’t really place why it doesn’t seem English enough, besides that I don’t think English tends towards so many simple consonant vowel or vowel consonant pairs.

# # R EH L ER AH N HH AE P AH TH AA N D IH CH AH K IH N D AA R L OW #

I think because there isn’t really a good length of word model built into the ngrams, the word lengths for trigrams tend to be much longer than the average English word (compared to the tendency of the bigram model to produce single phoneme words).

# # S T R AY B # If I have the phonetics correct about these phonemes, this is pronounced stribe (NOT Streeb, but stryb), which seems pretty weird. It is getting the R AY B sequence from mainly words like ascribe, prescribe, describe, etc., which, while they are common themselves, that sound is actually not very common, I think. So, this model of course can’t determine that these are all just transformations of some ‘scribe’ root.

Question 2

The trigram model seems to generate words that sound more natural, particularly because it creates words that have vowels between consonants. The bigram model tends to create more words that are one phoneme long and don’t work alone, or words that contain consonants next to each other that don’t work in English. Because the trigram model can look at more context, it can “see” better what can and cannot go together, even though it is still getting things wrong, mostly because it now tends to create too many consonant vowel consonant vowel runs.

Part 2

Question 3

The perplexity is as follows:

Bigram, x.txt: 13.9

Trigram, x.txt: 14.0

Bigram, y.txt: 61.0

Trigram, y.txt: 15.6

A relatively lower number means that a corpus is less perplexing and more likely given the training data. The set of the three words “abandon” “abandoned” and “abandoning” produces a perplexity of 8.5 under both bigram and trigram analysis; the set of all words starting with “a” in our training document produces a 14.2 under bigram and 9.9 under trigram; 25 randomly generated *trigram* words of length 5 phonemes (generated from a modified, un-extended, ngram script) produces a perplexity of 17.9 under bigram analysis and 12.4 under trigram analysis. So, a perplexity under twenty seems to be a good range for fairly English-like words, and under 15 seems to be for a set of words that is actually found in the training corpus/English language or produced directly from that data.

So x.txt is pretty close to as-English-like as 24 words could be that don’t show up in the training data, it seems. I would guess that it might have been made from a bigram word generator, basically. Y.txt is vastly less English-like, relative to x.txt, under bigram analysis, but is approximately as perplexing as x.txt when looked at under trigram analysis. I would hazard a guess that this means that y.txt is a list of words made randomly from a trigram generator. An add-1 smoothed trigram generator will have 64000 options to choice from, and I think the frequencies will have less effect on the probabilities at this level, so surely it will occasionally produce low-likelihood next-to-impossible-in-English words, but whose probability doesn’t look that terrible?[[1]](#footnote-1) A bigram analysis looks at y.txt and its never-before-seen phoneme combinations and determines that some of them don’t make sense, but a trigram analysis approves of their likelihood a bit more, due in large part to the add-1 smoothing.

(Relative to some of my original larger perplexity calculations, like 4.9\*10^157, y.txt is also preeeetty close to English-like, though. Ha. JK)

Question 4

Well, I think a bit of this was explained in question 3 (i.e. why one model might distinguish between the sets better), but the probability comparison can do more. A bigram analysis of y.txt shows that only three of the words of the corpus have a probability that falls in the range of the words in x.txt, and those are, of course, the highest three probability, and they are close only to the very lowest probability words of x.txt. A trigram analysis, though, has a much more overlapping probability range between the two corpra. Only y.txt’s 2 lowest probability words are outside the range of x.txt’s probabilities, and only x.txt’s top highest probable word is outside the range of the y.txt words.

So, the bigram model shows how the y.txt corpra is less English-like better than the trigram model and it is easier to pick a probability threshold for this model. I think a conservatively low probability to pick as a bigram threshold would be 9.9\*10-9. This would still capture some words in the y.txt corpra, but only three for this particular corpra, which out of 24 I think seems acceptable. Obviously, a better way to adjust is to change how we smooth. Changing from an add-1 smoothing model to add-lambda smoothing and a stepped way of testing (5-fold, leave one out, other(?)) should give us much better perplexities as they are more tailored to the ngrams themselves.

1. I think the lesson here is that the larger the n of the ngram, the larger one needs the training data to be, or else smoothing (especially add-1) will essentially level the probabilistic field. If there was only one three phoneme word [‘#’ ‘AH’ ‘AH’ ‘AH’ ‘#’] in the training data, and an add-1 smoothing trigram was used to look at it and then judge perplexity of a corpus (still based on 40 phonemes, even though they don’t show up in the data), every triplet of phonemes would have 1/64003 chance, except for #AHAH, AHAHAH, and AHAH#, which would each have 2/64003 ≈ 1/32000. So, the data-proven trigrams are twice as likely to show up as any one other trigram, but they are almost 32000 times less likely to show up than all the others combined. Whereas if this was done for a bigram, there would be 2 bigrams with 2/1600, one (‘AH’ ‘AH’) with 3/1600, and 1598 with 1/1600—more reasonable than the trigram probabilities, even though they are still not usefully reasonable. [My math may be a little off here, but the magnitude should be right, so the lesson is still valid! Ha.] [↑](#footnote-ref-1)