Report

Tokenization

Sentence level tokenization was done on the basis of the knowledge that each line in the .txt file was a new sentence. Word Tokenization was done using regex, that handles the following cases:

1. ? ! . punctuations.
2. Separates , ( ) ‘ - as different tokens
3. Processes don’t aren’t isn’t Sam’s as single tokens.
4. Doesn’t separate . in Mr. Mrs. Dr. and numerical values like 3.12

Perplexity scores

LM1

- *Avg. Train Perplexity*: 4.284909580178163
- *Avg. Test Perplexity*: 13229.186428230216

LM2

- *Avg. Train Perplexity*: 2.754752175818304
- *Avg. Test Perplexity*: 331.37547330408955

LM3

- *Avg. Train Perplexity*: 3.4184223445091204
- *Avg. Test Perplexity*: 20906.023541314356

LM4

- *Avg. Train Perplexity*: 2.215480397160927
- *Avg. Test Perplexity*: 666.7004429278854

For each LM, a text file with perplexity scores of each sentences is calculated in the following format:

```
avg_perplexity
sentence_1 <tab> perplexity
sentence_2 <tab> perplexity
...
```

Analysis

- The corpus given for training was very small, and consisted of only around 20,000 training sentences and a thousand testing sentences. This led to occurring of unappeared context in most of the testing sentences, and the way these unappeared context was handled drastically affected the performance of the Language Model.
- From the results, we can see that all four models give really good perplexities on the training data. This might be considered an indication that all the models are well-trained, and main issues are with unappearing words and context in the testing data.
- We can see that Witten Bell shows better perplexity than Kneyser-Ney, primarily because it is known to better distribute discounted value, and thus resulting in a better probability estimate.
- It might also be related to better handling of unknown context in Witten Bell as compared to Kneyser-Ney.