Introduction to NLP

Tokenization

Tokenization is done in the following manner :

1. Tokens are stored in a 2D list depicting tokens per sentence in the corpus.
2. 3 start tags and 1 end tag is added to all the sentences.
3. All the alphabets are converted to lowercase.
4. All the special characters and numbers are replaced with an empty string.
5. Handling of Unknown Words :
    - In order to handle unknown words in the test set or input sentence, the concept of unknown words is used which ensures we don't get probability = 0.
    - All the unigrams are sorted with their frequencies. Then 0.05% unigrams from the vocabulary are replaced with an unknown word - <unknownwordknown>.
    - Now if a word is encountered in the test set that is not present in the vocabulary, it is replaced with the unknown word.

Language Modelling

For the scope of the assignment, we are supposed to do 4-gram modeling.

For the 4-gram modeling, the data structure chosen is a dictionary in the following manner :

1. For the unigrams, a dictionary is created that represents all single words with their frequencies.
   Format :

   Unigram : c

2. For the bigrams, a 2-nested dictionary is created i.e.a dictionary of dictionaries is created where the key in the main dictionary is mapped to another dictionary which in turn stores the frequency of all possible bigrams in the text corpus.
   Format :
   Bigram : {
       Bigram : c
   }

3. For the trigram, a 3-nested dictionary is created i.e. a dictionary of dictionaries of dictionaries where the key in the main dictionary is mapped to a second dictionary which in turn is mapped to a third dictionary that stores the frequency of all possible trigrams in the text corpus.
   Format :

   ```
   Trigram : {
           Trigram : {
                   Trigram : c
           }
   }
   ```

4. For the trigram, a 4-nested dictionary is created i.e. dictionary of dictionaries of dictionaries of dictionaries where the key in the main dictionary is mapped to a second dictionary which in turn is mapped to a third dictionary which is also mapped to a fourth dictionary that stores the frequency of all possible four grams in the text corpus.
   Format :

   ```
   Fourgram : {
           Fourgram: {
                   Fourgram : {
                           Fourgram : c
                   }
           }
   }
   Fourgram_map =
   ```

Smoothing

- **Kneser Ney Smoothing**

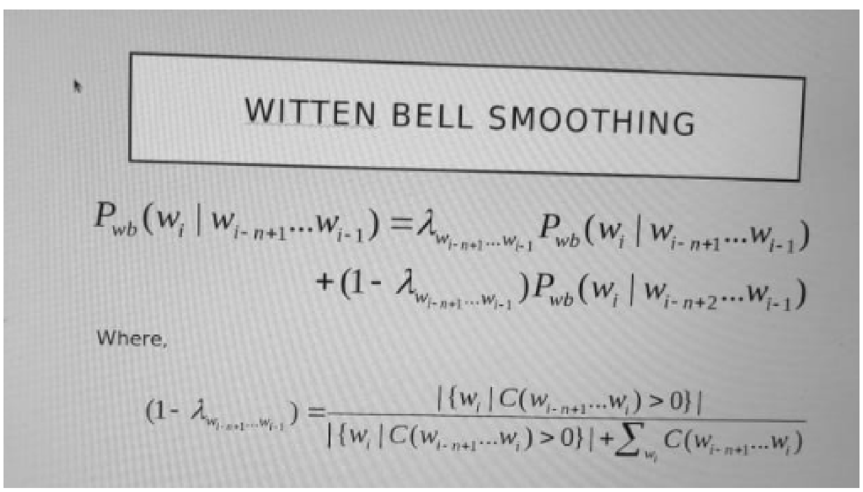
$$P_{\mathrm{KN}}(w_i|w_{i-n+1:i-1}) = \frac{\max(c_{KN}(w_{i-n+1:i}) - d, 0)}{\sum_v c_{KN}(w_{i-n+1:i-1}\, v)} + \lambda(w_{i-n+1:i-1}) P_{KN}(w_i|w_{i-n+2:i-1})$$

$$\lambda(w_{i-1}) = \frac{d}{\sum_v C(w_{i-1}v)} |\{w : C(w_{i-1}w) > 0\}|$$

$$c_{KN}(\cdot) = \begin{cases} \text{count}(\cdot) & \text{for the highest order} \\ \text{continuationcount}(\cdot) & \text{for lower orders} \end{cases}$$

$$P_{\mathrm{KN}}(w) = \frac{\max(c_{KN}(w) - d, 0)}{\sum_{w'} c_{KN}(w')} + \lambda(\epsilon)\frac{1}{V}$$

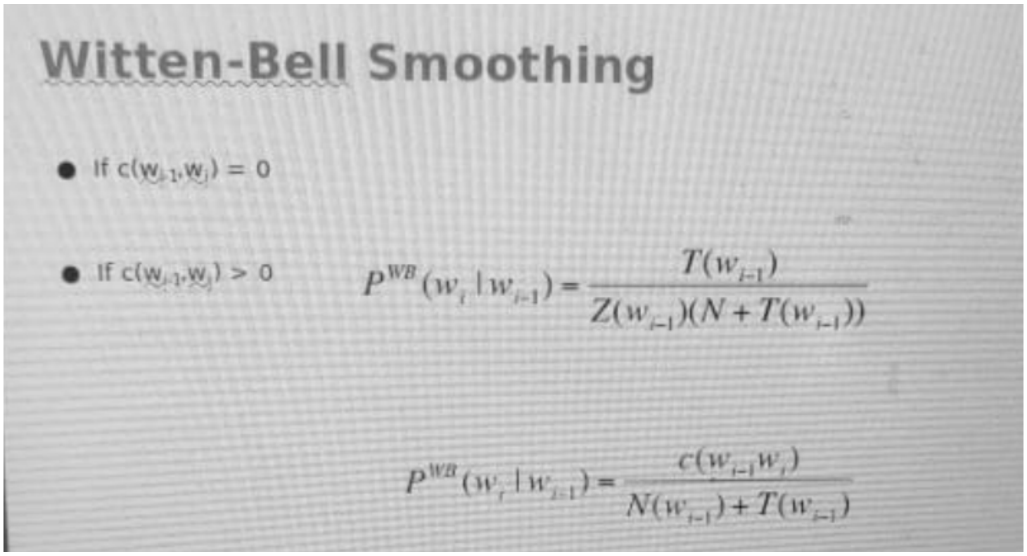
- **Witten Bell Smoothing**


WITTEN BELL SMOOTHING

$$P_{wb}(w_i | w_{i-n+1}\dots w_{i-1}) = \lambda_{w_{i-n+1}\dots w_{i-1}} P_{wb}(w_i | w_{i-n+1}\dots w_{i-1}) + (1 - \lambda_{w_{i-n+1}\dots w_{i-1}}) P_{wb}(w_i | w_{i-n+2}\dots w_{i-1})$$

Where,

$$(1 - \lambda_{w_{i-n+1}\dots w_{i-1}}) = \frac{|\{w_i \,|\, C(w_{i-n+1}\dots w_i) > 0\}|}{|\{w_i \,|\, C(w_{i-n+1}\dots w_i) > 0\}| + \sum_{w_i} C(w_{i-n+1}\dots w_i)}$$



Perplexity Score

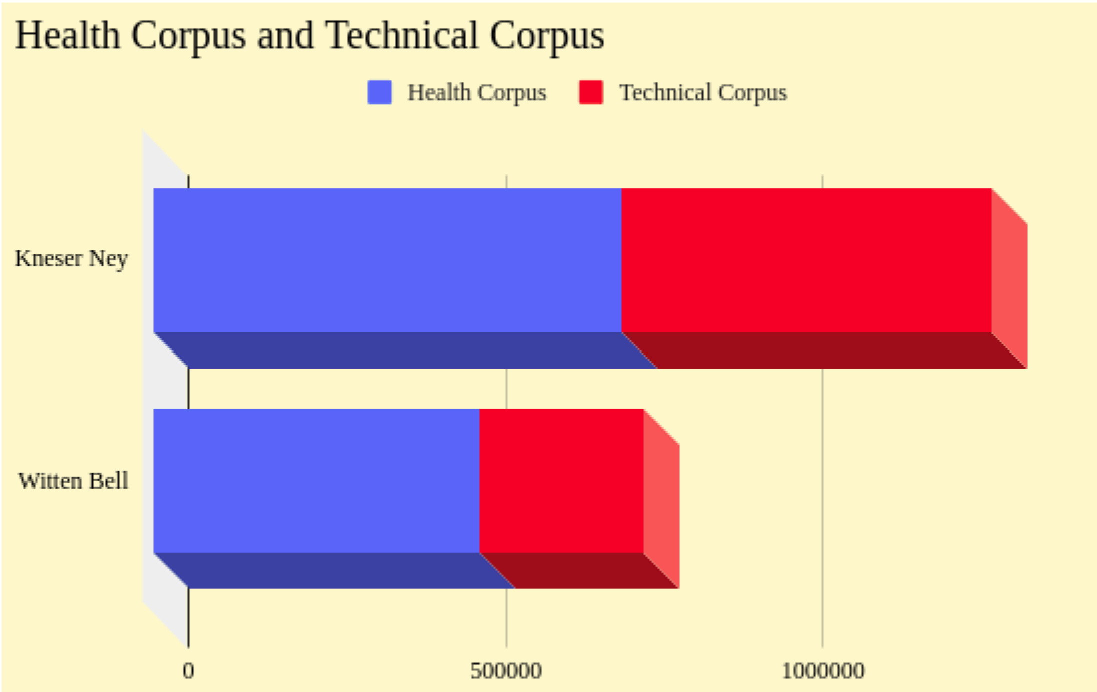
Perplexity is one of the metrics used for evaluating language models.

The perplexity (sometimes called PP for short) of a language model on a test set is the inverse probability of the test set, normalized by the number of words.

For a test set W = w1w2 . . . wN :

$$\begin{aligned} \mathrm{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}} \end{aligned}$$

We c



an use the chain rule to expand the probability of W :

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$$

Analysis

The following observations were made :

1. Kneser Ney took more time as compared to Witten Bell. Thus Witten Bell is more conservative.
2. On comparing the average perplexity scores on the test sets:
   - Health English Corpus
     Average Perplexity Score by Kneser Ney ~ 7 lakhs
     Average Perplexity Score by Witten Bell ~ 5 lakhs
   - Technical Domain Corpus
     Average Perplexity Score by Kneser Ney ~ 5 lakhs
     Average Perplexity Score by Witten Bell ~ 2 lakhs

   This clearly shows that Witten Bell is conservative.
3. Other possible reasons for more time complexity could be:
   - In Kneser Ney smoothing technique, we require to calculate all possible prefixes in the PContinuation Count.