Report

Steps:

```
$ unzip 2018114016.zip
$ cd 2018114016
$ python3 language_model.py <n-1,2,3> <k/n> corpus.txt
A prompt will occur:
$ input your sentence: <input test sentence here>
```

Cleaning the data:

Corpus data is cleaned/preprocessed. Numbers are replaced by empty string. Multiple \n are replaced by single \n. Parentheses, other brackets are removed. Alternate sentence endings such as !,? are replaced by .\n. Multiple spaces are replaced by single space. This data is then used as corpus and fed to the grams function which fills the unigram, bigram and trigram matrices accordingly.

Smoothing:

The idea behind smoothing is to compensate/manage grams which haven't been seen before so the overall probability does not become zero just because a new token occurred.

How kneserney does this is by using something what's called interpolation by absolute discounting. Basically, the Lower order model becomes more necessary/significant only when the count in higher order model and so weights are assigned accordingly. For counts greater than 1, a fixed discount value is subtracted and weights of different order models are assigned as per the count.

How Witten Bell tackles this issue is by doing what is called a backoff. If a higher model has 0 count, we backoff to a lower order model. It tries to capture the idea that if a context appears more number of times, a new token is more likely to appear in that context.

Results

n=2

Sentence	Witten Bell	KneserNey
I am a man	0.000144241470552597	1.68586977709641E-05
The king is not pretty	4.94549823315507E-06	2.66751547008925E-07
Apple is red	0.00748713865752082	Float division error

n=3

Sentence	Witten Bell	KneserNey
I am a man	0.0205246100767598	0.00146986771190593
The king is not pretty	0.00294012874306326	5.58177612116176E-05
Apple is red	0.143269230769231	0.0384615384615385