NLP A1 report

Performance of different smoothing techniques:

Perplexity was tested by splitting train data into 0.8:0.2

Words occuring only 1 time in training data were treated as OOV and replaced by UNK tokens. Total count of such UNK tokens was around ~ 21k. While calculating perplexity without smoothing, unseen token was treated as UNK. Although this is the suggested use of UNK tokens in all of literature, it gives a misleading picture since unknown words and contexts have unusually high probability.

Perplexity calculations:

No smoothing:

n	2	3	4
value	165	270	750

AddK:

k	1e-2	1e- 3	1e- 4
value	3000	2500	3000

Good turing:

n	2	3	4
value	5600	118	9

Stupid backoff:

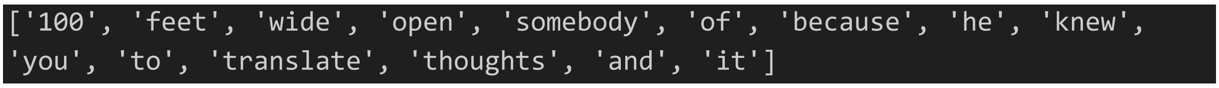
n	2	3	4
value	152	63	38

Interpolation:

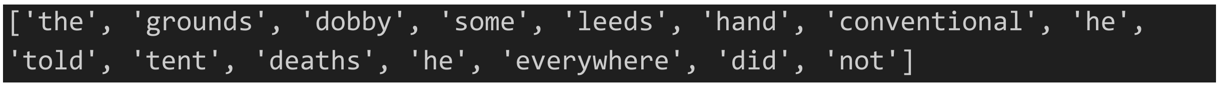
n	2	3	4	5
value	520	600	500	650

Kneyser-Ney:

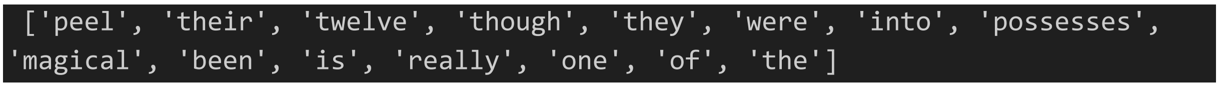
Example of generated text using various smoothings:



```
['100', 'feet', 'wide', 'open', 'somebody', 'of', 'because', 'he', 'knew',
'you', 'to', 'translate', 'thoughts', 'and', 'it']
```



```
['the', 'grounds', 'dobby', 'some', 'leeds', 'hand', 'conventional', 'he',
'told', 'tent', 'deaths', 'he', 'everywhere', 'did', 'not']
```



```
 ['peel', 'their', 'twelve', 'though', 'they', 'were', 'into', 'possesses',
'magical', 'been', 'is', 'really', 'one', 'of', 'the']
```

```
 ['anyone', 'gleaning', 'the', 'process', 'nearly', 'headless', 'nick',
'known', 'economy', 'to', 'be', 'happiness', 'satisfying', 'quiney', 'do']
 ['crookshanks', 'two', 'abolitionists', 'that', 'coatings', 'from',
'dumbledore', 'did', 'not', 'soften', 'gilderoy', 'hit', 'undressing', 'just',
'after']
```

ERROR CORRECTION

For a given word O (supposedly an error) in the test corpus, probability that there should have been another word S is P(S|O) = P(O|S) * P(S)/P(O)

We want a word S which maximizes this probability for a given O. We can ignore the denominator. P(S) can be approximated using the ngram model be calculating the probability of S given the context. P(O|S) is basically the probability of error.

3 ways to model P(O|S) were tested: Jaro – winkler similarity, sequence matcher similarity (difflib) and the transition probability as suggested by Church & Gale (1990).

The 3rd metric seemed logically stronger and turned out experimentally better. But the transition probability depends a lot on the dataset, hence it was optimal to determine them from the given misspelling dataset.

Since the data was not enough to find probability for transition from every letter, simple damerau – levenshtein distance was considered. Out of the 3187 words in test corpus:

- **Insertion errors**: 13
- **Deletion errors**: 51
- **Substitution errors**: 86
- **Transposition errors**: 9

This gave transition probabilities (like 86/3187 ~ 0.027) and the errors were assumed to be independent of each other hence probabilities were multiplied.
Punctuations and EOS, SOS, UNK tokens & digits were blacklisted from being replaced.

For candidate generation only candidates in the vocabulary upto edit distance 2 were considered since the search space of 3 edits became quite large and searching slow. Words of size 1 were ignored in the vocab.
Words in the test set which were not in vocab were assigned very low probability so that they have a high chance of being replaced. For each word the best replacement was decided(including the word itself) and the replacements were sorted on the basis of probability. Finally the top (length of sentence/5) corrections were considered.

Corrections were not considered in the context since incorrect words were generally far away in the test data

Whiile calculating log probability, instead of log P(O|S) + log P(S), we chose log P(O|S) + k*log P(S) where k is a hyperparameter, since ngram doesn't give a very accurate measure of P(S) and the order of P(O|S) and P(S) might differ significantly.

Also a probability threshold was kept for the ratio between probability of proposed replacement and the original word. Following are the results for edit distance =1 & 2, value of n for ngram = 1,2,3,4 & various smoothing techniques by finding optimal value for threshold and k (found using coordinate descent):

Smoothing for n=1,2,3,4

No smoothing		0.84	0.837	0.843	0.84
AddK		0.84	0.84	0.834	0.84
Interpolation		0.834	0.843	0.844	0.848
Stupid Backoff		0.834	0.85	0.848	0.85
Good turing		0.76	0.75	0.79	0.82
Kneser Ney		0.836	0.848	0.845	0.84