Homework 1
Abhijit Prakash Bhatnagar

Part 1

Report the perplexity scores of the unigram, bigram, trigram language models for your training, dev and test sets, and elaborate all your design choices (e.g., if you introduced START symbols in addition to STOP symbols, how you handled out-of-vocabulary words). Briefly discuss the experimental results. Do you see any issue with the current language model?

Perplexity scores:

	Train - small	Dev - small	Test - small
Unigram	1001.958852	837.592034	837.501113
Bigram	33.240650	infinity	infinity
Trigram	3.959869	infinity	infinity

Design choices:

1. I add both start and stop symbols.
2. I count symbols as words
3. I did not convert cases of words
4. Unking is done at the time of reading the raw data, and not as part of the Ngram code.
5. I unked 0.5% of words in train data set that appeared once.

The perplexity on previously seen data (train) decreases with increase “n” in ngrams (trigram > bigram > unigram). The problem with this language model is that without smoothing, the perplexity of previously unseen n-grams becomes infinity. This happens because the numerator when computing the maximum likelihood estimate is 0.

Part 2
Question 1

(a) Report perplexity scores on training and dev sets for various values of K. Try to cover a couple of orders of magnitude (e.g. K = 10, K = 1, 0.1, ...).

K	Train - small	Dev - small
10	27822.358529	29669.699039
1	11076.527281	17003.066190
0.1	2217.251929	9168.199637
0.01	380.062887	5601.955443
0.001	72.616991	3975.834031

(b) Putting it all together, report perplexity on the test set, using different smoothing techniques and the corresponding hyper-parameters that you chose from the dev set. Specify those hyper-parameters.

With add-k [K=0.001] smoothing, the perplexity on testCorpusSmall with trigrams is 3985.191197

With Linear interpolation and add-k smoothing [K=0.001000, L1=0.500000, L2=0.400000, L3=0.100000], the perplexity on testCorpusSmall with trigrams is 374.952455

Question 2

If you use only half of the training data, would it in general increase or decrease the perplexity on the previously unseen data? Discuss the reason why.

With only half the training data, the perplexity on previously unseen data will generally increase because the language model will not have the depth and variety of n-grams to represent the language correctly. Additionally. The unk-words will misrepresent relations within the words of an n-gram.

Question 3

If you convert all words that appeared less than 5 times as UNK (a special symbol for out-of-vocabulary words), would it in general increase or decrease the perplexity on the previously unseen data compared to an approach that converts only a fraction of words that appeared just once as UNK? Discuss the reason why.

The outcome really depends upon the similarity between previously seen and previously unseen data. If the two datasets are highly similar (for example, 2 excerpts from a Shakespeare play, or 2 articles from a single author), then I would expect over unking to increase the perplexity somewhat. However, for datasets that are not highly similar, over unking will decrease the perplexity on previously unseen data. This happens because the probability mass shifts towards “unk unk” type n-grams.

Part 3

Sketch out how you can make use of language models for text categorization. Use equations whenever possible.

We can use Naïve Bayes (probabilistic classifier) for text categorization. For a document d, out of all classes $c \in C$ the classifier returns the class $\hat{c}$ which has the maximum posterior probability given the document. An estimate of a class is denoted by a ^.

$$\hat{c} = \text{argmax } P(c|d) \quad \text{- Eq 1}$$

Using Bayes Rule,

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)} \quad \text{- Eq 2}$$

We substitute Eq 2 into Eq 1:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d) = \underset{c \in C}{\operatorname{argmax}} \frac{P(d|c)P(c)}{P(d)}$$

- Eq 3

The denominator is dropped because P(d) does not vary with each possible class (c).

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d) = \underset{c \in C}{\operatorname{argmax}} P(d|c)P(c)$$

- Eq 4

The most probable class $c\hat{}$ given some document D is that which has the highest product of 2 probabilities : the prior probability of the class P(c) and the likelihood of the document P(d|c):

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(d|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}}$$

- Eq 5

Any document d can be represented as a set of features:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(f_1, f_2, ..., f_n|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}}$$

- Eq 6

Naïve Bayes classifiers make 2 simplifying assumptions – (1) the position of a word does not matter; and (2) a conditional independence assumption: probabilities $P(f_i|c)$ are independent given the class c and hence can be 'naively' multiplied as follows:

$$P(f_1, f_2, ..., f_n|c) = P(f_1|c) \cdot P(f_2|c) \cdot ... \cdot P(f_n|c)$$

- Eq 7

Eq 7 can be expressed as:

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{f \in F} P(f|c)$$

- Eq 8

To apply naïve Bayes classifier to text, we consider word positions by simply walking as index through every word position in the document:

$$\begin{aligned} \text{positions} &\leftarrow \text{all word positions in test document} \\ c_{NB} &= \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in positions} P(w_i|c) \end{aligned}$$

- Eq 9

To reduce underflow errors, we express probabilities in the log space:

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} \log P(c) + \sum_{i \in positions} \log P(w_i|c)$$

- Eq 10

For text categorization tasks such as -- determining what language a given piece of text is written in, or who the most likely author of the text is – the naïve Bayes features in the above equation are replaced with n-gram probabilities instead.

Part 4

Question 4.1

Use the full dataset provided under CSEP517-HW1-Data-Full.zip for problem 1 and 2 above, and report the numerical results.

Problem 1:

	Train - Full	Dev – Full	Test - Full
Unigram	1033.365181	908.892538	908.611301
Bigram	41.585063	infinity	infinity
Trigram	4.916969	infinity	infinity

Problem 2:

K	Train - small	Dev - small
10	34725.034217	38111.692445
1	11776.535027	19867.568063
0.1	2231.352480	10029.610863
0.01	391.491873	5749.768621
0.001	78.056750	3855.158297

Question 4.2

(a) Report perplexity scores on training and dev sets for various values of λ_1, λ_2, λ_3. Report no more than 5 different values for your λ's.

K	λ_1	λ_2	λ_3	Perplexity – Train small	Perplexity – Dev small
0.01	0.4	0.4	0.2	131.408409	495.464993
0.001	0.5	0.4	0.1	43.808302	375.630557
0.0005	0.65	0.3	0.05	33.027767	399.608084
0.001	0.6	0.3	0.1	46.759688	425.560746
0.0051	0.9	0.1	0	135.389468	812.910972

(b) Report perplexity on the test set, using the corresponding hyper-parameters that you chose from the dev set. Specify those hyper-parameters.

I chose the hyperparameters: [K=0.001000, L1=0.500000, L2=0.400000, L3=0.100000] because of the lowest perplexity on previously unseen data (dev-small). The test set perplexity for these hyperparameters is 374.952455