Jingyuan Xing
11-411 Natural Language Processing
HW3

Subtask 1

Q1 It's basically each n-gram's dictionary size

For uniform model, we need to multiply the probability of each word from the training data, as if they are equally likely to appear. Therefore the number of unique N-gram types is equal to number of unique words in the training data. = 1734

For unigram model, we need to multiply the real probability of each word which appeared in the training data. Therefore the number of unique N-gram types is equal to number of unique words in the training data. = 1734

For bigram model, we need to multiply the count of (word i-1, word i) tuple divide by the count of word i-1. Therefore the number of unique N-gram types is equal to the number of such unique tuples in the training data. In another word, we are looking for the number of unique subsequent pairs. = 28748

For trigram model, we need to multiply the count of (word i-2, word, i-1, word i) tuple divide by the count of tuple (word i-2, word i-1). Therefore the number of unique N-gram types is euqal to the number of such unique three-element-tuples in the training data. = 71809

Q2

For uniform, the most common n-gram is
For unigram, the most common n-gram is ','
For bigram, the most common n-gram is 'UNK ,'
For trigram, the most common n-gram is 'this privacy policy'

Q3

A few interesting n-grams for the sports category are 'personally identifiable information', 'football talk llc', and the nba.com network.

Q4

It's interesting to see that a lot of privacy policy are about online privacy and nba stuff. Also it's interesting to note the among all of n-grams we have generated, there are no words that uses first person narrative such as "I".

Subtask 2

train data = sports.txt, test data = games.txt

My approach is that I start by setting lambda for one model as 1, and the other three as 0. Do this for all four models. Then after comparing the perplexity output, I found that the trigram is giving the smallest perplexity, so I start by lowering the trigram lambda a bit, rising other models a bit, and see if I can get a smaller perplexity.

The table for hyper-parameter tuning is attached below

perplexity	uniform	unigram	bigram	trigram
1642.6467	1	0	0	0
206.4902	0	1	0	0
41.1128	0	0	1	0
6.0246	0	0	0	1
6.1252	0.01	0.01	0.01	0.97
6.5880	0.05	0.05	0.05	0.85