Che-An Wu

ECS 189G

Zhou, Yu

HW#2 Written Parts

Part I

1) Size = 100

```
error rate by word:      0.512476007677543 (20559 errors out of 40117)
error rate by sentence:  0.993529411764706 (1689 errors out of 1700)
```

2) Size = 500

```
error rate by word:      0.383478325896752 (15384 errors out of 40117)
error rate by sentence:  0.975294117647059 (1658 errors out of 1700)
```

3) Size = 1000

```
error rate by word:      0.281651170326794 (11299 errors out of 40117)
error rate by sentence:  0.948235294117647 (1612 errors out of 1700)
```

4) Size = 1500

```
error rate by word:      0.234962734003041 (9426 errors out of 40117)
error rate by sentence:  0.920588235294118 (1565 errors out of 1700)
```

5) Size = 5000

```
error rate by word:      0.108258344342797 (4343 errors out of 40117)
error rate by sentence:  0.8 (1360 errors out of 1700)
```

6) Size = 20000

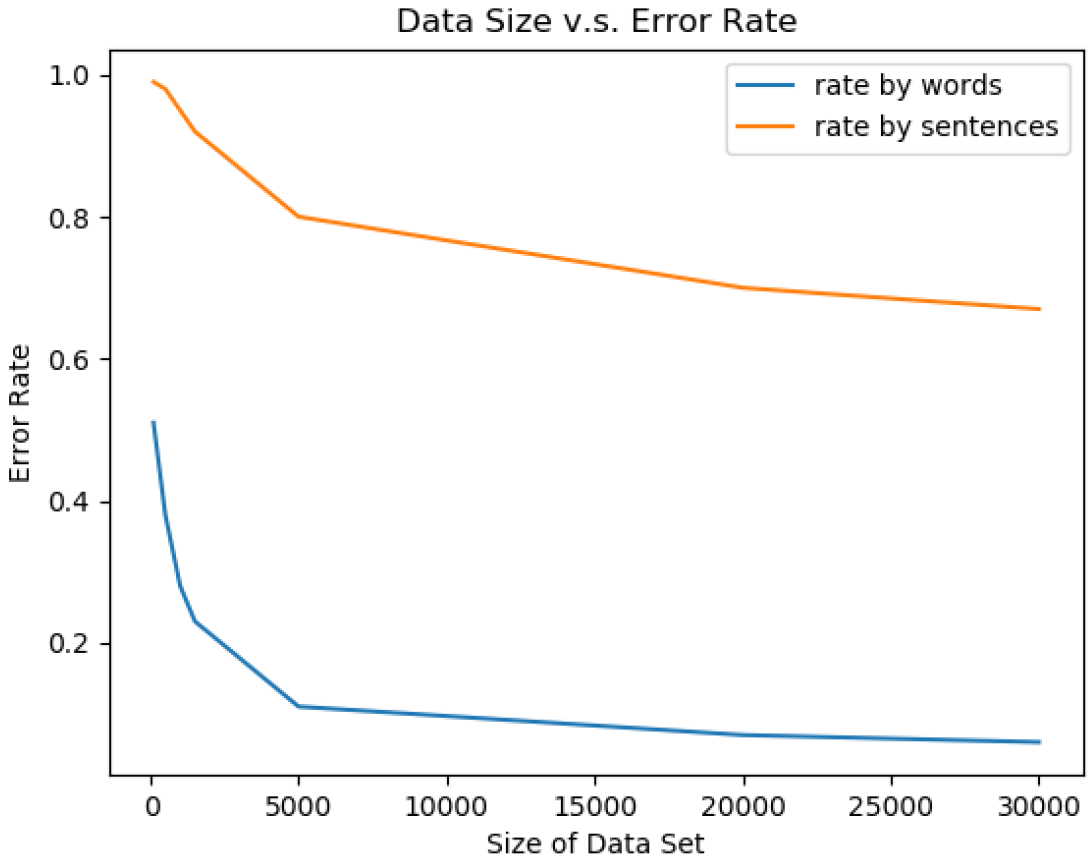
```
error rate by word:      0.0655831692300023 (2631 errors out of 40117)
error rate by sentence:  0.703529411764706 (1196 errors out of 1700)
```

7) Size = 30000

```
error rate by word:      0.0571827404840841 (2294 errors out of 40117)
error rate by sentence:  0.667647058823529 (1135 errors out of 1700)
```

8) Full data size

```
$ ./tag_acc.pl ptb.22.tgs my.out
error rate by word:      0.0540917815389984 (2170 errors out of 40117)
error rate by sentence:  0.655882352941176 (1115 errors out of 1700)
```



From the 6 evaluations I did based on training data sets with different size (i.e. 5000, 10000, 15000, 20000, 25000, and 30000), I observe a decreasing trend in error rate as the data set's size increases, for both error rate by words and by sentences. Overall, we see that the error rate by sentences is higher than error rate by words for all data set's size. For both by words and by sentences, the error rate decreases almost exponentially when data set size increases from 0 to 5000, but the decreasing trend begins to saturate as the data set size continues to increase from 5000 to 10000, 15000, 20000, 25000, and 30000.

Overall, the plot tells us that – as data size increases, the error rates decrease, which means that our tagging becomes more accurate and is able to match more tags/tokens in the test set. Thus, the more data we can train our model with, the better and more accurate result we get.

Part 3

Initially, in my attempt to improve my bigram mode, I approached the problem by implementing the back-off approach that I used in the previous homework (HW1). The idea is to always use bigram whenever possible, but "back-off" to use the unigram in certain cases. For this homework, I think that it is possible to have the transition probability from word A to word B in

my training set to be zero, which is an over-generalizing assumption that might bring down my accuracy. Thus, for these cases of probability equal to zero, I use the back-off approach and use the unigram probability of word A only. However, this model with back-off did not perform well. In fact, it performs worse than my original unsmoothed bigram model.

After this failure attempt, I decided to try the Add-k Smoothing. At first, I decided to smooth both emissions and transitions. The reason to smooth emissions is that if there is any word W in the test data such that P(W | Ti) = 0 for all tags T, then the whole joint probability will become 0, which is a dangerous claim! For transitions, we need smoothing because if transition probability is calculated to 0, then this tag bigram will never be predicted! Thus, I decided to use the Laplace Smoothing method where I increases the count of each transition and each emission entry by 1, and add |V| to the denominator. (Alpha is chosen to be 1). It turned out that this also makes the result worse.

After testing different k values for Add-K smoothing, I ended up realizing that I should only smooth the transitions probability and not the emission, as smoothing emission by any k-value will always increase my error rate. After more trials and experimenting, in the end, I ended up doing Add-K Smoothing on my transition probability and also absolute discounting of 2 on my transition probability. This improved my result for both error rate by word and by sentence!

Original unsmoothed bigram model, evaluated on ptb.22.tgs

```
$ ./tag_acc.pl ptb.22.tgs my.out
error rate by word:      0.0540917815389984 (2170 errors out of 40117)
error rate by sentence:  0.655882352941176 (1115 errors out of 1700)
```

Model with Add-k Smoothing + absolute discounting, evaluated on ptb.22.tgs

```
$ ./tag_acc.pl ptb.22.tgs my.out
error rate by word:      0.0538923648328639 (2162 errors out of 40117)
error rate by sentence:  0.655294117647059 (1114 errors out of 1700)
```

Part 4

Overall, I would say that the POS tagging for Japanese is the most accurate, follows by English and then Bulgarian. However, in terms of error rate by word, English is slightly better than Japanese and a lot better than Bulgarian. In terms of error rate by sentence, Japanese is a lot more accurate than English and Bulgarian, whereas English is slightly better than Bulgarian.

I believe that Japanese is most accurate in sentence accuracy because Japanese is a SOV (subject- object- verb) language. In Japanese, online sources say that verb always come at the end of clauses and sentences. Japanese parts of speech are usually marked with words called “particles” that follow the word they modify. Japanese is flexible in terms of word-order due to use of particles. Sentences, however, generally have the following structure:

Sentence Topic, Time, Location, Subject, Indirect Object, Direct Object, Verb.

The pattern that we see in Japanese's sentence structure shows that there is more consistency and correlation between two POS tags. For example, almost most of the times, the tag **sentence topic** will go before the tag **time**, in Japanese. Unlike languages like English and Bulgarian, in which one POS tag can be followed by many other possible POS tags, in Japanese – a POS tag generally is followed by one specific POS tag in most cases, due to their sentence structures.

English is more accurate than the other two languages in terms of error rate by word, because there is least ambiguity in English words as to which tag matches to an English word. For example, "eat" is always a verb, and there is no other possible POS tag that can match eat. Unlike other languages, where a word can be matched by various POS tags. However, because in English, a POS tag like VERB can be followed by various other possible POS tags(i.e. noun, adverb, adjective..etc), the error rate by sentence is higher.

Old Bulgarian had a system of seven cases, From Wikipedia, "**Bulgarian** verbs are the most complicated part of **Bulgarian grammar**. They are inflected for person, number and sometimes gender. They also have lexical aspect (perfective and imperfective), voice, nine tenses, five moods and six non-finite verbal forms.". From researching online, it became clear to me that Bulgarian is the most complex language among the three languages we're comparing. Even just a verb can be a person, a gender, instead of just actions like in English and Japanese. This makes it very hard for the tagger to be accurate in terms of error rate by words, because a word can simply be more than one POS tags, if not many. In terms of sentence rate, it is very inaccurate, as can be seen from the ~75% error rate by sentence. This is due to the fact that Bulgarian has complicated part of speech, such as its verbs that can represent plethora of things, which means that one word can be more than one POS tags and that the word following this current word may be of more than one POS tags as well.

Bulgarian bigram model

```
$ ./tag_acc.pl btb.test.tgs btb.out
error rate by word:      0.115942028985507 (688 errors out of 5934)
error rate by sentence:  0.751256281407035 (299 errors out of 398)
```

Bulgarian custom model

```
$ ./tag_acc.pl btb.test.tgs btb.out
error rate by word:      0.11577350859454 (687 errors out of 5934)
error rate by sentence:  0.748743718592965 (298 errors out of 398)
```

Japanese bigram model

```
$ ./tag_acc.pl jv.test.tgs jap.out
error rate by word:      0.0628611451584661 (359 errors out of 5711)
error rate by sentence:  0.136812411847673 (97 errors out of 709)
```

Japanese custom model

```
$ ./tag_acc.pl jv.test.tgs jap.out
error rate by word:      0.0632113465242514 (361 errors out of 5711)
error rate by sentence:  0.136812411847673 (97 errors out of 709)
```