1.

Train on small-train.xml and test on text_1.xml. What is the accuracy?

A: `93.80423090713518 %`

Train on small-train.xml and test on text_2.xml. What is the accuracy?

A: `84.44613876484043 %`

Examine the output of the tagger and see if any patterns emerge regarding where the tagger tends to make more errors.

A: There are a few words that have been incorrectly tagged. This is mainly due to the fact that the data used is small. When a certain amount of data is being fit into a small dataset, it's bound to lead to degradation of the result.

Can you explain the discrepancy in accuracy between them?

A: The difference is in the size of the file. Since these are two small files the number of words that will be correctly tagged will reduce as the size increases. By this we mean, as the number of words in the file increases. The reason big datasets work best is because these algorithms work well will optimally sized corpora. The algorithms work by calculating and depending on probability of occurrences. There will be a greater chance of catching the uncommon occurrences of words, thereby calculating the probabilities. This is not the same situation when it comes to small data.

The unigram tagger will tag the most commonly occurring word. Using the tagged data which is collected as a training set, it will be used to tag the test set. Hence the discrepancy.

What conclusions can you draw? What do you think might improve the tagger's performance?

A: There should be very few errors in the dataset set itself. For a basic method like unigram tagging, as mentioned above the data should have some bizarre words to be tagged and therefore increases the efficiency in obtaining probabilities of the word occurrences.

Conclusions that can be drawn are: accuracy decreases and the file size and content increases but the accuracy will be very good if the dataset is optimally large.

```
Unigram smallTest on test1:
Similarity = 13081 / 13945 = 93.80423090713518 %
------
Unigram smallTest on test2:
Similarity = 7753 / 9181 = 84.44613876484043 %
------
Unigram smallTest on test1:
Similarity = 7086 / 9181 = 77.18113495261954 %
-----
Unigram smallTest on test2:
Similarity = 10809 / 13945 = 77.51165292219433 %
```

2.
→ POSTag1.java has the modified code. The changes have been documented in the form of comments.

Basically a new hashmap has been added which is going to store the transition probabilities.

3.

```
Bigram small-Test on test1:
Similarity = 13228 / 13945 = 94.85837217640731 %
-------
Bigram small-Test on test2:
Similarity = 7874 / 9181 = 85.76407798714737 %
-------
Bigram medium-Test on test1:
Similarity = 11163 / 13945 = 80.05019720329868 %
-------
Bigram medium-Test on test2:
Similarity = 7249 / 9181 = 78.95654068184294 %
-------
Bigram big-Test on test1:
Similarity = 11566 / 13945 = 82.94012190749372 %
-------
Bigram big-Test on test1:
Similarity = 7572 / 9181 = 82.47467596122426 %
------------
```

The bigram tagger has been tested on all the three types of data: small, medium and large. The tagger is stable using the big dataset. The bigram tagger will use tag-to-tag transition probability. If you compare the unigram and bigram results, there is clearly the better performance using small datasets. The performance for the medium dataset is also better. When you compare all the bigram results, although the unigram results provide the best accuracy, I feel the big dataset result is most reliable since more the data that you come across more accurate the result.

The quality of the training set is most important.

4.

→ POSTagEnchancement.java has the modified code. The changes have been documented in the form of comments.

Basically the new modification is the addition of a new hashmap to POSTag1.java which stores the frequency of frequencies.

5.

Is the accuracy improved?

A: Unigram tagging on test 1 and test 2 respectively:

```
Similarity = 12649 / 13945 = 90.70634636070277 %
---------
Similarity = 7499 / 9181 = 81.67955560396472 %
---------
```

Therefore, According to the result there seems to be a drop in the accuracy values.

Does your chosen enhancement reduce any of the types of errors that you previously noted for the base tagger? If so, which and why?

A: There seems to be a decrease due to production of errors in tag-to-word probability. Appropriate Decision mechanisms like a rule-based theory can help increase the accuracy.

How do you think you might improve the tagger even more?

A: Introduction of machine learning techniques like Support Vector Machines can help increase the accuracies. Singular Value Decomposition that involves vectorization will also help in increasing the accuracies. This I can say because these concepts help in determining a hidden factor which helps providing better probability values, in turn better accuracies.