***Approach**:*

The main idea to correct the erred file is to use POS tagging. I have used NLTK library for POS tagging. The "*tagger.py*" program takes a single argument which is the input file to be corrected. It then POS tags each individual line using NLTK POS tagger and then the whole input is read word by word looking for the given 10 words. For each relevant found word in the file, it generates a file to be passed to MegaM to predict which word should have been here according to our models. The models are generated for each pair of words, in our case it will be five model files where pairs will become the class itself. The features taken are only the previous word tag and the next word tag with the word as a class. So prediction using the model file will give us the predicted word in place. For e.g., if in the erred file the word is "*to*" or "*too*" (there might be capital letters also), then our tagger will set the appropriate model file for it and prediction using this will give us the predicted word "to" or "too" accordingly.

***Fscore** on provided Development data:*

Overall FScore is based on total number of changes made. I am only looking at the total number of changes which my "*tagger.py*" has made. There are 7398 words need to be corrected. Here are the results:

Precision: 0.924148606811
Recall: 0.968369829684
FSCore: 0.945742574257

***Optimization**:*

We can try different sets of features to generate the model files and test it. Currently, only two features are generating the model files. I tried one more set of features which had current word's tag and, its previous two and next two word's tag, but it wasn't providing the better results.