NLP Homework on NLTK

Vibha Belavadi

Net-Id: vxb141530

Problem Statement:

The problem statement required us to perform sentiment analysis by applying Maximum Entropy Classification to movies review data and to observe the affect on accuracy by the discriminating features of stop words, punctuations, lemmatization and also changing the amount of training data to be fed. We were also required to perform analysis on the unbalanced collection – changing proportions of poisitive and negative samples in training data.

The following case studies were proposed:

Case Study I: Maximum entropy classification on a) RawData, b) With stop words, c) without punctuation, d) with lemmatization, for all the words assuming equal proportions of positive and negative examples

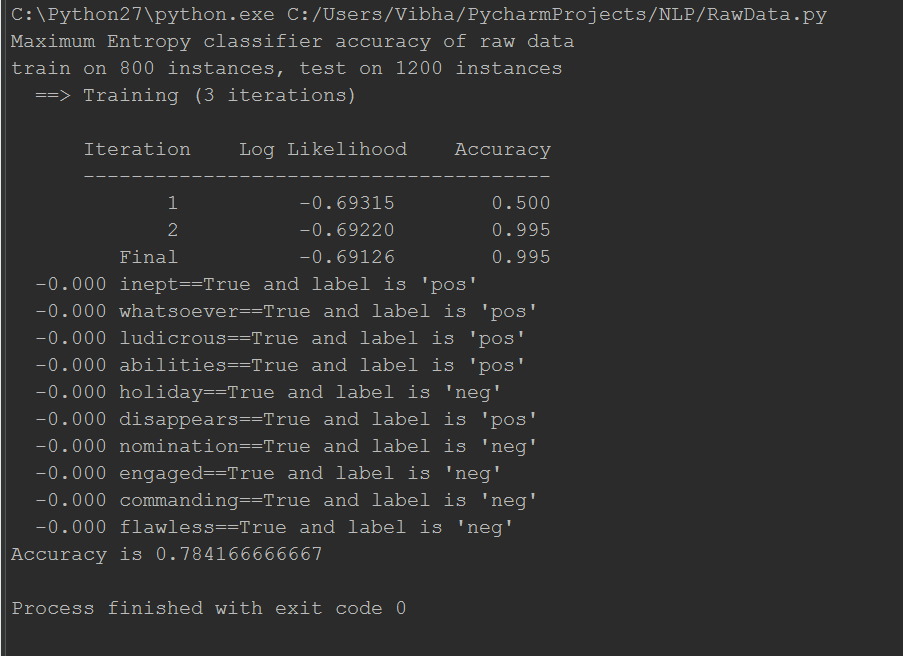
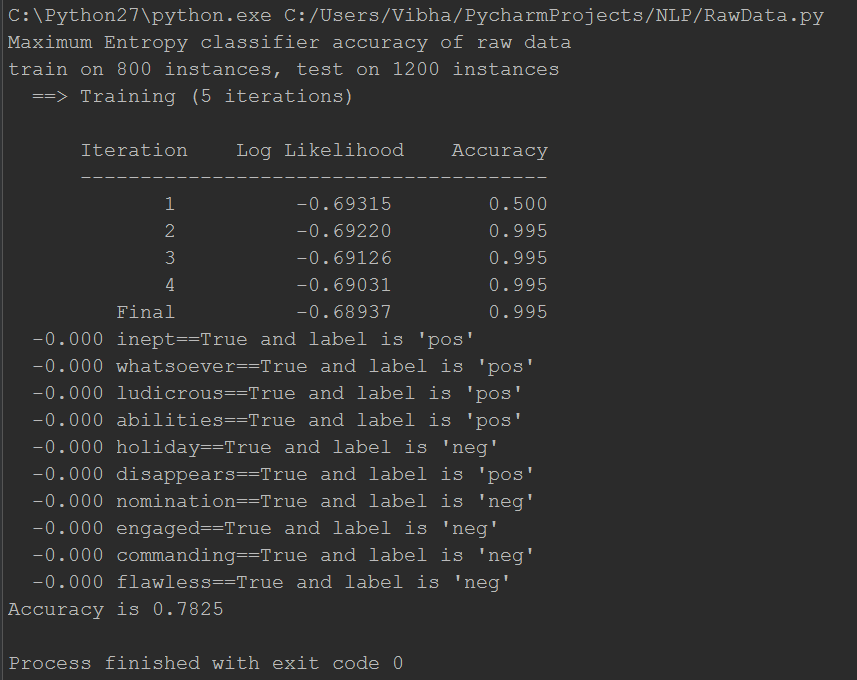
Case Study II: Maximum entropy classification on a) RawData, b) With stop words, c) without punctuation, d) with lemmatization, for top 500 words assuming equal proportions of positive and negative examples

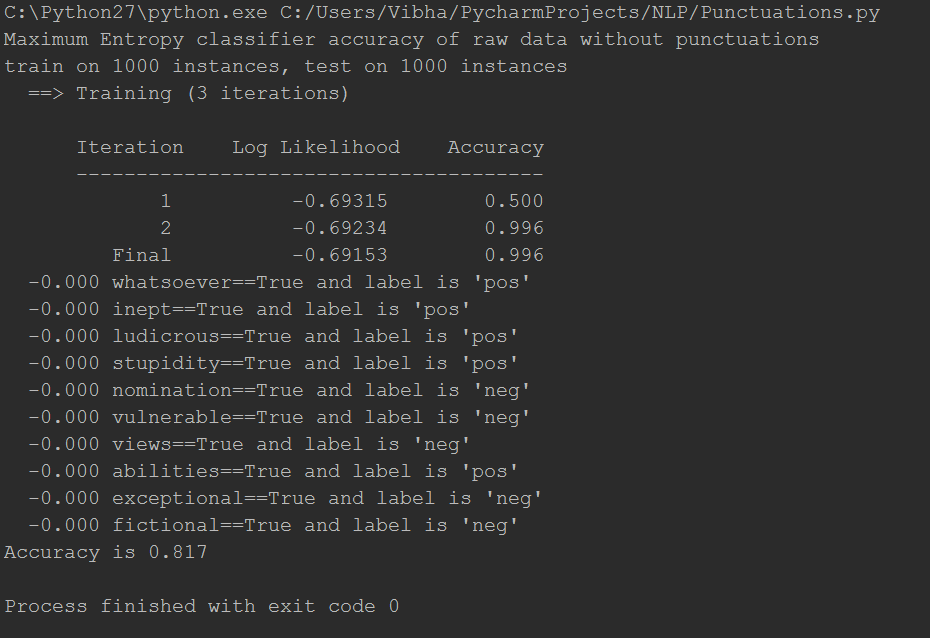
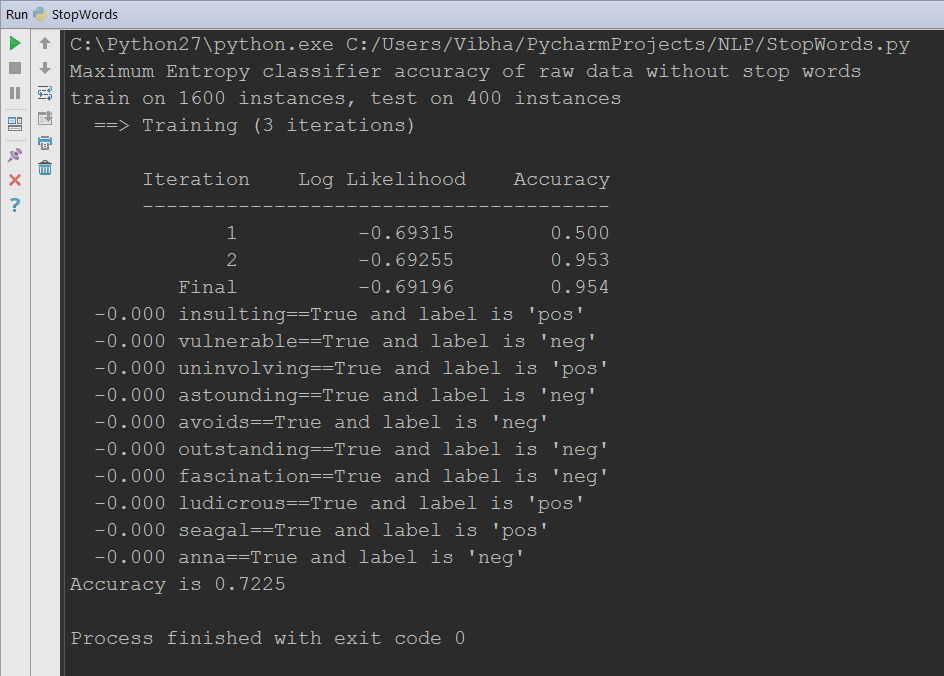
Case Study III: Maximum entropy classification on a) RawData, b) With stop words, c) without punctuation, d) with lemmatization, for top 1000 words assuming equal proportions of positive and negative examples

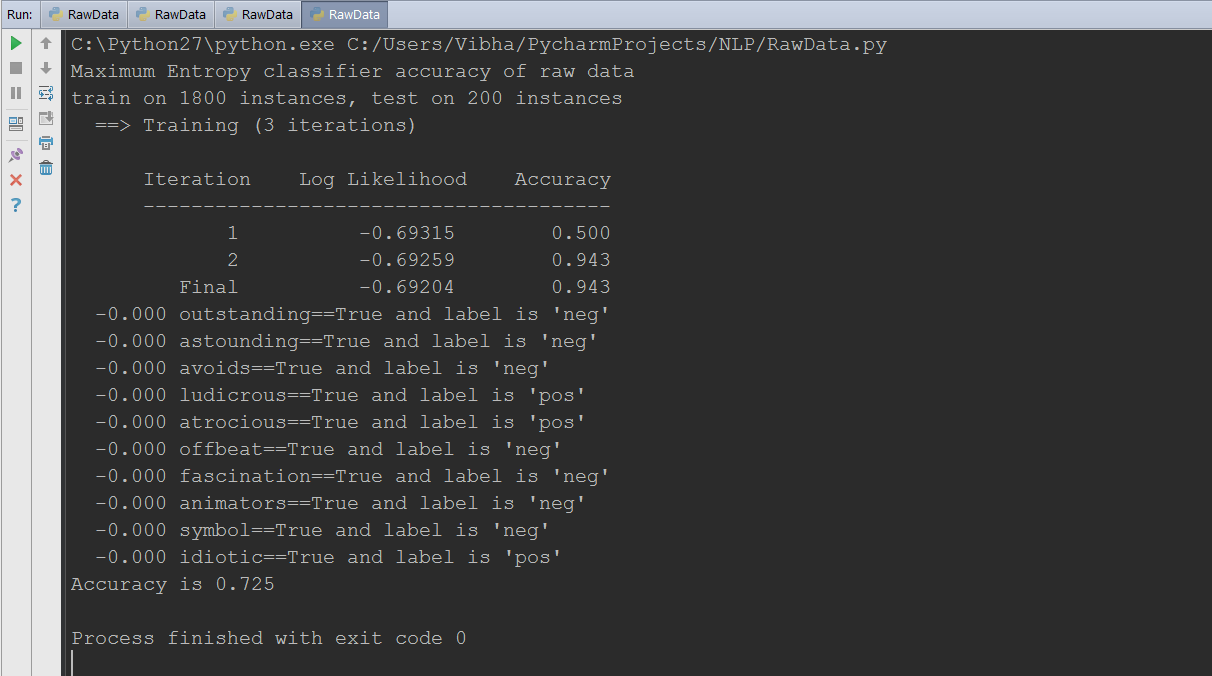
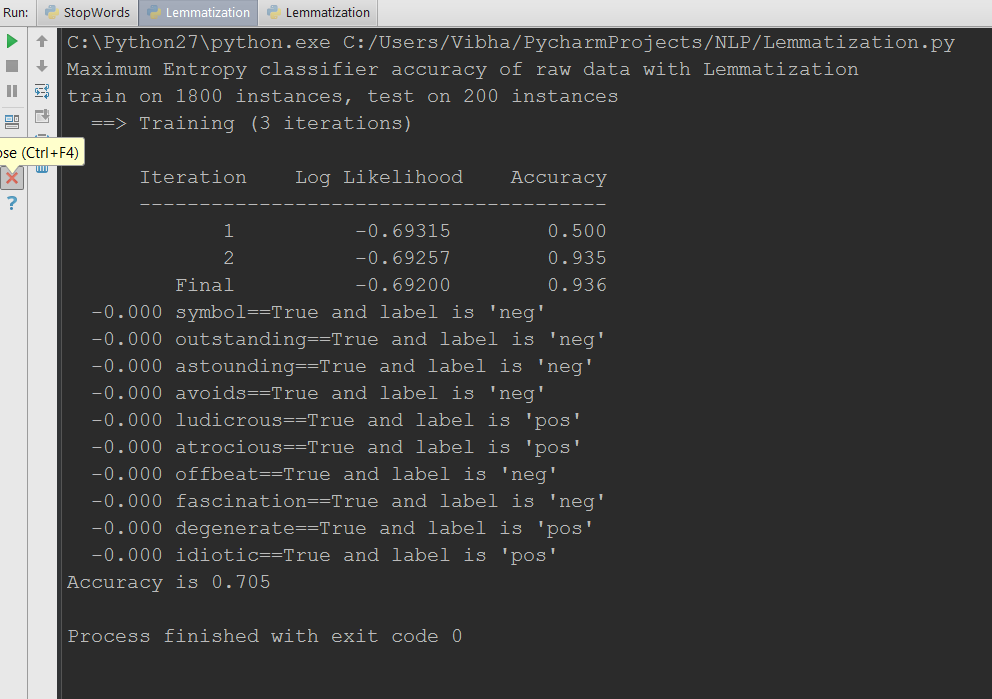
Case Study IV: Maximum entropy classification on a) RawData, b) With stop words, c) without punctuation, d) with lemmatization for all the words assuming unequal proportions of negative and positive examples

Case Study V: Maximum entropy classification on a) RawData, b) With stop words, c) without punctuation, d) with lemmatization for all the words assuming only negative examples

Snapshots of the NLTK program running in pycharm:

   
The effect of number of iterations on the result



Some snapshots of the program getting executed

Experimental Results:

Case Study I: The Accuracy table considering all word with 3 iterations assuming equal positive and negative example proportions:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 25% training data | 50% training data | 60% training data | 70% training data | 80% training data | 90% training data |
| Raw Data | 0.813 | 0.817 | 0.76 | 0.723 | 0.727 | 0.725 |
| Without Stop Words | 0.813 | 0.812 | 0.758 | 0.722 | 0.722 | 0.725 |
| Without Punctuations | 0.813 | 0.817 | 0.7575 | 0.7167 | 0.7225 | 0.725 |
| With Lemmatization | 0.802 | 0.813 | 0.76 | 0.7167 | 0.73 | 0.705 |

Case Study II: Accuracy table considering only the top 500 words with 3 iterations assuming equal positive and negative example proportions:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Train Set | 25% training data | 50% training data | 60% training data | 70% training data | 80% training data | 90% training data |
| Raw Data | 0.542 | 0.556 | 0.566 | 0.568 | 0.57 | 0.58 |
| Without Stop Words | 0.556 | 0.566 | 0.555 | 0.565 | 0.56 | 0.585 |
| Without Punctuations | 0.545 | 0.556 | 0.566 | 0.572 | 0.57 | 0.585 |
| With Lemmatization | 0.802 | 0.814 | 0.76 | 0.72 | 0.74 | 0.71 |

Case Study III: Accuracy table considering only the top 1000 words with 3 iterations assuming unequal positive and negative example proportions:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Train Set | 25% training data | 50% training data | 60% training data | 70% training data | 80% training data | 90% training data |
| Raw Data | 0.585 | 0.605 | 0.5975 | 0.61 | 0.6275 | 0.64 |
| Without Stop Words | 0.584 | 0.608 | 0.60125 | 0.6117 | 0.6225 | 0.64 |
| Without Punctuations | 0.583 | 0.604 | 0.5975 | 0.6117 | 0.6275 | 0.64 |
| With Lemmatization | 0.802 | 0.814 | 0.76 | 0.72 | 0.74 | 0.71 |

Case Study IV: Accuracy table considering all the words with 3 iterations assuming unequal proportions of negative and positive examples for a training model:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Train Set | 100% negative samples | 10% positive examples, 90% negative examples | 20% positive examples,  80% negative examples | 25% positive examples, 75% negative examples | 40% positive examples, 60% negative examples | 60% positive examples, 40% negative examples | 75% positive examples, 25% negative examples | 80% positive examples, 20% negative examples | 90% positive examples, 10% negative examples | 100% positive samples |
| Raw Data | 0 | 0.1 | 0.2 | 0.25 | 0.4 | 0.6 | 0.75 | 0.8 | 0.9 | 1 |
| Without Stop Words | 0 | 0.1 | 0.2 | 0.25 | 0.4 | 0.6 | 0.75 | 0.8 | 0.9 | 1 |
| Without Punctuations | 0 | 0.1 | 0.2 | 0.25 | 0.4 | 0.6 | 0.75 | 0.8 | 0.9 | 1 |
| With Lemmatization | 0 | 0.1 | 0.2 | 0.25 | 0.4 | 0.6 | 0.75 | 0.8 | 0.9 | 1 |

Graphical Representation of the experimental results:

Observations:

* Number of iterations is taken 3 since there is not much noticeable change in the experimental results if the number of iterations is taken to be more than 3.
* Using numpy’s array is highly recommended as it packs the data to be sent to nltk’s Maxent algorithm efficiently and gives the result quickly.
* The accuracy across all the observation is highest when 25% of the data is used as training data irrespective of the case study.
* The accuracy rate keeps on decreasing as we increase the training data across all the case studies
* Lemmatization works best if we are calculating the accuracy for top N words in the corpus (top 500 and top 1000). However, if we apply Lemmatization normally across all the words in the corpus, its performance is the worst.
* The accuracy for all the other discriminating features, except Lemmatization, decreases when Maximum entropy is applied for top 500 words or top 1000 words, as compared to applying the algorithm on all the words in the corpus.
* For varying values of positive and negative examples, the accuracy depends on the ratio of positive and negative examples, and this observation is irrespective of the case study or the discriminating features.

Source Code Files attached in the project:

* RawData.py
* RawDataTopWords.py
* StopWords.py
* StopWordsTopWords.py
* Punctuations.py
* PunctuationsTopWords.py
* Lemmatization.py
* LemmatizationTopWords.py