PROJECT REPORT

Author : Zhekai Jin (Scott)

Course : ECE 467 Natural Language Processing

Professor : Carl Sable

Description

A text categorization using **Naive Bayes** method with Laplacian Smoothing. [Laplacian Smoothing to aviod zeroing out] A classifier was trained using the training set, and result in a learned classifier. Then this learned classifier is used to classify new documents.

Dependency

- Python3
- NLTK
  - Tokenize
  - PorterStemmer

Usage

```
python3 scott.py [k]
```

k stands for the Laplacian Smoothing Factor Then answer the question as prompted.

Assumptions

- [Bag of Words] assumption: Assume the position of the words in the document doesn't matter.
- Conditional Independence: Assume the feature probabilities P(xi l cj) are independent given the class c.

Workflow

Given:

- an input document
- an labeled training sets contains category that this document belongs to

Train:

- obtain the count of total documents. N
- obtain the count of documents that have been mapped to this category Nc.
- if we encounter new words in this document, add them to our vocabulary, and update our vocabulary size |V|.
- update count(w, c) => the frequency with which each word in the document has been mapped to this category.
- update count (c) => the total count of all words that have been mapped to this class.

Categorize:

- Calculate P(c) = Nc / N (prior probablity)
- Calculate P(w | c) = [count(w, c) + k] / [count(c) + k * |V|]
- Multiply each P(w | c) for each word w in the new document, then multiply by P(c), and the result is the probability that this document belongs to this class.
- Computing this product with standard floating point will soon get into problems of underflow. Instead, the logarithm is used.
- And log operation also ease the mutiplication to addition.

Running Time

- Offline (learning the classifier): O(max(size of the corpus, number of different words * number of categories)); in practice, this is almost always O(size of the corpus).
- Online (applying the classifier to a document): O(length of document number of categories).

Robust under noise:

Small changes in the texts or inclusion of a small number of anomolous documents lead to small changes in the probability estimates. [tested]

Tokenization:

- Tokenization used the NLTK's `word_tokenize` for both training and test result.

- This Tokenizers divide strings into lists of substrings. For example, tokenizers can be used to find the words and punctuation in a string. To be noted, this tokenizer is based on regular expression. Punctuation is then removed from the probablity calculation for the testing file to avoid noise.

Performance Testing

- The testing was performed with 30% of the provided training set taken out to be test set and the traning was perfromed on the rest of the system to avoid overfitting on the first corpus.
- Cross validation was also performed on three corpus:

  Optional:

  - K-fold cross-validation was performed to give a decent accuracy. More emphasis on accuray was put on than recall over the precision / recall tradeoff.
  - confusion matrix was seen by using provided testing perl script with the F1 score.
  - roc*auc*score was calculated to give a high metrix over 90%. [auc ==> area under the curve]

Optional Parameter

- case sensitivity was taken care by stemming, lemmatization could have been performed to avoid fuzzy matching, but little improvement was seen since the same stemmer was used for both the new document and the training set.
- Laplacian smoothing factor is default to 0.06 with trial and error, and is an optional argument for the program.
- Stemming and smoothing factor's choice does matter. k = 1 generally only have about 80% accuracy and stemming imporve the accuracy by 5%.

Future Improvement

- Weighting:

  - If we have a sentence that contains a title word, we can upweight the sentence (multiply all the words in it by 2 or 3 for example), or we can upweight the title word itself (multiply it by a constant).

- Upweighting: give more weight to words occur more times.

- Feature selection:

  - we can look for specific words in the document that are good indicators of a particular class, and drop the other words [semantically empty]. Note punctuations was already taken out.