Maral Dicle Maral
03.01.2022

CMPE 493 INFORMATION RETRIEVAL

ASSIGNMENT 2: A Simple Search System for Phrase and Free Text Queries

In this assignment Multinomial Naive Bayes (NB) and k Nearest Neighbor (kNN) algorithms for text classification is implemented. Reuters data was utilized for training and test sets.

Classification with kNN and NB:

```
python3 classify.py ./reuters21578/ ./stopwords.txt
```

Assignment consists of 3 main steps:

1- Data Preprocessing
2- Test and Training Set Creation
3- Naïve Bayes Classification
4- kNN Classification
5- Evaluation
6- Statistical significance

1- Data Preprocessing

Reuters data was read via an HTML file processing library, beautifulSoup. As the documents are read, they are trimmed from punctuations according to the string.punctuation list and cleaned from stop words specified by the user. Latin-1 encoding was used for the documents. Multiprocessing which creates a process for each document was used to speed up the data reading at data preprocessing step. Final vocabulary consists of 27045 words in my design.

2- Top 10 Classes, Test and Training Set

Number of documents in the test set is 2532 while it is 6454 for training set. I excluded the empty and not labeled documents from the sets. (The documents that have empty BODY, TITLE and TOPICS value and the ones labeled as REUTERS TOPICS="NO" are excluded.)

Maral Dicle Maral

03.01.2022

The top ten classes in the documents are: 'earn', 'acq', 'money-fx', 'grain', 'crude', 'trade', 'interest', 'wheat', 'ship', 'corn'.

Afterwards I separated the data into test and training set according to the LEWISSPLIT label. For **test set**, there are 'earn': 1080, 'acq': 718, 'money-fx': 179, 'grain': 148, 'crude': 186, 'interest': 131, 'trade': 116, 'wheat': 71, 'ship': 87, 'corn': 56 documents. For **training set,** there are 'earn': 2861, 'acq': 1648, 'money-fx': 534, 'crude': 385, 'grain': 428, 'trade': 367, 'interest': 345, 'ship': 191, 'wheat': 211, 'corn': 180 documents. Number of multi-labeled documents in the test set is 211, while it is 613 for training set.

3- Naïve Bayes Classification

For naïve bayes, I used word counts as features for the classification. I calculated P_cj and P_wk_cj values from the training set as described in the class. Class probabilities (P_cj) are kept in a dictionary of classes and P_wk_cj are kept as a bag of words in a dictionary structure. I applied add-one smoothing while computing the probabilities, therefore I also kept the number of words in the documents and classes. While predicting the test case classes, I used P_wk_cj and P_cj probabilities and the necessary smoothing terms.

In order to get better predictions, I utilized the naïve bayes log version that was described in the class:

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}}[\log P(c_j) + \sum_{i \in positions} \log P(x_i \mid c_j)]$$

4- K Nearest Neighbors Classification

For kNN classification, I first applied feature selection based on term frequencies of the words in the training set, I utilized 1 percent of the features due to the efficiency issues on my computer. (Normally I should have left out only the words with counts less than 3.)

After feature selection, I created the tf-idf vectors of both training and test set according to the selected features.

Later on, I created a development set from the training set in order to find the best performing k value (in terms of F measure). I run the kNN and accuracy_measure algorithms on the development set for k values ranging from 7 to 30 and calculated the best performing k for my development set as 10. Therefore, I set the k for the test set evaluation as 10.

Finally, I merged the development set with the rest of the training set and run the kNN algorithm for the all the training set and test set, and predict the test set classes.

Maral Dicle Maral

03.01.2022

kNN algorithm finds the k closest neighbors to the given test point and evaluates its class accordingly. I applied cosine similarity on the tf-idf vectors to find the most k similar neighbors to the test documents. Because we already implemented cosine similarity algorithm in the second assignment, and because of the efficiency issues, I utilized numpy library for cosine similarity calculation. I used np.norm() and np.dot() functions from the numpy library instead of doing calculations cell by cell in the matrix.

5- Evaluation

Below precision and recall values for naïve bayes and kNN classification suggests the naïve bayes has overall better performance. The macro F1 score for naïve bayes is 0.7403 while it is 0.6626 for kNN.

Although I use the best performing k value for kNN selection, I used a small feature set (1 percent of the top term-frequency terms) for kNN while I did not apply a feature selection for NB. This might be the reason why naïve bayes performs better compared to kNN.

Accuracy Scores for Naive Bayes are:

Macro Precision: 0.842 Macro Recall: 0.6592 Macro F1 Score: 0.7395

Micro Precision: 0.8752 Micro Recall: 0.9581

Accuracy Scores for kNN are:

Macro Precision: 0.7786 Macro Recall: 0.5594 Macro F1 Score: 0.651

Micro Precision: 0.7994 Micro Recall: 0.8752

P Value Evaluation:

In order to find the significance of this accuracy difference between two algorithms, I utilized the randomization test. The null hypothesis was used classification algorithms (naive bayes and kNN) are not different. Therefore, I swapped the predicted results (y_pred) of the two algorithms with 0.5 possibility and find the macro F1 accuracy measure for both swapped y_pred lists for 10 times and found the p value as 0.0909 (>0.05), which is not scientifically significant. Therefore, I concluded that the difference between the two algorithms is by chance. **(Figure 1)**

Maral Dicle Maral

03.01.2022

```
Last login: Tue Jan  4 15:29:29 on ttys001
maral@192 ~ % cd /Users/maral/Documents/Classes/INFO_RETRIEVAL/Assignment3_Classification_NB_KNN
maral@192 Assignment3_Classification_NB_KNN % python3 classify.py ./reuters21578/ ./stopwords.txt
#### ACCURACY SCORES FOR NAIVE BAYES ####
Macro Precision:  0.842    Macro Recall:  0.6592   Macro F1 Score:  0.7395
Micro Precision:  0.8752    Micro Recall:  0.9581

#### ACCURACY SCORES FOR KNN ####
Macro Precision:  0.7786    Macro Recall:  0.5594   Macro F1 Score:  0.651
Micro Precision:  0.7994    Micro Recall:  0.8752

P value 0.0909
- Classification process ends in 54.497910 seconds -
maral@192 Assignment3_Classification_NB_KNN %
```

Figure 1 (Accuracy Measurements for Naïve Bayes and kNN, p Value for the randomization test)