FACULTY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER ENGINEERING

Artificial Intelligence - ENCS434

Automatic Spam Review Detection

Prepared by: Rawan Yassin 1182224 Alaa Zuhd 1180865
Instructor: Dr. Aziz Qaroush

Section: 1

BIRZEIT
January 25, 2021

1 ABSTRACT

The aim of the project is to become familiar with machine learning techniques. This project targeted building an automatic spam review detector, after pre-processing the data, extracting extra features and building different classifiers. An automatic spam detector is a popular and challenging application in the artificial intelligence field, its importance comes from the fact that most people nowadays use online shopping to make purchase decisions based on other reviews. Helpful and Non-fake reviews become very important so that customers are provided with true facts. In this project we have used python programming language with the help of the Scikit-Learn, NLTK and matplotlib libraries. We have implemented four main models : Decision tree classifier, Neural Network classifier, Naive Bayse classifier, and Random Forest classifier.

Contents

2 INTRODUCTION

2.1 Machine learning

Machine learning is branch of artificial intelligent that concerns how the computer can learn and adapt new circumstances, there are many learning techniques based on the desired outcome from the technique and the input available at the training process. Machine learning most well known techniques are

1. **Supervised learning :** when an agent tries to find the function that matches one of the example from the data set, for each input in the data set there exist an specified output class ,this type of algorithms are used for classification problems e.g. Decision Tree, Artificial neural network.

2. **Unsupervised learning :** when an agent tries to learn from patterns without a specified output e.g. clustering.

3. **Reinforcement learning :** in this type the agent doesn't know the exact output for an input, but it receives feedback on the desirability of its behavior. This feedback can become from the outside environment or the agent itself.

2.2 Automatic Spam Review Detection

Detection of spam reviews have become one of the major applications on machine learning, due to the wide range of the online reviews about almost all the different products. Using reviews property on the online sites, spammers are trying to write fake reviews about any product, which can be positive or negative, depending on their goals. For-example some of them write spam reviews in-order to increase the general rating for a specific product, while others may be totally thinking the opposite. Due to all the mentioned reasons, we observe that reviews can affect the general rating of a product, and hence increasing the demand for that product, or the opposite. From here and since the science seeks to enhance the positive side of technology and development, there are thousands of researches that tried to make an automatic spam review detector.

Spam Reviews can be one of the following categories according to Jindal et. al. :

1. **Fake Reviews :** Providing fake reviews about the product in-order to demote and promote that product.

2. **On Brands Reviews :** This kind of reviews contains information about the brand, or the cost of its different products. Even if this review is a true fact but it is considered a spam review since it doesn't provide a useful information about this specific product.

3. **Non-Related Reviews :** this kind of review can't provide any help-full information about the product, it may be on another product or any other thing non-related to the intend review.

2.3 Why Scikit-Learn Library

The Sklearn library is a collection of machine learning algorithms for data mining tasks. Sklearn contains tools for data pre-processing, classification, regression, clustering, and association rules. It is also well-suited for developing new machine learning schemes. Sklearn supports several standard data mining tasks, more specifically, data pre-processing, classification, and feature selection.

2.4 Data Set

The data set used for training the differed classifiers was part of the Yelp dataset. The Yelp dataset is a subset of Yelp's businesses for use in educational, and academic purposes.

3 METHODOLOGY

In this section, we will briefly summarize all the steps we have taken in order to complete our spam review detector.
We have followed the following steps:

1. We have first read the raw data and implemented all the pre-processing mentioned below techniques.
2. We have then implemented all the mentioned below feature extraction techniques.
3. All pre-processed with feature extracted data were rewritten in a csv file.
4. We have then decided what model to train, made sure that we are dealing with balanced data and then trained the model.
5. We have stored all trained models for future use and new test samples.

Implementing all the above steps required dealing with different python libraries and learning many different techniques to achieve the goal. We have also had to deal with Google Colaboratory. Using it provided us with the needed memory to do the feature extraction on more than 30000 data entries. It has also allowed us to use the computing power of the Google servers instead of our own machine. Running python scripts requires often a lot of computing power and can take time. By running scripts in the cloud, you don't need to worry. Following is a detailed description of each performed step:

3.1 Step 1: Pre-processing

A pre-processing (cleaning and organizing the raw data) was done before building the trained models. Pre-processing is an important step that enhances the quality of data. Our pre-processing procedure consisted of the following steps implemented in each raw entry :

1. **Handling Spaces in Review Content:** This step aimed to ensure that only a space exists between a word and another. It is important as counting the words in each content correctly helps in classifying spam from ham reviews. Spammers attempts to write less words than non-spammers.
2. **Removing Special Characters from Review Content:** This step would help us when considering the stemming process and finding the similarity after having each word count. In this step all non-alphabetical or numerical characters were removed.
3. **Converting the words in Review Content to Lower Case:** As different reviews may have same words but one written with upper case letters and the other with small, this step ensures that all review contents are in lower case.

4. **Stop-word Removal of Review Content:** This pre-processing step is needed mainly to ease the process of stemming and calculating the similarity afterwords. It removes the stop-words such as (the, are, is, am..) from the review content.

5. **Stemming the Review Content:** Stemming is the process of reducing inflected or derived words to their word stem, or root. This step is crucial as we have implemented in the feature extraction process a cosine similarity technique, which is based of the term frequency, and so considering (smiling, smile, smiled) as the same term would have no negative effects but will surely decrease the space needed to do the term frequency.

Note: raw data entries with missing features were discarded, as there is a huge number of data entries and we only needed 0.25% of them to be able to run on our machines and train our models.

3.2 Step 2: Features Extraction

Raw Data includes redundant and some non-informative features, and as our goal is to build a good classifier, we need to get a complete and meaningful set of the needed features. Feature Extraction aims to create and deduce new features from existing ones. In this project, our feature extraction process has all the following:

1. **Word Count of each Review Content:** Counting the number of words in a review is a benefit. Spammers tend to write less words than non-spammers, and according to Yelp studies, it was found that about 80% of spammers are bounded by 135 words in average review length which is quite short as compared to non-spammers.

2. **Percentage of Capitalized Words in each Review Content:** Before converting the review content to lower case, as mentioned in the pre-processing step, this feature was extracted due to its importance. According to some research, spammers tend to write with capitalized letters so as to grab the attention of other readers.

3. **Whether each Review Content Has URL or Not:** This feature gives us a hint whether this review is a spam or not, reviews with URLs have higher probability of being spams.

4. **Average Number of Reviews for a Reviewer Per Day:** According to Yelp, writing many reviews in a day is abnormal. In order to be able to use this feature in a classifier, we first have to extract it. We have decided to loop through the data, calculate the number of reviews for a specific reviewer ID as well as the distinct days in which he/she has written these reviews, then an average for this specific reviewer ID was calculated. Adding this feature to each raw data has had a good effect of the accuracy of our classifiers.

5. **Average Rating Per Reviewer ID:** About 15% of the spammers have less than 80% of their reviews as positive, for example, a majority 85% of spammers rated more than 80% of their reviews as 4-5. Based on the previous assumption, the need for an average rating for each reviewer ID arises, and so extracted.

6. **Calculating the Similarity Between Different Reviews of the Same Reviewer:** Achieving this extraction was done in two step:

   (a) Calculating the TF-IDF: The TF-IDF stands for Term Frequency - Inverse Document Frequency. The term frequency measures the number of times a term (word) occurs in a document, the document in our project consists of all the reviews read from raw data after pre processing stage is done. The inverse document frequency aims to weight down the effects of too frequently occurring terms as well as weighting up the effects of less frequently occurring terms due to the fact that the terms occurring less in the document can be more relevant. As a result the TF-IDF would mean that for each term in the document multiply its term frequency with its IDF for each separate document.

   (b) Measuring the cosine similarity: After using TF-IDF as a tool to convert text into numbers so that it can be represented by a vector, we were able to calculate the cosine similarity. The cosine similarity is a similarity metric that depends on dot product of two given vectors and dividing that by the product of their norms.

3.3 Step 3: Handling Unbalanced Data Set

A dataset without equal class distributions, or with extreme disproportion among the number of samples per class is said to be imbalanced. The class with abundant examples is called the major or majority class, whereas the class with few examples is called the minor or minority class. Imbalanced dataset pose a challenge for training a classifier. Almost all of the classification algorithms in machine learning are based on the assumption that an equal number of examples is presented for each class. There are many approaches that could be implemented to handle this situation, below is the approach we have depended on: **Combination of SMOTE and Tomek Links Undersampling.** We have considered this approach due to the fact that a combination of SMOTE and under-sampling performs better than plain under-sampling or plain over-sampling.
SMOTE is an oversampling method (generating new dataset samples) that synthesizes new plausible examples in the majority class. **Tomek Links** refers to a method for identifying pairs of nearest neighbors in a dataset that have different classes. Instances that are in Tomek Links are either boundary instances or noisy instances as only boundary instances and noisy instances will have nearest neighbors, which are from the opposite class. What Tomek links approach do is removing one or both of the examples in these pairs and this has the effect

of making the decision boundary in the training dataset less noisy or ambiguous. Using the imblearn library helps achieve the above mentioned technique using the following two lines of smt = SMOTETomek(random_state=42)
X_smt, y_smt = smt.fit_resample(x, y) , x and y represents the data set, x for the features and y for the class.

In addition to the mentioned approach above, we have tried to read balanced data from the file using two counters, one for positive filtered outputs and another for negative. This technique worked well when we only needed to read up to 20000 records, in which we were sure we were reading an equal number of samples from each class.

3.4 Step 4: Models Training

In this project, we have implemented and trained four different classifiers as follows:

1. **A Decision Tree:** It is a supervised machine learning method used for classification and regression. It continuously splits data based on certain parameters. It aims to create a model capable of predicting the targeted variable. It uses simple decision rules inferred from the data features. Implementing this tree was done using Scikit-learn after splitting the data, an 80% for training the model and a 20% for testing. Then an object of the classifier was created, given the training data, each as a feature vector with the class output.

2. **A Neural Network:** It is a supervised machine learning method that uses a variety of algorithms that aims to find relationships in a set of data through a process that mimics the way the human brain operates. Neural network adapts to changing input, generating the best possible result.

3. **A Naive Bayes classifier:** It is a probabilistic machine learning model that's used for classification task. It is based on the Bayes theorem.

4. **A Random Forest classifier:** is an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees.

4 TESTING AND RESULTS

In this section we will depend on a 20000 processed data entries stored on a (Processed20000.csv) file.
The final features that we have decided to depend on are:

1. **reviewerID:** each reviewer has a unique numerical ID.

2. **AvgRating:** the average rating of all the reviews done by the reviewer with the specified reviewer ID.
3. **AvgReviewPerDay:** the average number of reviews per day by the specified reviewer ID.
4. **usefulCount:** number of times this review marked as useful.
5. **firstCount:** the number of times this reviewer is the first one that make review
6. **reviewContentLength:** the number of words calculated from the review content after handling the spaces.
7. **PerCapWords:** the number of capitalized words in the review content divided by the total number of words in the review content.
8. **hasURL:** whether a review content contains URL or not.
9. **similarity:** the maximum calculated similarity of this reviewers' different reviews.
10. **filtered:** is the label, 0 for non-spam, 1 for spam.

Before going in depth with the different outputs, we need to consider the following metrics:

1. **Precision:** is the number of correctly classified positive examples divided by the total number of examples that are classified as positive.
2. **Recall:** is the number of correctly classified positive examples divided by the total number of actual positive examples in the test set.
3. **Accuracy:** is the number of correct classification divided by the total number of test cases.
4. **F1 score:** is the harmonic mean of precision and recall, and it tends to closer the smaller of the two.

All of the mentioned will be presented in a classification report, which is a visualizer that displays the precision, recall, F1, and support scores for the model.

Following is the classification report for the trained decision tree: We notice that the accuracy is 68% which is close enough to the one found in the "What Yelp Fake Review Filter Might Be Doing?" research. Having this accuracy with almost the same recall which we are considering mainly indicates that we are going the right way. An improvement by 1 in the accuracy was introduced when using the 20000 entries file -which we are showing below- over the 10000 entries file.

```
Choose The number of samples in the DataSet :
10000 Samples
20000 Samples
30000 Samples
Enter the number of Samples : 20000
Decision Tree
time needed = 0.061864376068115234s
              precision    recall  f1-score   support

           0       0.68      0.69      0.68      1572
           1       0.68      0.68      0.68      1539

    accuracy                           0.68      3111
   macro avg       0.68      0.68      0.68      3111
weighted avg       0.68      0.68      0.68      3111
```

Figure 1: Decision Tree Report

Following is the classification report for the Neural Network: An improvement in the spam recall was introduced when considering the 30000 entries file, but all other metrics remained unchanged. When training the neural network model, a scaling for feature introduced a huge improvement. Below is the classification report, after applying the scaling.
Feature Scaling: is a technique to standardize the independent features present in the data in a fixed range. It is performed to handle highly varying magnitudes or values.

```
Choose The number of samples in the DataSet :
10000 Samples
20000 Samples
30000 Samples
Enter the number of Samples : 20000
Neural network
time needed = 3.2760825157165527s
              precision    recall  f1-score   support

           0       0.61      0.81      0.70      1554
           1       0.72      0.49      0.58      1557

    accuracy                           0.65      3111
   macro avg       0.67      0.65      0.64      3111
weighted avg       0.67      0.65      0.64      3111
```

Figure 2: Neural Network Report

Following is the classification report for the Naive Bayes model: Before training the model, we had handled the unbalanced data and did feature scaling. Following is the report based on the 30000 entries file.

```
Choose The number of samples in the DataSet :
10000 Samples
20000 Samples
30000 Samples
Enter the number of Samples : 30000
Naive Bayes
time needed = 0.008972644805908203s
              precision    recall  f1-score   support

           0       0.61      0.59      0.60      4226
           1       0.60      0.61      0.60      4177

    accuracy                           0.60      8403
   macro avg       0.60      0.60      0.60      8403
weighted avg       0.60      0.60      0.60      8403
```

Figure 3: Naive Bayes Report

Following is the classification report for the Random Forest model: As Random forest is an improvement of the decision tree we notice an improvement in both the recall and precision compared to these values obtained from the decision tree. We have used the 20000 entries file.

```
Choose The number of samples in the DataSet :
10000 Samples
20000 Samples
30000 Samples
Enter the number of Samples : 20000
Random Forest
time needed = 0.35005950927734375s
              precision    recall  f1-score   support

           0       0.70      0.83      0.76      1587
           1       0.78      0.63      0.70      1524

    accuracy                           0.73      3111
   macro avg       0.74      0.73      0.73      3111
weighted avg       0.74      0.73      0.73      3111
```

Figure 4: Random Forest Report

As many variations can be considered and have many predicted effects, this table tries to summarize as many cases as possible. It is built using the processed 30000 unbalanced file, one without handling unbalanced data and the other with, for all the mentioned classifiers:

	Class	Decision Tree				Naïve Bayes				Neural Network				Random Forest			
		A	R	P	F1	A	R	P	F1	A	R	P	F1	A	R	P	F1
Un-Balanced Data	0	.73	.82	.83	.82	.76	.90	.80	.85	.79	.97	.80	.88	.82	.95	.84	.89
	1	.73	.45	.44	.45	.76	.27	.47	.34	.79	.19	.68	.29	.82	.36	.70	.47
Balanced Data	0	.79	.77	.79	.79	.6	.59	.61	.60	.65	.81	.62	.70	.88	.93	.84	.88
	1	.79	.80	.78	.78	.6	.61	.60	.60	.65	.50	.72	.59	.88	.82	.92	.87

Figure 5: Comparing the different classifiers with un/Balanced Data

From the table, we can see an improvement in the recall, precision, and f-score between the two different cases.

5 CONCLUSION AND FUTURE WORK

In this project we have learned a lot about machine learning tools in python, such us scikit-learn, and NLP libraries such us nltk library for text processing, also we have learned new general techniques for data processing, and finally we have succeeded in implementing an automatic spam detector with satisfying results for the accuracy, recall, precision, and f1 score as compared with the previous results for training YELP's data-set by machine learning researchers. As a future plan, this project encourages us to dive more and more in machine learning algorithms, and trying to continue developing our models to get a higher performance on YELP's data-set.

6 REFERENCES

1. What Yelp Fake Review Filter Might Be Doing
2. On the Temporal Dynamics of Opinion Spamming: Case Studies on Yelp• S. Dixit and A. Agrawal, "Survey on review spam detection," in Int J Comput Technol ISSN (PRINT), 2013.
3. https://stackabuse.com/scikit-learn-save-and-restore-models/ [Accessed on 18/Jan/2021 at 1:00 pm]
4. http://cs229.stanford.edu/proj2017/final-reports/5229663.pdf [Accessed on 18/Jan/2021 at 1:40 pm]
5. https://www.investopedia.com/terms/n/neuralnetwork.asp [Accessed on 18/Jan/2021 at 3:00 pm]
6. https://www.geeksforgeeks.org/ml-feature-scaling-part-2 [Accessed on 18/Jan/2021 at 4:00 pm]
7. https://stackabuse.com/scikit-learn-save-and-restore-models/ [Accessed on 19/Jan/2021 at 1:40 pm]
8. https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c [Accessed on 19/Jan/2021 at 3:20 pm]