The Ohio State University, Department of Computer Science and Engineering

Homework 6: Prediction of rating from Yelp review text

Grover, Karan & Arora, Pragya & Ghai, Piyush

{grover.120, arora.170, ghai.8}@osu.edu

November 30, 2016

1 Problem Statement

1.1 About Yelp

Yelp [1] a famous website as well as a mobile app which publishes crowd sourced reviews about food joints and businesses. It also has a division which handles online reservations for restaurants. **Yelp Dataset Challenge**[2] is a publicly open contest sponsored by Yelp, in which the participants are challenged to use Yelp's data in an innovative way.

1.2 Our Mission

The Yelp dataset downloaded from Yelp dataset challenge website is huge and consists of over 2.7M reviews by roughly 687k users for over 86k businesses [2]. For this project, we chose to predict a review's rating based on the review text. The rating will be done on a scale of 1-5, where 1 stands for awful and 5 stands for excellent. We built multiple models and accessed which models would fit our use case the best. This is explained in more depth in the later sections of this report.

1.3 About the dataset

The Yelp Dataset consists of several files in JSON format of the data. The main files as per our use-case were the *yelp_academic_dataset_business.json* & *yelp_academic_dataset_review.json*. The two data files were **2.13 GB** & **73.6MB** in size respectively. The data representation in the for acadmic_dataset_business is as follows :

```
{
    "type": "business",
    "business_id": (encrypted business id),
    "name": (business name),
    "neighborhoods": [(hood names)],
    "full_address": (localized address),
    "city": (city),
    "state": (state),
    "latitude": latitude,
    "longitude": longitude,
    "stars": (star rating, rounded to half-stars),
    "review_count": review count,
    "categories": [(localized category names)]
    "open": True / False (corresponds to closed, not business hours),
    "hours": {
        (day_of_week): {
            "open": (HH:MM),
            "close": (HH:MM)
        },
        ...
    },
    "attributes": {
        (attribute_name): (attribute_value),
    },
}
```

The data representation in the for acadmic_dataset_review is as follows :

```
{    "type": "review",
     "business_id": (encrypted business id),
     "user_id": (encrypted user id),
     "stars": (star rating, rounded to half-stars),
     "text": (review text),
     "date": (date, formatted like "2012-03-14"),
     "votes": {(vote type): (count)},
}
```

The given datasets were first converted and exported into csv formats using a simple python script. The python script is a part of the **PreProcessing1.py** python file.

2 Program Description

The code has been broken down into the following parts:

1. **Pre-Processing 1**
   In pre-processing 1, Yelp provided JSON files were converted into the CSV format. The required CSV files were loaded as Python data-frames and then merged, filtered and stored locally for later usage.

2. **Pre-Processing 2**
   In pre-processing 2, the review text is cleaned up, grouped and stored according to the rating of the reviews, which further gets partitioned into training and testing dataset.

3. **Models**
   We implemented a total of 8 models and each model has been separated into a different python file. Following are the models which were implemented and compared:
   - **Baseline model**
   - **TFIDF model**
   - **Bag of words model**
   - **Bigram model**
   - **Trigram model**
   - **Bi and Tri-gram model**
   - **LDA Model**
   - **LDA + Sentiment layer model**

3 Model Description

The following models were implemented and evaluated:

1. **Baseline model**
   In the baseline model, the average rating across all reviews is assigned as the predicted rating for each review. The average rating of all reviews was found to be 3.9 which was rounded off to 4 and used as baseline prediction. This was used as a simple benchmark for all the other models we applied to this dataset.

2. **TFIDF model**
   In TFIDF or term frequency-inverse document frequency, each review text is represented as a vector having dimension equal to the number of words in the dictionary and each dimension representing a weight of importance of the word in the review text. This TFIDF vector representations of the review corpus become the features to the classification model.

3. **Bag of words model**
   In the bag of words or the unigram model, the review texts are converted to term frequency vectors, where each dimension represents the count of the term in the review text. The term frequency representations of the entire corpus become the features to the classification model.

4. **Bigram model**
   In the Bigram model, the review text is broken down into bigrams - set of two words occurring consecutively in the text and the document is then represented as the term frequency vector of these bigrams. Each dimension of the term frequency vector represents the count of the bigram in the review text.

5. **Trigam model**
   In the Trigram model, the review text is broken down into trigrams - set of three words occurring consecutively. The term frequency vectors of these sets become the features of the review text where each dimension represents the count of the trigram in the review text.

6. Bi and Tri-gram model
   In the Bi and Tri-gram mode, the review text is broken down into both bi and tri-grams. The term frequency vectors of these sets become the features of a review text where each dimension represents the count of the bigram or trigram in the review text.

7. **LDA model**
   All the previous models consider all the words/bigrams/trigrams and their frequency as the training features. In the case where the corpus is huge, the dimensionality of the input features would be huge. To reduce the dimensions, we have used the Latent Dirichlet allocation (LDA) algorithm which discovers a given number of topics or themes in the corpus. Each review text is then represented as vector of probability distribution across these topics. These topic distribution vectors become the features to the classification model. We have used the genism package in python to create the LDA model:

```
{
        ldamodel = gensim.models.ldamodel.LdaModel(corpus,
           num_topics=15, id2word = dictionary)
}
```

   Parameters:

   - num_topics: The number of topics that should be discovered.
   - id2word: The dictionary created from the corpus

8. **LDA model + Sentiment Layer** In this model, sentiment of the review was fed in as the feature alongside the topic distribution of the review. The sentiment of the review was extracted using the NaÃŕve Bayes classifier.

4 Coding Contribution

4.1 Data Transformation Pre-Processing

Since the files were in JSON format, they were first converted to CSV format. After converting to CSV format. After converting to CSV format, the business and review data frames were joined on *business_id* for restaurant establishments. From the combined dataset, we got over 1.6M reviews for restaurants alone. The dataset now consisted of just two columns : *Review Text* & *Star Rating*.

```
{
        len(resto_review_data)
```

```
        1630712
}
```

The length of reviews was tweaked based on our observations from Exploratory Dataset Analysis. Since a very small review might not contribute meaningfully to our models, we decided to limit the minimum characters in a review to 50, while the maximum character limit was fixed to 500 (this was kept low, because at higher ranges, the dataset was still too huge and required more computing power to run than was possible on our MacBooks). This further reduced the number of records to a little over 900k.

```
{
        resto_review = reduceReviewBasedOnLength(resto_review_data=
           resto_review_data, minReviewLen=50, maxReviewLen=500)
len(resto_review)
        910340
}
```

4.2 Cleaning

Since we are dealing with real world text reviews, the data available to us will contain plenty of punctation words as well common English stopwords. We followed a standard process for text mining to further cleanse and process the review dataset. In this, we first converted all the reviews to **lowercase**. We then removed **numbers** & **punctuations**. After this, we **tokenized** the text using the NLTK library's tokenizer. Post this, we removed the **stopwords** and we also performed **stemming** of the words. The cleaned dataset was thus used as an input to all our models trained.

4.3 Creating Training and Testing Corpus

We used **70-30** sampling to create a training and testing corpus from the cleaned text reviews. In order to ensure sufficient representation of all the star label values in our training corpus, we split the **original corpus on the basis of the star labels**, thus creating 5 corpora, one for each star label. The 70-30 sampling was then carried out on each of the 5 corpora and the training and test files thus created were then combined to create one big training and testing corpus. The number of reviews in training corpus for each star label is given in Table 4.1.

Table 4.1: Star rating distribution in the training corpora

Dataset	Count
Rating 1	58739
Rating 2	47154
Rating 3	72107
Rating 4	174334
Rating 5	284902

4.4 LDA Model Development

The LDA[4] model is present in **gensim package** in python. The inbuilt library method was not so straightforward and required a **vectorized bag of words** corpus as an input. It also required a dictionary developed from the available training corpora. The parameters that could be tweaked while developing the model were the corpora size and the total number of topics we want to extract. We chose the **total topics as 7**. In a normal vectorized corpus, the dimensionality would have

been the entire size of the disctionary, which is very huge. Selecting the total topics essentially will reduce the dimensionality of our training corpora to merely 7 selected topics. The topic probability distribution dataset was used as a feature to create new training corpora which was used to train off the shelf classifiers such as *MultinomialNaiveBayes, LogisticRegression, RandomForestClassifier, AdaBoostClassifier*. The performance of the models is discussed in a separate section on Model Evaluation.

4.5 Contributions from team members

All team members had equal contributions to this assignment. (33% each). The key contributions are as follows :

1. Karan Grover.
    - Performed the conversion of json to csv.
    - Suggested the merger of business and reviews file into one.
    - Performed the cleaning and pre-processing of the dataset.
2. Pragya Arora
    - Performed the exploratory analysis of the dataset, and suggested using a limit size range of reviews.
    - Suggested and worked on Bi-Grams, Tri Grams & Bag of Words approach.
    - Jointly contributed with Piyush in TF-IDF Vectorized model.
3. Piyush Ghai
    - Suggested to keep the representation of all star labels same in the training and testing split.
    - Suggested and worked on LDA & LDA + Sentiment Analysis approaches for classification.

5 Tools and Technology Stacks

For this assignment, we've used a variety of tools. The coding for this assignment was done using Python. The following is an exhaustive list of modules/tools used :

- Python 2.7
- Pandas (For manipulating the dataframes)
- nltk (For corpus, dictionary, stopwords, stemming etc)
- NumPy (For classification model results)
- seaborn (For graphs)
- cPickle (For saving pickle files and using them later in models)
- iPython (The interactive python shell, which was used for development)
- sci-kit learn (For various classification algorithms)
- Gensim (For LDA model)

References

[1] ggplot Library is used in this assignment to plot most of the graphs in this assgnment `http://ggplot2.org/`.

[2] ggplot Library is used in this assignment to plot most of the graphs in this assgnment `http://ggplot2.org/`.

[3] ggplot Library is used in this assignment to plot most of the graphs in this assgnment `http://ggplot2.org/`.

[4] ggplot Library is used in this assignment to plot most of the graphs in this assgnment `http://ggplot2.org/`.