Data Mining Group Project

Part 1: Text Classification

Group4

Teacher: Nicolas TURENNE

1730026058 Alvin LIANG

1730014072 Michael XU

1730026153 Hawthorn ZHAO

Data Science

2019.12.23

1 Introduction to Text Classification

Text classification is the process of assigning tags or categories to text according to its content. It's one of the fundamental tasks in Natural Language Processing (NLP) with broad applications such as sentiment analysis, topic labeling, spam detection, and intent detection.

Unstructured data in the form of text is everywhere: emails, chats, web pages, social media, support tickets, survey responses, and more. Text can be an extremely rich source of information, but extracting insights from it can be hard and time-consuming due to its unstructured nature. Businesses are turning to text classification for structuring text in a fast and cost-efficient way to enhance decision-making and automate processes.

But what is text classification? How does text classification work? What are the algorithms used for classifying text? In order to answer these questions, our group did a small demo of sentiment analysis based on the Amazon Review Polarity Dataset. In the following sections, we will talk about the general process to finish a text classification task and give some explanations on the technologies we used as well as the small demo we did.

Besides, in part 2, we will talk about another interesting task in NLP – Text Clustering. Then, in part 3, we will talk about a powerful model – BERT and another demo using BERT.

2 Data Acquisition

For this part, we use the Amazon Review Polarity Dataset (Version 3, Updated 09/09/2015). It is constructed by Xiang Zhang (xiang.zhang@nyu.edu) from the Amazon reviews dataset (The Amazon reviews dataset consists of reviews from amazon. The data span a period of 18 years, including about 35 million reviews up to March 2013.). It is used as a text classification benchmark in the following paper: Xiang Zhang, Junbo Zhao, Yann LeCun. Character-level Convolutional Networks for Text Classification. Advances in Neural Information Processing Systems 28 (NIPS

2015).

The original structure of the dataset is listed below:

	Class	ReviewTitle	ReviewText
0	2	Great CD	My lovely Pat has one of the GREAT voices of h...
1	2	One of the best game music soundtracks - for a...	Despite the fact that I have only played a sma...
2	1	Batteries died within a year ...	I bought this charger in Jul 2003 and it worke...
3	2	works fine, but Maha Energy is better	Check out Maha Energy's website. Their Powerex...
4	2	Great for the non-audiophile	Reviewed quite a bit of the combo players and ...
5	1	DVD Player crapped out after one year	I also began having the incorrect disc problem...
6	1	Incorrect Disc	I love the style of this, but after a couple y...
7	1	DVD menu select problems	I cannot scroll through a DVD menu that is set...
8	2	Unique Weird Orientalia from the 1930's	Exotic tales of the Orient from the 1930's. "D...
9	1	Not an "ultimate guide"	Firstly,I enjoyed the format and tone of the b...

As you can see, there are 3 attributes. The first one is the “Class”. “Class” attribute is the label that we want to do the prediction during our classification task. There are only 2 possible value for “Class”, [1,2]. 1 means the user gives a negative review on the product, 2 means the user gives a positive review on the product. The ReivewTitle and the ReviewText are the title and the content of the review, respectively.

By observation, we can find that the review titles contain some useful information but are not special enough for us to treat them as a single part. For convenience, we combine them(ReviewTitle, ReviewText) into one attribute “Review”. The tuple structure after combination is listed below:

	Class	Review
0	2	Great CD : My lovely Pat has one of the GREAT ...
1	2	One of the best game music soundtracks - for a...
2	1	Batteries died within a year ... : I bought th...
3	2	works fine, but Maha Energy is better : Check ...
4	2	Great for the non-audiophile : Reviewed quite ...
5	1	DVD Player crapped out after one year : I also...
6	1	Incorrect Disc : I love the style of this, but...
7	1	DVD menu select problems : I cannot scroll thr...
8	2	Unique Weird Orientalia from the 1930's : Exot...
9	1	Not an "ultimate guide" : Firstly,I enjoyed th...

For the size of the dataset, the original of this dataset is about 2GB. Considering the computation limitation and for convenience, we use 2000 records of it.

3 Text Preprocessing

Since, text is the most unstructured form of all the available data, various types of noise are present in it and the data is not readily analyzable without any pre-processing. The entire process of cleaning and standardization of text, making it noise-free and ready for analysis is known as text preprocessing.

In this section, we will mainly talk about 3 steps. They are Noise Removal, Lexicon Normalization, and Object Standardization.

Also, the general steps to get the "clean text" is listed below:

3.1 Noise Removal

In Text Mining, any word which is not relevant to the context of the data can be specified as the noise. According to this definition, language stop words, commonly-used words of a language, URLs, links, social media entities, special patterns, and punctuations are all noisy. In this step, we need to remove all types of noisy entities in the text.

There are mainly two ways to do the noise removal:

- The first one is to prepare a dictionary of noisy entities, and then check all the words in the text object. If the noisy words occur in the text, eliminate them.
- The second one is to use the regular expressions while dealing with special patterns. By using regular expression, we can remove all the special pattern but remain all the other parts.

Here is an example:

```
# A sentence that we obtained from the social network

sentence = """#Data Mining : This course is really, really, really
interesting. The teachers are very, very, very nice! I love them
so so so so so so much!"""

remove_noisy_word(remove_pattern(sentence))
```

```
'Data Mining course interesting teachers nice love much'
```

As you can see, after the noisy removal. The core content (or the main topic) of the text does not change. But its length becomes shorter and it becomes “cleaner”.

3.2 Lexicon Normalization

After removing all the obvious noisy words, there are still some textual noise which is about the multiple representations of a single meaning word. For example, the word ”play” may occur in the form of “plays”, ”played”, ”playing”, ”player”. You may think they have slightly different meanings but they are contextually similar. We can convert them into their normalized form (lemma). Through this normalization step, we can convert the high dimensional features to the low dimensional space. There are mainly two common lexicon normalization practices:

- Stemming: A rudimentary rule-based process of stripping the suffixes (“ing”, “ly”, “es”, “s” etc.) from a word.
- Lemmatization: An organized & step by step procedure of obtaining the root form of the word, it makes use of vocabulary (dictionary importance of words) and morphological analysis (word structure and grammar relations).

By using the NLTK package in Python, we get the following example:

```
word = "multiplying"

print(lem.lemmatize(word, "v"))
print(stem.stem(word))
```

```
multiply
multipli
```

As you can see, by providing an extra parameter, the lemmatization algorithm seems to get a better normalization result.

3.3 Object Standardization

Besides all the noise mentioned above, text data often contains acronyms, hashtags with attached words, and colloquial slangs. These pieces are not recognized by search engines and models. So, we also need to deal with them, With the help of manually prepared data dictionaries, this type of noise can be fixed.

An example is listed below:

```
# A sentence that we obtained from the social network

sentence = "xswl! The moive is awsm. I luv it!"

print(lookupWords(remove_pattern(sentence)))
print(remove_noisy_word(lookupWords(remove_pattern(sentence))))
```

```
It is really funny The moive is awesome I love it
funny moive awesome love
```

As you can see, after changing, the sentence becomes more standard and we can use the method mentioned above to make it even cleaner.

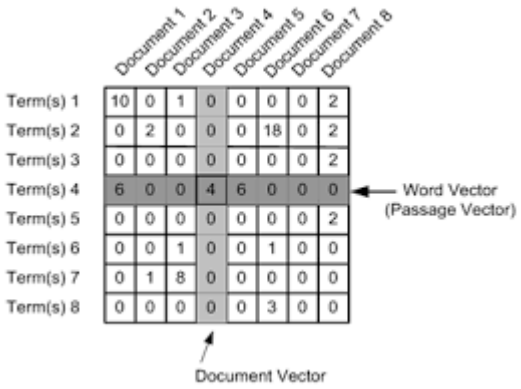
3.4 Notes

Besides these 3 steps, if you want to further improve your later performance, you can do the grammar checker, spelling correction. And all the other preprocessing techniques can be used.

4 Feature Engineering (Text to Features)

No matter how simple the meaning of the text is, the computer can only understand numerical data. So, to analyze a preprocessed data, it needs to be converted into features (A numerical vector or matrix). Depending upon the usage, text features can be constructed using different techniques. In this section, we will talk about the count vector, TF-IDF, text vector, and topic modeling.

4.1 Use Count Vector as Feature Vector



Count vector is the most naïve way to do the feature engineering. It is just another representation of the original dataset. In the count matrix, each line is a word vector while each column is a document vector. The a(ij) entity in the count matrix is the frequency of term i in document j.

4.2 Use (Word Level) TF-IDF vector as Feature Vector

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

tf_{ij} = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Term Frequency – Inverse Document Frequency (TF_IDF) is a weighted model commonly used for information retrieval problems. It aims to convert the text documents into vector models on the basis of occurrence of words in the documents without taking considering the exact ordering. For Example – let say there is a dataset of N text documents, in any document “D”, TF and IDF will be defined as:

- Term Frequency (TF) – TF for a term “t” is defined as the count of a term “t” in a document “D”
- Inverse Document Frequency (IDF) – IDF for a term is defined as logarithm of ratio of total documents available in the corpus and number of documents containing the term T.

TF-IDF formula gives the relative importance of a term in a corpus (list of documents).

4.3 Use N-Gram TF-IDF vector as Feature Vector

A combination of N words together is called N-Grams. N grams (N > 1) are generally more informative as compared to words (Unigrams) as features. Also, bigrams (N = 2) are considered as the most important features of all the others. By using the TF-IDF formula computing the N-Grams’ TF and IDF, we can get the N-Gram TF-IDF vector.

4.4 Use Text Vectors as Feature Vector (Word Embedding)

Word embedding is the modern way of representing words as vectors. The aim of word embedding is to redefine the high dimensional word features into low dimensional feature vectors by preserving the contextual similarity in the corpus. They are widely used in deep learning models such as Convolutional Neural Networks and Recurrent Neural Networks.
In part 2 and part 3, we will talk about this in detail.

4.5 Use Topic Model as Feature Vector

Topic modeling is a process of automatically identifying the topics present in a text corpus, it derives the hidden patterns among the words in the corpus in an unsupervised manner. Topics are defined as “a repeating pattern of co-occurring terms in a corpus”. A good topic model results in – “health”, “doctor”, “patient”, “hospital” for a topic – Healthcare, and “farm”, “crops”, “wheat” for a topic – “Farming”.
In part 2, we will talk about this and its implement using Latent Dirichlet Allocation (LDA).

4.6 Notes

Count or Density based features can also be used in models and analysis. Under some situations, it can show a great impact in learning models. These features include: word count (people with strong feelings may have longer review), the average length of words (words in a story book for child may be shorter than others in academic paper), the number of words of a particular part of speech (more adjective words in a descriptive essay). We should construct and choose the feature vectors case-by-case.

5. Modeling

After getting the feature vectors, the classification for text is nothing different with the general classification. So, in our demo, we use 6 kinds of different models, they are:

- Naive Bayes
- Logistic Regression
  - A linear classifier that use logistic / sigmoid function to estimate the probability in order to do the classification task.
- SVM
- Random Forest (Bagging Model)
  - A bootstrapping algorithm with Decision tree (CART) model. It tries to build multiple CART models with different samples and different initial variables.
- Xgboost (Boosting Model)
  - Extreme Gradient Boosting (xgboost) is similar to gradient boosting framework but more efficient. It has both linear model solver and tree learning algorithms. So, what makes it fast is its capacity to do parallel computation on a single machine.
  - Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss

function.

- KNN(K=9)

Then, just as what I have mentioned before, there are 3 vectors we can use as the feature vectors. They are:

- Count Vector
- Word-Level TF-IDF
- N-Gram TF-IDF(N=2,3)

So, their performance is listed below:

Model	Feature Vector	Accuracy Score
Logistic Regression	Count Vectors	0.814
Naive Bayes	Count Vectors	0.812
Logistic Regression	Word-Level TF-IDF	0.81
Naive Bayes	Word-Level TF-IDF	0.796
Naive Bayes	N-Gram Vectors	0.796
Logistic Regression	N-Gram Vectors	0.784
Xgboost	Word-Level TF-IDF	0.754
Xgboost	Count Vectors	0.75
KNN	N-Gram Vectors	0.744
Random Forest	Word-Level TF-IDF	0.734
Random Forest	Count Vectors	0.732
KNN	Word-Level TF-IDF	0.702
Xgboost	N-Gram Vectors	0.678
Random Forest	N-Gram Vectors	0.66
KNN	Count Vectors	0.63
SVM	Count Vectors	0.52
SVM	Word-Level TF-IDF	0.52
SVM	N-Gram Vectors	0.52

6 Error Analysis

6.1 Overall

By using Count Vectors as the Feature Vector and use Logistic Regression Model, my demo gets its highest accuracy score 0.814. In general, it is not very good, the reasons that I thought it should be are listed below:

- Small size of the training set

As I mentioned before, there are totally 2000 records used in this demo. Besides the unseen records used for testing, the training set only contains about 1000 records. For

a NLP task, it is quite small. Without enough training, the accuracy of the classifier may not as good as it should be.

- The "dirty" raw data

Even though I talk about the text preprocessing techniques in section 3, I use the raw data directly without any cleaning. This is because both noisy removal and standardization needs manually-provided dictionary as a source list. At current stage, our group do not have a perfect dictionary to do this task. Noisy in the raw data will obviously influence the performance of the classifiers.

- Naïve implement of Feature Engineering

Small size of training set, poor quality raw data with noise, both of them will make the feature matrix very sparse. Using such kind of feature vectors may influence the performance of the model.

6.2 Relatively Well-Performed Models (Naive Bayes and Logistic Regression)

Under the situation of this sentiment analysis, all the models that based on the probability of the occurrence of specific words should perform the classification quite well. It is obvious that a review contains the words
"happy", "good", "awesome", "amazing", "useful" will express a positive meaning. While a review should be labeled as "negative" if it contains
"bad", "rubbish", "waste", "broken", "poor", "useless".

6.3 Relatively Poor-Performance Model and Feature Vectors (SVM and N-Gram Vectors)

For SVM, it is quite obvious for us to know that the points representing the text data is not linearly-separable in this case. So, SVM fails to do the prediction.
For N-Gram Vectors, you may think it should be more accurate than the word-level TF-IDF vectors because it contains more information. But after the N-gram, the data must be raised into a high dimensional space. Recall that there are only about 1000 records in the training set, such small amount of the records is not enough to train the

classifier sufficiently. So, it influences its performance.

6.4 A Special Case: KNN based on Count Vector.

It is quite surprise to see that the KNN based on Count Vector performed relatively poor. But recall the KNN algorithm, it uses the k nearest neighbor to estimate the class of the current point. Just as what I have mentioned before the feature matrix is sparse. So, the distance between two points may be very big. Besides, another important fact is that reviews may have different length. This will make two points with similar meanings but different lengths extremely far from each other. Both of these two reasons will decrease the accuracy of it.