Hotel Review Opinion Mining

> The first term project for the lecture Natural Language Processing, National Taiwan University, Spring 2017.

Team name: “That Good Good”

Name	Student ID	Email registered
林祐萱	B03902055	linamy85@gmail.com (mailto:linamy85@gmail.com)
廖瑋中	B02902105	b02902105@ntu.edu.tw (mailto:b02902105@ntu.edu.tw)
鄭筱樺	B03902024	jke.evie@gmail.com (mailto:jke.evie@gmail.com)

Methodology

We decide to divide the task into two parts, aspect extraction and polarity classification.

First, we utilize aspect extraction to determine which aspects each review is related to. Then, we label each review as positive or negative by polarity classification. Finally, we merge the two results. If the review is determined to be unrelated to an aspect, we label it as 0. Otherwise, we label it as 1 or -1, according to the result of polarity classification.

Note that in the previous method, all aspects in a review will be labelled as the same polarity. However, it is unreasonable. Therefore, we improve it by labelling each sentence in the test review. Eventually, we count whether positive or negative sentences are more in a review for each aspect. If no sentence is labelled as the aspect, the aspect is unrelated. Otherwise, the aspect of the review is labelled as 1 or -1, according to the number of positive and negative sentences.

We elaborate the details of aspect extraction and polarity classification in the following subsections.

Segmentation

We use Jieba (https://github.com/fxsjy/jieba) as segmentation tool.

Aspect Extraction

(A) Word Embedding – Word2Vec

Package: gensim (http://radimrehurek.com/gensim/index.html)

- First, we preprocess the datasets, including `aspect_review.txt` , `test_review.txt` , and `polarity_review.txt` , by segmentation and stop words removal.
- Then, we train the word embeddings with these datasets.
- By selecting the words in the vocabulary whose similarity with aspect terms are higher than a threshold, aspect-alike terms are found.

(B) Labeled Latent Dirichlet Allocation (LDA)

LDA is a traditional topic model. However, in our case, there are 200 data with aspect labeled, making this more like a semi-supervised problem. As a result, we decided to use Labeled LDA.

Package: JGibbLabeledLDA (https://github.com/myleott/JGibbLabeledLDA).

- First, we preprocess the datasets as the way in (A).
- Second, the corpus made of three files is used to train the Labeled LDA.
- Several obvious aspect terms doesn't get high probability in the corresponding topic, so we need to assign a high probability to the topic sentence included its aspect terms.
- In the end, we assign labels with probability higher than given threshold to each document (review).

Polarity Classification

Package: sklearn (http://scikit-learn.org/stable/)

For polarity classification, mainly we refer to the wonderful paper writtten by Ho-Cheng Yu et al [1]. Its main idea is to compute sentiment orientation (SO) for each word, and then use SO as feature values to do classification. The details are as following.

First, compute SO for each word in training data `polarity_review.txt` , where

$$PSO = \frac{\#\text{ of positive reviews containing the word} + 1}{\#\text{ of positive reviews}}$$

$$NSO = \frac{\#\text{ of negative reviews containing the word} + 1}{\#\text{ of negative reviews}}$$

$$SO = log(\frac{PSO}{NSO}).$$

Then, we remove the words which have low frequency or low $|SO|$ value. The remaining words consists of the feature space. Subsequently, we can embed our training data into a matrix $X \in \mathbb{R}^{N \times k}$ where N is the size of training data and k is the size of feature space. We use TFSSIDF as the value for the elements of X, where

$$TF = \frac{\text{\# of the word in the review}}{\text{\# of words in the review}}$$

$$TFIDF = TF \times log(\frac{\text{\# of reviews containing the word}}{\text{\# of reviews}})$$

$$TFSOIDF = TFIDF \times (|SO| + 1)$$

$$TFSSIDF = \begin{cases} TFSOIDF \times k & \text{, if the word is in NTUSD} \\ TFSOIDF & \text{, otherwise.} \end{cases}$$

For the detailed reasons why these numbers are used, please refer to the paper.
Finally, after we embed these words into a matrix, we can do classification with it. Simply, we use SVM to solve the binary classification problem. We use LIBSVM[2] in the SVM package sklearn (http://scikit-learn.org/stable/), which is a popular Python package for machine learning, to do the SVM training.

For each new testing data in `test_review.txt`, we first embed each review into vectors. Then, we can use the vectors with the model we have trained to predict the polarity of each review.

Experiments

Word2Vec

- Since we only want to calculate the accuracy of aspect learning, we use the following strategy:
  - If the review string contains the specific aspect, output 2; Else output 0

vector size	CBOW window	min count	threshold	public accuracy	private accuracy
100	5	1	0.37	0.33986	0.32796
100	5	1	0.45	0.43318	0.41312
100	5	1	0.57	0.51613	0.50058
150	5	30	0.57	0.51728	0.50058
150	10	30	0.57	0.52074	0.50403

Word2Vec + SVM

- SO threshold is the threshold that uses to decide feature space. If the word's $|SO|$ value is greater than SO threshold, it will be in the feature space. Otherwirse, it is discarded.
- NTUSD k is the k mentioned in the TFSSIDF formula. $k = 1$ simply means we do not use NTUSD. $k = 2$ is the best result presented in the paper.

Word2Vec threshold	SO threshold	NTUSD k	Label per sentence or review	public accuracy	private accuracy
0.44	1.0	1	review	0.69700	0.65938
0.44	1.0	1	sentence	0.75461	0.7560
0.57	1.0	1	review	0.79263	0.76985
0.57	1.0	1	sentence	0.79608	0.79977
0.57	0.2	1	sentence	0.79263	0.79402
0.57	0.5	1	sentence	0.80069	0.79287
0.57	0.5	2	sentence	0.80184	0.79287
0.60	0.5	2	sentence	0.80415	0.79402
0.65	0.5	2	sentence	0.76613	0.74914

Labeled LDA + SVM

- LDA_thr = Labeled LDA threshold
- iter = iteration
- SO_thr = SO threshold

LDA_thr	iter	SO	NTUSD k	Label per sentence or review	public accuracy	private accuracy
0.35	500	0.5	2	sentence	0.74885	0.72957
0.3	500	0.5	2	sentence	0.75346	0.73763
0.3	500	1.0	1	review	0.75115	0.74223
0.31	1000	1.0	1	review	0.74654	0.73648
0.28	600	1.0	1	review	0.73502	0.72152
0.25	300	1.0	1	review	0.72811	0.69965

Discussions

Segmentation tool

In the “dict.txt.big” dictionary file provided by Jieba, there are terms like “周遭環境” that includes “周遭” and “環境”, and also “環境” is a important aspect term in this problem. However, Jieba parses the sentences “這裡的周遭環境很棒” into [“這裡”, “的”, “周遭環境”, “很棒”]. but the term “周遭環境” seldom occurs in our training corpus, making it difficult to recognize “周遭環境” as a important aspect term.

- Some may say that substring can be used to fix this issue, however it doesn’t solve all the cases of this issue.

Word2Vec

- The parameters of the model do not affect the outcome very much.
- It is “threshold” of the similarity that decides the overall accuracy.

Polarity classification

By observing the experiment result, we can imply the following conclusions.

1. Labelling each sentence is better than just labelling each review.
2. SO threshold does not affect the result very much.
3. NTUSD does not affect the result very much.

Therefore, we can say that the key factor in the classification problem is the quality of feature. That is to say, the threshold of Word2Vec. Then, we can simply choose SO threshold and then use SVM to train the model, reaching a pretty good result.

Reference

1. 游和正、黃挺豪、陳信希。 (2012). 領域相關詞彙極性分析及文件情緒分類之研究。*Computational Linguistics and Chinese Language Processing,* pp.33-48.
2. Chang, Chih-Chung & Lin, Chih-Jen. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology,* pp. 27:1–27:27.