ERG3020: Web Analytics and Intelligence
Project Report
Classification for Online Social Network Analytics
Yixuan Liang (115010043) Shixin Liang (115010042)

1. Introduction

To get familiar with Tweet classification, we implemented the Naive Bayes classifier using Tweets data. We first worked on the Tweet Classification for Disaster Relief Project, which aims to build a system to automatically classify real tweets from Sandy, to get familiar with the classification method. Then we used the same method to build an application based on a new data classification from amazon-fine-food-review data. This application aims to judge whether a review is a satisfied one or a dissatisfied one.

2. Method and example: Tweet Classification for Disaster Relief

2.1 Data

The data are real tweets from Sandy. The tweets have been categorized and labeled by a human into five classes: Food, Water, Energy, Medical, None. Each tweet has been judged whether it means people need resource (need) or they can provide resource (Resource) or means nothing (N/A). Each tweet data contains three parts: Text, Category and Need/Resource. For each tweet, we will get an unique data like Table 1:

1	Tweet	Category	Need/Resource
2	Donations of batteries, flashlights, and cleaning supplies are always welcome at our donation drop off at 23-74 38th st btwn 23rd ave & Astoria Blvd. Thanks!	Energy	need
3	I want hurricane Sandy to cone so I can be stuck in my house with my family :) .....NOT !!!!!	None	N/A
4	Hi, I can help prepare food, serve food, offer clean up assistance but don't necessarily have any tools, just my hands and willingness to help. I only have a bicycle and could maybe help with deliveries. I'm within walking distance of Red Hook.	Food	resource
5	I cant believe Sandy.....	None	N/A
6	I have children and adult clothes including jacket , blanket, shoes , and bottle water	Water	resource
7	I have two small children but we would like to donate to families who need clothes or food or even toys for kids! But we don't know where and would rather not donate to red cross. We would rather directly donate. Thanks in advance if you can help! Melissa	Food	resource
8	Supermarket had no food #scary #hurricanesandy #nopower #instahub #instafood #instagram @ Super Stop & Shop http://t.co/hY0TRezH	None	N/A
9	spooky: the total lack of power from 34th st - south. #sandy	Energy	need
10	Will these storms ever give L.I. a break? First sandy, now a nor'easter. Still no power in long beach expected for a few wks. #c' monman	Energy	need

Table 1: Format of tweet data from Sandy

Because our aim is to build a system to classify tweets according to the Category part, we can ignore the Need/Resource part.

We hoped to use these data to build our AI system, and then test how well our system works by comparing the labels from the dataset with the labels our system produces. Therefore, we need to split the data into training examples and test examples. Of 1401 tweets, we randomly choose 281 (approx. 20%) tweets as test dataset and the rest as training dataset. We then developed the classifier using the training dataset, and used the classifier to label all the test dataset. And finally we compared the classifier's labels with the true labels and measure performance.

2.2 Naive Bayes classifier

We used Naive Bayes, a Machine Learning model, to develop the classifier. To construct Naive Bayes classifier, we first need to calculate two things: prior probabilities of categories and conditional probabilities of tokens.

Prior probabilities of categories

We need to calculate $P(C_i)$ for each category $C_i \in \{\text{Energy}, \text{Food}, \text{Medical}, \text{Water}, \text{None}\}$.

We estimate $P(C_i)$ by $\frac{\#\text{ tweets about } C_i}{\#\text{ tweets}}$

Conditional probabilities of tokens

For each token (i.e. word) x_j and each category C_i, we need to calculate $P(x_j|C_i)$.

We estimate $P(x_j|C_i) = \frac{P(x_j \text{ and } C_i)}{P(C_i)}$ by $\frac{\#\text{ tweets about } C_i \text{ containing } x_j}{\#\text{ tweets about } C_i}$

To calculate the prior probabilities, we should calculate the total number of tweets and the number of tweets that are about category c. Then we can calculate the probability of c by using $P(C_i)$ by $\frac{\#\text{ tweets about } C_i}{\#\text{ tweets}}$.

To calculate the conditional probabilities of token, we should first tokenize the tweets, and then calculate the number of category-c tweets containing that token. We can calculate the conditional probabilities of token by using $P(x_j|C_i) = \frac{P(x_j \text{ and } C_i)}{P(C_i)}$ by $\frac{\#\text{ tweets about } C_i \text{ containing } x_j}{\#\text{ tweets about } C_i}$. And we can use the conditional probabilities of token to get the most discriminative tokens by calculating the tokens that maximize P(c|token).

And we got the following result:

```
In [4]: prob_food, token_probs_food = calc_probs(tweets, "Food")
        prob_water, token_probs_water = calc_probs(tweets, "Water")
        prob_energy, token_probs_energy = calc_probs(tweets, "Energy")
        prob_medical, token_probs_medical = calc_probs(tweets, "Medical")
        prob_none, token_probs_none = calc_probs(tweets, "None")

        Class Food has prior probability 0.47
        Class Water has prior probability 0.09
        Class Energy has prior probability 0.12
        Class Medical has prior probability 0.04
        Class None has prior probability 0.28
```

prior probabilities of categories

2.3 Build a Naive Bayes classifier

After calculating P(Ci) and P(xj|Ci), we can start to classify tweets. We need to calculate the Posterior probability of categories for each category and then determine which is largest.

Posterior probability of categories

Given a tweet which is a set of tokens $\{x_1, \ldots, x_n\}$, the posterior probability of each category C_i is

$$P(C_i|x_1, \ldots, x_n) \propto P(C_i) \times P(x_1|C_i) \times P(x_2|C_i) \ldots \times P(x_n|C_i)$$

We have got the prior probabilities and the conditional probabilities of token by using the training data, we then can calculate the posterior probability of categories by using the test data. We calculated the posterior probability of all the five categories for one tweet. The category of the maximum posterior probability can be considered as the most-likely category for this tweet. This is our Naive Bayes classifier.

2.4 Evaluate the Naive Bayes classifier

We first got the predicted labels of the test dataset by using the Naive Bayes classifier we build above. We compared true labels and predicted labels in a table:

	Text	True category	Predicted category
0	i have a lot of canned goods and some clothing , but i can also buy and bring things as needed . please let me know what you need most .	Food	Food
1	how the **** am i supposed to get @ meekmill new album when i ai n't got power ? **** outaaa here sandy !	Energy	Energy
2	frankenstorm wo n't stop the bean ! thx for staying open for the neighbors who need coffee and treat ! (@ the bean) http : //t.co/zw7oa0tq	Food	None
3	deodorant toothpaste shampoo/conditioner baby shampoo kids toothbrush bar soap mouth wash q-tips painkillers	Medical	Medical
4	clothes for baby kids woman men , food , non perishable food , tools , toys , paper , furniture , any products ,	Food	Food
5	i have blankets , socks , non perishables , baby wipes , diapers	Food	Food
6	i can bring some clothing , non perishables , hygiene products and some baby supplies	Food	Food
7	oyster creek power plant is on alert for flooding ... . it 's about 80 miles away . great . # sandy	None	None
8	nonperishable food hygiene products temporary shelter	Food	Food
9	soo it 's almost 2 am and people are still waiting on that line to get gas.. it 's the shortest line i 've seen though -__- # gas # sandy # nyc	Energy	None

Because the result of each category is binary, we decided to use Precision and Recall to evaluate the Naive Bayes classifier.

For a binary classifier, the two categories can be regarded as positive and negative.

If the classifier's predicted label is positive, the labeling is a true positive if the prediction is correct, false positive if the prediction is wrong. If the classifier's predicted label is negative, the labeling is a true negative if the prediction is correct, false negative if the prediction is wrong.

Take our data as an example. When the classifier predicted a tweet's category is Food, we have:

		Predicted label	
		Food	Not Food
True label	Food	True positive	False negative
	Not Food	False positive	True negative

Then we can calculate Precision and Recall.

Precision: "Of all those labeled positive, how many were correctly labeled?"

$$\text{Precision} = \frac{tp}{tp + fp}$$

Where tp: true positive, fp: false positive

High precision means the classifier has few false positives.

Recall: "Of all the true positive examples, how many did the classifier detect?"

$$\text{Recall} = \frac{tp}{tp + fn}$$

Where tp: true positive, fn: flase negative

High recall means your classifier has few false negatives.

Because both of these two measures are important, we combined them into a single balanced measure called F1 score:

$$\text{F1} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

F1 is called the harmonic mean, a type of average. And it is only high if both precision and recall are high.

At the end, we combined F1 for our 5 categories by taking the average:

$$\text{Average F1} = \frac{\text{F1}_{\text{Energy}} + \text{F1}_{\text{Food}} + \text{F1}_{\text{Medical}} + \text{F1}_{\text{Water}} + \text{F1}_{\text{None}}}{5}$$

Here is our result:

```
Energy
Precision:  50.0
Recall:  60.0
F1:  54.54545454545455

Food
Precision:  83.56164383561644
Recall:  94.57364341085271
F1:  88.72727272727272

Medical
Precision:  85.71428571428571
Recall:  46.15384615384615
F1:  60.0

None
Precision:  82.85714285714286
Recall:  73.41772151898734
F1:  77.85234899328859

Water
Precision:  80.0
Recall:  40.0
F1:  53.333333333333336

Average F1:  66.89168191986984
```

F1 score of the test dataset

```
Energy
Precision:  91.33333333333333
Recall:  99.27536231884058
F1:  95.13888888888887

Food
Precision:  96.6355140186916
Recall:  97.91666666666667
F1:  97.27187206020695

Medical
Precision:  97.77777777777777
Recall:  100.0
F1:  98.87640449438202

None
Precision:  97.98657718120805
Recall:  94.49838187702265
F1:  96.21087314662273

Water
Precision:  100.0
Recall:  91.08910891089108
F1:  95.33678756476684

Average F1:  96.56696523097348
```

F1 score of the training dataset

We can see that average F1 score for the training dataset is much higher than average F1 score for the test dataset. It means that the classifier works well in the training dataset and there is no problem in the data. And the classifier's behavior in the test dataset is acceptable.

We also printed out a confusion matrix as a HTML table. Rows are true label while columns are predicted label. Predictions is a list of (tweet, predicted_category) pairs. The confusion matrix table is:

	Energy	Food	Medical	None	Water
Energy	23	10	0	6	1
Food	3	118	1	6	1
Medical	2	5	6	0	0
None	16	2	1	60	0
Water	1	8	0	1	10

From the table, we can clearly see the result, and it shows that our prediction is quiet accurate expect for the Medical and Water categories. We guess the inaccuracy due to their low frequency of occurrences.

3. Build an application based on amazon-fine-food-review data to judge whether a customer is satisfied or not

3.1 Data

We downloaded the data from Amazon's website. The data is customers' review about the food they bought from Amazon. As the whole dataset is huge, we only chose a part of the data, whose ID is larger than 500000, to build the Naive Bayes classifier. Then there are total 68454 data. Each data contains three parts: ID, Score and Text. The score represents customer satisfaction (1 means least satisfied and 5 means most satisfied). For each review, we got an unique data like Table 2:

	Id	Score	Text
0	500001	5	I was looking for an easy and convenient way t...
1	500002	5	DO NOT be freaked out by the ostrich! This tas...
2	500003	5	I bought the Ostrim with a little apprehension...
3	500004	5	At only 80 calories these are a great bang for...
4	500005	5	These are quite tasty and by far the leanest, ...

Table 2: Format of amazon-fine-food-review data

Then we looked at the data, the amount of each score is:

Score	1	2	3	4	5
Amount	5981	3509	5031	9611	44322

From the table we can know we have unbalanced categories since Score 5 is much more common. Therefore, we used the smallest number as split criterion. We let 3500 as total for each category, then split them into 2500 (approx. 70%) and 1000 as trainning and test dataset.

3.2 Build a Naive Bayes classifier and do evaluation

We used the same method in part 2: Tweet Classification for Disaster Relief. First, we tokenized the Text part of each review data. Second, we calculated the prior probabilities and the conditional probabilities of token by using the training data. Third, we calculated the posterior probability of categories by using the test data. At the end, we evaluated the Naive Bayes classifier by calculating average F1 score. Here are our results:

```
1
Precision:  67.31301939058172
Recall:  24.3
F1:  35.70903747244673

2
Precision:  30.971128608923884
Recall:  59.0
F1:  40.61962134251291

3
Precision:  25.979772439949432
Recall:  41.1
F1:  31.835786212238574

4
Precision:  33.601841196777904
Recall:  29.2
F1:  31.24665596575709

5
Precision:  69.6113074204947
Recall:  19.7
F1:  30.70927513639906

Average F1:  34.024075225870874
```

F1 score of the test dataset

```
1
Precision:  74.29352780309937
Recall:  32.6
F1:  45.31554072838476

2
Precision:  36.05080831408776
Recall:  62.44
F1:  45.710102489019036

3
Precision:  36.51188501934771
Recall:  52.84
F1:  43.18404707420726

4
Precision:  48.69179600886918
Recall:  43.92
F1:  46.18296529968455

5
Precision:  79.33333333333333
Recall:  38.08
F1:  51.45945945945945

Average F1:  46.370423010151015
```

F1 score of the training dataset

We can see that F1 score is quite low, so we checked F1 score of the training data and printed out the confusion matrix table:

	1	2	3	4	5
1	815	1181	397	78	29
2	130	1561	679	111	19
3	76	783	1321	298	22
4	38	421	765	1098	178
5	38	384	456	670	952

Table 3:Confusion matrix table

F1 score of the training dataset is not high either. It seems that our model is not a good one and the problem may due to the classification we did among the data. By checking the confusion matrix table, we can see that the Naive Bayes classifier we built cannot distinguish score 1 and score 2. For example, nearly half of the reviews whose score is actually 1 are predicted as 2 incorrectly. Score 4 and score 5 are in the same situation. Therefore, we can do improvement by categorizing the score into good reviews and bad reviews base on the scores. 1,2 as dissatisfied and 4,5 as satisfied.

3.3 Improvement

we categorized the score into good reviews and bad reviews base on the scores, 1,2 as dissatisfied and 4,5 as satisfied, and deleted score 3 reviews. There are 53933 satisfied reviews and 9490 dissatisfied reviews. We used the smallest number as split criterion. We let 9400 as total for each category, then split them into 7500 (approx. 80%) and 1900 as trainning and test dataset. Again, we used the same method to build a Naive Bayes classifier and do evaluation. Here are the results:

```
Satisfied
Precision:  90.79919408999328
Recall:  71.15789473684211
F1:  79.78754794924757

Dissatisfied
Precision:  76.28732150584163
Recall:  92.78947368421052
F1:  83.73308002849679

Average F1:  81.76031398887218
```

F1 score of the test dataset

	Satisfied	Dissatisfied
Satisfied	1352	548
Dissatisfied	137	1763

Confusion matrix table

The average F1 score increased a lot and the confusion matrix table shows that our new Naive Bayes classifier works well, but it seems that the classifier is more likely to make dissatisfied predictions.

We selected the key words which appear frequently in the text of satisfied reviews and dissatisfied reviews. We printed word clouds of the two categories to visualize the key words. Here are the word clouds:

Fig1. Word cloud of satisfied reviews

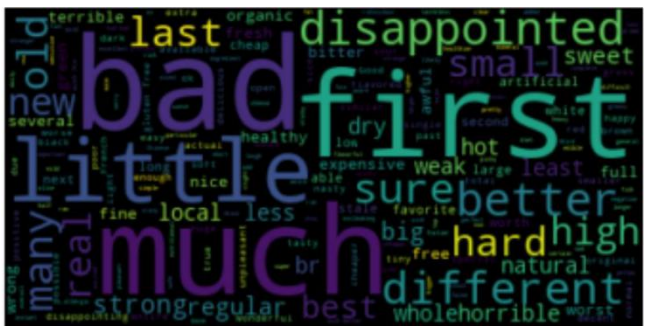

Fig2. Word cloud of dissatisfied reviews

From the word clouds, we can clearly see good words in the word cloud of satisfied reviews and bad words in the word cloud of dissatisfied reviews. For example, the word “best” usually appears in a satisfied review and the word “bad” usually appears in a dissatisfied review. These words can help us to quickly make a judgement whether a review is satisfied or not. We can also find some common words in both picture, which means these words frequently appear in a review but with no emotion. For example, the word “little” and the word “much” are big in both pictures. We guess these words may have a bad effect on the accuracy of our Naive Bayes classifier, so our future job is to reduce the influence of those common neutral words.

4. Additional Experiment

During the process of building our language model, we used a more complex language model: N-gram Language Model. In a bigram Language Model, each pair of adjacent words (=bigram) has a probability. For example, we know the probability that "York" follows "New" is high. Again, we can learn from data via counting, and get the probability of a sequence (pair of adjacent words) by calculating the product of the probabilities of the pairs. For example, P("I instantly recognized it") = P("I") P("instantly" | "I") P("recognized" | "instantly") P("it" | "recognized"). This idea can be extended to trigrams (word triples), 4-grams, 5-grams and so on. Sometimes this idea can improve the classifier.

However, in both our implementations, classification for tweets and sentiment analysis for reviews, the N-gram language model not only does not seem to improve our initial results but may cause overfitting problems. Thus, we decided to stick to our simply naïve Bayesian model.

The result for our sentiment analysis for reviews using bigram language model is as follow, we can see that the average F1 score for bigram model is only 80.26, which is less than 81.76, the score of the uni-gram model used before.

```
Satisfied
Precision:  86.29792583280955
Recall:  72.26315789473684
F1:  78.65940991120023

Dissatisfied
Precision:  76.1430511543685
Recall:  88.52631578947368
F1:  81.86906789973229

Average F1:  80.26423890546626
```

Last but not the least, we also tried to remove stopwords from our input before running the classifier. The stopwords library we used is from the nltk.corpus package. However, the result also does not improve our model accuracy significantly.

5. Conclusion

In this project, we tried to build a AI system which can automatically judge whether a review from Amazon is a satisfied one or a dissatisfied one. We used the Naive Bayes classifier to do classification. We worked on the Tweet Classification for Disaster Relief Project to get familiar with the method. We learned how to build a Naive Bayes classifier by using prior probabilities, conditional probabilities, posterior probability and bigram Language Model. We learned how to evaluate the Naive Bayes classifier by calculating precision, recall and F1 score. After finishing the Tweet Classification for Disaster Relief Project, we used the same method to build an application based on amazon-fine-food-review data. We first classified the score data by 1, 2, 3, 4, 5. However, the result is less than satisfactory. We did improvement by categorizing the score into good reviews and bad reviews base on the scores. 1,2 as dissatisfied and 4,5 as satisfied. And the result showed it indeed made an improvement on our model. We also printed word clouds of the two categories to visualize the key words in satisfied reviews and dissatisfied reviews. We think we could improve our model in the future by reducing the influence of the common neutral words which appear in both word clouds.