Data Mining Capstone Task 6

1. Abstract

In this task, I explored multiple ways of doing classification, using algorithms provided in python sklearn library. I also explored multiple kinds of features for classification, and ended up with LDA model with 300 topics, along with additional information as the feature, for a blended classifier of 6 different ensemble classifiers.

2. Implementations

2.0 Text Representations

In this task, I have tried multiple text representations to feed the classifiers as the features. These include two major categories:

1. LDA topic modeling using unigrams
2. Uni- / bi-gram term frequency of top terms with high mutual information

2.0.1 LDA topic modeling using unigrams

In this set of experiments, I tried following ways of preprocessing texts and doing LDA topic extractions:

1. Only doing stopword removal before LDA, and feed the LDA model with TFIDF-modified features. Essentially, I used smoothened IDF and sublinear TF modifiers. Topics number varies among 10, 30, 40, 100, 300;
2. Doing stopword removal and stemming before LDA, and doing the same as above subsequently.

It is worthwhile to mention that even before stopword removal and stemming, it is necessary to do special token removal, such as tokens like *&#160;* which is pretty frequent in the data and probably is review separator.

Results and Analysis

With the same classifier (RandomForestClassifier), on the train set, it is shown from cross validation that only doing stopword removal with 300 topics extracted yielded the best F1 score.

This result is contradictory to some colleagues pointed out on the forum, that ~30 or 40 topics would suffice to produce a good result. The only reason that I can think of to explain 300 topics doing well, is that maybe more topics correspond to detailed topic modeling, which may not only include dish reviews or general comments, but also certain topics about, for example, bathrooms or something like that, which might be a good indicator of hygiene condition.

But according to the feature_importances_ result of RandomForestClassifier, it is more obvious that general comments and comments of dishes are of higher importance to distinguish restaurants.

2.0.2 Uni- / bi-gram term frequency of top terms with high mutual information

In this set of experiments, I first do stopword removal and stemming on the original text input, and then compute the count of each unigram / bigram terms in each text (reviews of one restaurant). Then based on these counts, I computed the mutual information of each term with the labels of the restaurants. And I then used the count of the top 500 / 1000 terms in terms of their mutual information value, as part of the features (with rest being the additional information).

Results and Analysis

The results of this set of experiments are generally slightly inferior to the one got by 300-topic result.

From the result of RandomForestClassifier, the most important features (terms) are shown as followed. It somehow shows that dishes are still very important at distinguishing restaurants' hygiene conditions.

In fact, when looking at the mutual information table, it is even more surprising to see words like "dirty" or "clean" are not among the 1000 top mutual information terms – i.e. they are not as correlated to the labels as dishes as shown below.

If we consider why, we may consider that customers are more likely to discuss dishes if the restaurant is acceptably clean, and for such restaurants, they are more likely to pass the hygiene check.

rating
review_count
noodl
steam
zipcode_98105
crisp
zipcode_98101
tofu
chees
cuisine_Fast Food

Figure: most important features of MI selected terms combined with additional information

noodl
tofu
steam
great price
rice
pork
shrimp
deal
curri
think

Figure: top terms with highest MI

2.1 Additional Information

In fact, the initial features that I used are from additional information – cuisines, zipcodes, review count and average rating.

For cuisines, I mapped each cuisine to a feature column, and mark 1 to those restaurants of such cuisines, and 0 for otherwise. Therefore, one restaurant can have multiple cuisine columns with value 1.

For zipcodes, it is similar to cuisines, except that restaurants only have one zipcode columns having value 1.

From the feature_importance output of RandomForestClassifier, it is evident that rating and review counts are the top-2 most important features.

In fact, using multiple ensemble classifiers blended together, I achieved > 0.55 F1 score on the test data already, using additional information as the features only.

Based on these additional information and 300-topic features, I achieved 0.5559 as my best submission so far.

2.2 Algorithms and Best Performing Method

The toolkit I used is python with libraries *gensim, sklearn, numpy* and so on.

I applied following algorithms of different parameters:

SGDClassifier, RandomForestClassifier, ExtraTreesClassifier, GradientBoostingClassifier, and LogisticRegression (used to blend all other classifiers, inspired by: http://mlwave.com/kaggle-ensembling-guide/)

Best Performing Method

The best performing method is resulted from the blended classifers, with each of them having 100 estimators, and default configurations as provided by the python *sklearn* library.

The text is, as mentioned in 2.0, preprocessed with stopword removal only, and only unigrams are then fed to be modified by TFIDF, and then fed to LDA to produce 300 topics.

The probabilities of each topic are assigned to each corpus (reviews of one single restaurant), thereby forming a 300-column features.

Also, additional information are used to form the rest 131 feature columns, after proper modifications as mentioned in 2.1.

One Over-fitting Method

In fact, I'd like to mention that when using SGDClassifier, i.e. stochastic gradient descent (SGD) learning algorithm, I achieved ~0.74 on the train set with cross validation, but only ~0.52 on the test set.

The parameters for 0.74 SGDClassifier is obviously making the model overfitting to the train set. At the same time, the test set is highly skewed. That's probably why this method under this parameter failed to achieve a good result.