Task 6: Hygiene Prediction

Xiaoming Ji (xj9)

Overview

In this task, I'm going to predict whether a set of restaurants will pass the public health inspection tests given the corresponding Yelp text reviews along with some additional information such as the locations and cuisines offered in these restaurants. Making a prediction about an unobserved attribute using data mining techniques represents a wide range of important applications of data mining.

I evaluated term frequency, TFIDF, Topic Model and Document Embedding text representations, together with Naive Bayes, Logistic Linear Regression, SVM and XGBoosting classification models, in order to find the best combinations. Interestingly, simple TFIDF with Naive Bayes gives me the best results with F1 score of **0.702**. I also found that just by using the additional info of restaurant without review, I can also achieve F1 score of **0.654**.

Data Preparation

For this task, I have the dataset (**hygiene.dat**) that contains concatenated reviews for 13299 restaurants. A label file (**hygiene.dat.labels**) have the first 546 restaurants' binary labels, where a 0 indicates that the restaurant has passed the latest public health inspection test, and 1 means that the restaurant has failed the test. An 3rd file (**hygiene.dat.additional**) provides additional information regarding a restaurant.

For the review dataset, I applied the preprocessing steps as,

- Tokenization: Split the sentences into words. Lowercase the words and remove punctuation. Words that have fewer than 2 characters are removed (through gensim.utils.simple_preprocess).
- All stopwords are removed (stopwards defined in gensim.parsing.preprocessing.STOPWORDS)
- Words are lemmatized. Words in third person are changed to first person and verbs in past and future tenses are changed into present (by nltk.stem.WordNetLemmatizer).
- Words are stemmed. Words are reduced to their root form (by nltk.stem.SnowballStemmer).

To leverage extra info in **hygiene.dat.additional**, I applied the following steps,

- Cuisines: vectorized by using CountVectorizer (in sklearn). This give me 117-size cuisine vector for each restaurant.
- Zipcode: vectorized by using CountVectorizer (in sklearn). This give me 30-size zip code vector for each restaurant.
- Number of reviews: make it a 1-size vector.
- Average rating: make it a 1-size vector.

Feature Engineering

I have 2 major feature sets for this classification tasks: reviews and additional information of the restaurant. In my dataset, all reviews of a restaurant are concatenated as one big text and I used the following text embedding methods to represent these reviews.

- **Word Count**: this is basic bag-of-words approach that represents a document by counting word frequency. *CountVectorizer* (in sklearn) is used for this task.
- **TF-IDF**: a bag-of-words approach that uses TF-IDF algorithm to calculate weight of a word. Besides the regular 1-gram TF-IDF, I also evaluated (1,3) word level n-gram. The feature/vocabulary size is limited to 10,000. *TfidfVectorizer* (in sklearn) is used for this task.
- **Topic Model**: I use LDA algorithm (LdaMallet in gensim) to mine 200 topics and then calculate the topic distribution of each restaurant. This gives me a 200-size vector to represent each restaurant and the weight is the distribution value of each topic for this doc.
- **Doc2Vec**: similar to word2vec algorithm, Doc2Vec (as described in [1])[i] learn paragraph and document embeddings via the distributed memory and distributed bag of words models. I used doc2vec (in gensim) to get 200-size vector for each restaurant.
- **TopWord**: similar to TF-IDF representation, but I limit the vocabulary size to 200.

- **Topic Model + TopWord** (**TT** for short): combines [1]topic model and TopWord representations and this give me 400-size vector for each restaurant.
- **Doc2Vec + TopWord** (**DT** for short)**:** combines Doc2Vec and TopWord representations and this give me 400-size vector for each restaurant.

For the additional information of restaurant, I make 2 features,

- **MetaInfo**: combines all additional info of a restaurant. E.g., cuisines, zipcode, number of reviews and average rating. This gives me a 149-size vector for each restaurant.
- **Cuisine+Zipcode** (**CZ** for short): combines cuisines and zipcode. This gives me a 147-size vector for each restaurant.

Classification Models

Since the positive and negative class of this classification problem is imbalanced, I use F1 score of 5-fold CV to evaluate my model. The F1 measure will be based on the macro-averages of precision and recall. I also used GridSearchCV and RandomizedSearchCV (in sklearn) to find the optimal parameters for each model (implemented by sklearn and XGBoosting) using the following search spaces.

- **Naive Bayes**[2]: default parameters.
- **Linear Regression**: penalty = ['l1', 'l2'], C = [10, 1, 0.1, 0.01]
- **Support Vector Machine (SVM for short)**: kernel = ['linear', 'rbf'], C = [0.1, 1, 10, 100], gamma = np.logspace(-2, 2, 5)
- **Random Forest**: n_estimators = [100, 200, 500], max_features = [None, 0.25, 0.5, 0.75], max_depth = [None, 5, 10], min_samples_leaf = [0.0005, 0.01, 0.05, 0.1], min_samples_split = [2, 5, 10]
- **XGBoosting**: min_child_weight = [1, 5, 10], gamma = [0.5, 1, 1.5, 2, 5], subsample = [0.6, 0.8, 1.0], colsample_bytree = [0.6, 0.8, 1.0], max_depth = [3, 4, 5]

I evaluated these models against all review and additional information combinations. It takes about 8 hours in my workstation (Xeon 2.0G 4 Core CPU and 32G Memory) to complete this task. The results are illustrated in table 1.

Features\Models	Naive Bayes	Linear Regression	SVM	Random Forest	XGBoosting
MetaInfo	0.606	0.572	0.629	**0.644**	0.641
Cusine + Zipcode (CZ)	0.627	0.642	0.642	**0.654**	0.646
WordCount	0.662	0.649	**0.671**	0.648	0.635
WordCount + MetaInfo	0.662	0.650	**0.669**	0.633	0.658
WordCount + CZ	0.656	0.656	**0.670**	**0.666**	0.646
TF-IDF	**0.679**	0.652	0.667	0.641	0.639
TF-IDF + MetaInfo	**0.702**	0.572	0.638	0.653	0.646
TF-IDF + CZ	**0.674**	**0.667**	0.640	0.638	0.630
TF-IDF (1,3)-gram	0.649	0.652	**0.667**	0.615	0.618
TF-IDF (1,3)-gram + MetaInfo	**0.691**	0.572	0.646	0.633	0.646
TF-IDF (1,3)-gram + CZ	**0.666**	0.654	0.649	0.641	0.649
Topic Model	0.654	0.617	**0.665**	0.586	0.591
Topic Model + MetaInfo	0.614	0.566	**0.628**	0.592	0.593
Topic Model + CZ	0.638	**0.645**	0.637	0.593	0.599
Doc2Vec	n/a	0.580	**0.672**	0.621	0.617
Doc2Vec + MetaInfo	n/a	0.596	**0.671**	0.619	0.609
Doc2Vec + CZ	n/a	0.592	**0.667**	0.635	0.599
TT	0.652	0.634	**0.680**	0.623	0.598
TT+ MetaInfo	0.607	0.575	**0.629**	0.627	0.600
TT + CZ	0.648	0.647	**0.650**	0.611	0.603
DT	n/a	0.575	**0.669**	0.644	0.615
DT + MetaInfo	n/a	0.603	**0.671**	0.617	0.642
DT + CZ	n/a	0.594	**0.674**	0.630	0.617

Table 1: F1 Scores for each Feature and Model Combinations

[1] Combines mean use numpy.hstack() to stack multiple vectors in one vector.

[2] sklearn implementation of Naive Bayes doesn't work on negative values, this make my doc2vec representation doesn't work with this model.

Surprisingly, simple representation and the simplest model (**TF-IDF + MetaInfo** with **Naive Bayes**) is the winner and give me the best test F1 score of **0.702**. Naive Bayes model also achieved many best scores among different feature combinations, including the runner-up **TF-IDF (1,3)-gram** + **MetaInfo** with F1 score of **0.691**. It is also the fastest model to run. **TT** with **SVM** get the 3rd-place with F1 score of **0.68**.

This is surprising because many sophisticated text representation and models perform badly. The reasons for this result I believe are:

- Combine more feature may not always help because this could also add more noises.
- More sophisticated models (for example: random forest or XGBoosting) could overfit the data and doesn't work well in my task.

This proved that there is no best model or presentation for any machine learning task.

Discussion

For a classification problem, this result (F1 score: 0.702) isn't very impressive. In previous related work [2] [ii], the authors achieved 82.68% accuracy in similar task using TFIDF (1,2)-gram presentation with SVM model. [4] [iii]did even better. However, since the real data is imbalanced, it is not appropriate to use accuracy measure to evaluate the model. Even with accuracy measurement, my best model can only get 63.36%. As discussed in [3][iv], [2]'s result is overly optimistic due to the extreme imbalanced sampling strategy.

Another important finding is how effective we can use reviews to predict hygienic condition. I intentionally evaluated the models against the additional information without review. As highlight in table 1, **Cuisine** + **Zipcode** alone with **Random Forest** can give me F1 score of **0.654**. To compare, best results from review only feature is **TopicModel** + **TopWord** with **SVM** and this gives me F1 score of **0.68**. No significant difference. Given cuisine and zipcode information of a restaurant is easier to acquire, it is doubtful that we can effectively infer hygienic condition from reviews.

I also tried other SOTA text classification techniques. For example, I used FastText and Flair NLP package to build more sophisticated text embedding and deep neural network-based classification. But the result is disappointing and F1 score is below 0.6. But I have to admit that I don't spend much time to fine-tune these models and this could be something I can explore in the future.

Conclusion

In this assignment, I explored different ways of representing restaurant reviews and used many popular classification models to predict the hygienic condition of a restaurant. I found out that the best approach is achieved by simple representation and model. I also analyzed the results and challenged the idea of using review data as effective way to make such prediction.

References

[i][1] Quoc Le and Tomas Mikolov. Distributed Representations of Sentences and Documents.

[ii][2] Jun Seok Kang, Polina Kuznetsova, Michael Luca, and Yejin Choi. Where not to eat? improving public policy by predicting hygiene inspections using online reviews.

[iii][3] Kristen M. Altenburger and Daniel E. Ho. Is Yelp Actually Cleaning Up the Restaurant Industry? A Re-Analysis on the Relative Usefulness of Consumer Reviews.

[iv][4] Samantha Wong, Hamidreza Chinaei, and Frank Rudzicz. Predicting health inspection results from online restaurant reviews