Fall 2015 ILS Z534 Search Project Report

YELP DATASET CHALLENGE

Team Members

Huzefa Yousuf Dargahwala (hydargah@indiana.edu)
Kamalkanth Chowdary Adusumilli (kadusumi@indiana.edu)
Rama Raghava Reddy Arvabhumi (ramarvab@indiana.edu)
Nipurn Doshi (nidoshi@indiana.edu)
Yang Yang (yang266@indiana.edu)

Under the guidance of
Professor Xiaozhong Liu
at
School of Informatics and Computing
Indiana University Bloomington

Introduction

The yelp dataset includes information about different businesses in 10 cities across 4 countries. In this project, we use this dataset to 1) predict the label of a business from review and tip text and 2) predict the rating and helpfulness of a review from its text. Accurate classification of a business into categories can help Yelp to improve business suggestions to the Yelpers. Predicting the rating and helpfulness of a review are useful to Yelp and the customers, as new customers will receive handy inputs about a business from reviews written by customers who visited earlier.

Task 1: Predicting the categories of a business from review and tip text

We implemented both the approaches, suggested to us, Machine Learning (ML) Approach and Information Retrieval (IR) Approach. We even tried to attack the problem in two ways using the IR approach.

A. Machine Learning Approach

Data Preprocessing

The data given was clean and did not have any discrepancies, which made working with it easier. The only preprocessing done was removing stop words and converting all text to lowercase.

For each business, its reviews and tips were combined into one long string. This string was then used to create the features to be used for training the model.

Features and Class Labels

Each business was represented by a long string of its reviews and tips. These were then converted into tf-idf scores of each word in the vector space model and were used as features to be given as input to an appropriate machine learning algorithm.

The problem at hand is a good example of a multilabel classification problem, which means each business can belong to multiple categories. To achieve this, we use the One vs Rest approach. We train as many classifiers as there are class labels, and each classifier learns a model for one label, and predicts whether or not a feature vector belongs to that target label. In the end, we combine the outputs of all classifiers, and calculate the applicable evaluation metrics like precision, recall and F1 score.

Algorithms

The data was split in a 60:40 manner for training and testing respectively. The partitioning was done using Stratified Sampling to ensure that the model will not be affected by the class imbalance problem.

Multinomial Naive Bayes (MNB) and *Linear Support Vector Classifier* (Linear SVC) were used for learning models from the aforementioned feature vectors. We chose the above two classifiers for reasons of simplicity. They gave us good results along with faster training times.

Results

MNB performed poorly on the task but Linear SVC gave excellent results. Maximum precision reached for some categories was 1 but certain categories were always giving a precision of 0. The probable reason for it will be discussed in the Analysis section.

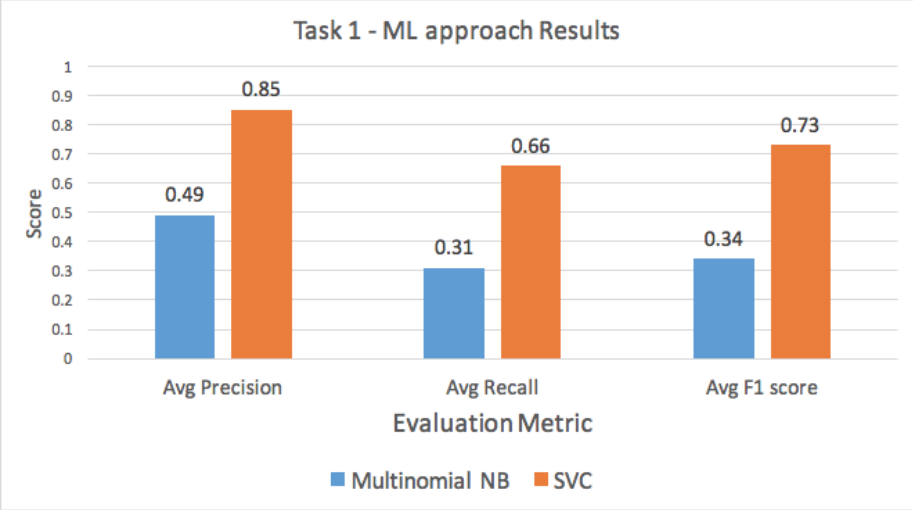

Fig 1: Average evaluation metrics for Multinomial NB and Linear SVC

We will now only discuss the results that we got from Linear SVC. The ROC curves below show most of the labels being close to the the top and left margins which shows high TP rate.

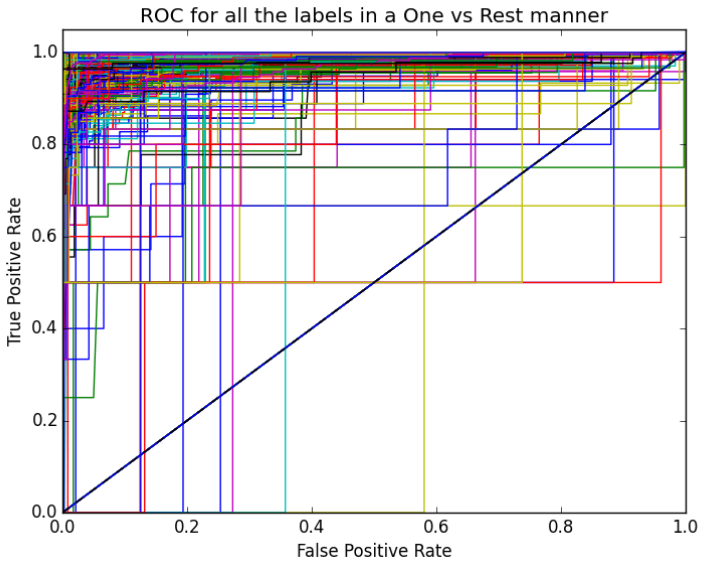

Fig 2: ROC curves for all the class labels after classification with Linear SVC

To gain a better perspective of the results, we plot the "average" ROC curve and report the Area Under the Curve (AUC) to get a general idea of how the Linear SVC performs. The larger the AUC, the better our model performs at maximizing the true positives while minimizing the false positives.

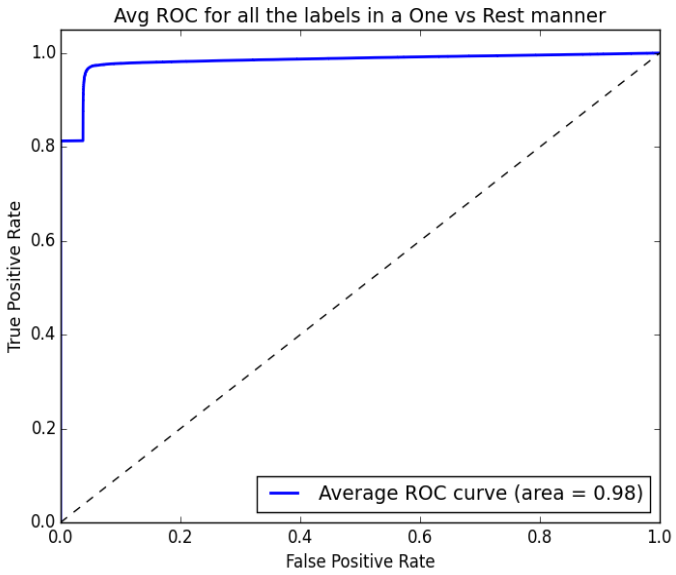

Fig 3: Average ROC for Linear SVC

B. Information Retrieval Approaches

(i) Approach One

We use Lucene Java API and Mongodb Java API in this approach.

For data manipulation, we chose Mongodb for JSON files. We first build three Mongodb collections, one each for business, review and tip files, and we then build indexes for these collections to speed up the search process. Next, we build a new collection "test_set". It includes the business information, reviews and tips for each business_id. It has 61184 records. Each record represents a business with the reviews and tips combined together. Then, we use lucene to build index for "test_set" collection based on the fields business_id, reviews and tips.

For each category, we break the query and simply take each term as a query. For example, *for "Indian" category, query can be: reviews: "Indian" , tips: "Indian".* We got 910 categories after processing the categories data in the test_set.

Each business_id already has few categories assigned to it. We will use this information as the ground truth file. For example, if business_id1 has category [category2], then in our ground truth file we will have the following data:

business_id1	category1	0
business_id1	category2	1

For each query, we search both the fields “review” and “tip”, and find the top 10 related business_id’s based on the scores. From those top 10 businesses, we can find the recommended categories. We use four different algorithms to calculate the scores: Vector Space Model, BM25, Language Model (Dirichlet Smoothing), Language Model (Jelinek Mercer Smoothing). We compare the results with the ground truth file.

Evaluation metrics

The precision, recall and F1 score for implementing different algorithms for review and tip fields are shown below in the following tables.

Review

	BM25	LM(D)	LM(J)	VSM
P	0.4361	0.4396	0.3637	0.3637
R	0.1281	0.1230	0.1133	0.1133
F1	0.1244	0.1228	0.1071	0.1071

Tip

	BM25	LM(D)	LM(J)	VSM
P	0.2367	0.2575	0.2192	0.2192
R	0.3067	0.0366	0.0343	0.0343
F1	0.0462	0.0472	0.0430	0.0430

LM(D): Language Model with Dirichlet Smoothing
LM(J): Language Model with Jelinek Mercer Smoothing

As shown in the table above, the scores for the reviews are higher than the scores for the tip field. That’s because there’s more review data than the tip data in the original data set. Among all the four algorithms, Language Model with Dirichlet Smoothing gives the highest score. The highest precision score is 0.4396, and recall is 0.1230 for reviews. However the recall is very low. So, we implement the query expansion in the next IR approach.

(ii) Approach Two

As shown in Figure 4, we split the data into two parts, 60% training data and 40% test data. We then index both training and test datasets on category, reviews and tips. We perform query

expansion based on the training data. We predict the categories for the business in the test data based on reviews and tips data.

For the training data, we first find all the reviews and tips for each category and combine them as a field. So, each document is a category with a long review_tip text. Then, we use lucene to build an index on these two fields. For each category we find the top 10 words based on tf-idf scores. We use these 10 words as the queries.

For the test data, we extract the business ID, and all the reviews and tips for each business. Each document is a business ID with long review_tip text. We index those two fields using lucene. Then, we use the queries generated from training data, and search through the test index, and predict a category for each business ID.

As every business already has few categories associated with it, we can use this to check the precision, recall and F1 score.

We use BM25 for checking the precision, recall and F1 scores. The results are shown below:

Number of Features	Precision	Recall	F1 Score
5	0.4106985	0.22550444	0.2693946
10	0.4224871	0.24187267	0.2816566
50	0.4307334	0.26838273	0.2980597

If the number of features passed to a query increased, the precision and recall values increased marginally.

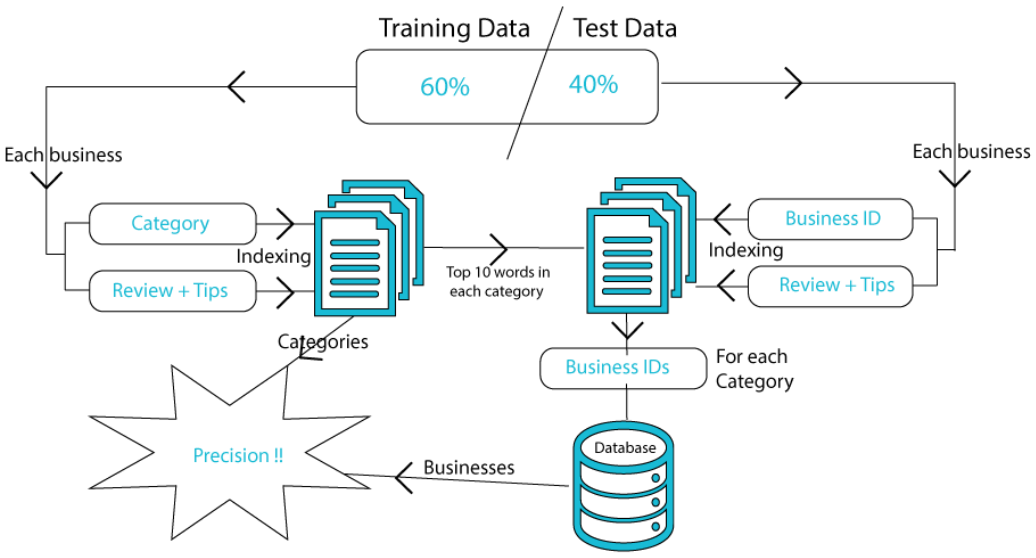

Fig 4: Flow Chart represents the process followed in predicting the categories from test data.

Analysis of Task 1

In the Machine learning approach, the results achieved were very good but they could have been better. Some categories were giving a precision of 0. This was because, for few categories, there were very few training examples. Some categories had only one training example. This did not allow the classifier to capture all the variance in the data for that label.

We clearly see that the Machine learning approach performs much better than IR approach, the reasons are multiple. Firstly, we did not employ any relevance feedback mechanisms in our method. A group of our colleagues used Amazon Mechanical Turk to obtain feedback and got good results. Secondly, the query formulation techniques used are simple and straightforward. More sophisticated techniques for query formation may have proven more effective. Lastly, the analyzer used for the task may not have been able to remove misspelt words and improper sentence formation. The language used by all the users cannot be assumed to be grammatically or even syntactically correct.

Both Linear SVC and Multinomial NB are parameter dependent classifiers. We run both of them with their default parameters. There may exist a set of parameters for SVC that would have caused SVC to surpass the performance of NB. Also, SVC in general performs better when there is a lot of overlap in the features and the support vectors are not affected by it. Since many words appear in a lot of reviews, there may have been feature overlap in the problem at hand.

Comparison between ML and IR Approaches

	ML(Multinomial NB)	ML(SVC)	IR(50)
P	0.49	0.85	0.43
R	0.31	0.66	0.27
F1	0.34	0.73	0.30

Task 2

(a) Predicting the rating of a review from its text
(b) Predicting the helpfulness of review from its text

The techniques undertaken are similar for both tasks and we report the approach for both tasks together.

Data Preprocessing

The data did not have any missing values or any other such discrepancies. To the words which were preceded by "not", we appended the word "not" to the word to capture its negated meaning. Ex: not good will be not_good. All stopwords were removed, followed by lemmatization of the remaining words using the WordNet lemmatizer. Finally, we removed all the punctuation from the review text.

Features and Class Labels

For all the preprocessed reviews, we generate the tf-idf scores for each word in the vector space model and create feature vectors of these scores for each word. We then use the feature vectors to train the appropriate machine learning algorithm.

For Task 2.a, we used the ratings as the class labels, and for Task 2.b we aggregated all helpful numbers greater or equal to one as one. This makes it a binary classification problem and eliminates class imbalance to large extent.

Algorithms

Task 2 (a)

The task of predicting a rating can be viewed as a classification problem or as a regression problem. We undertook both.

When treating the problem as a classification task, we used Multinomial Naive Bayes (MNB) and Linear Support Vector Classifier (Linear SVC). As the dataset was very large, in order to ensure that the algorithms converge in a timely manner, we used Recursive Feature Elimination (RFE) with Linear SVC, with RFE set to remove 90% of the features at each iteration.

When treating the problem as a regression task, we used Classification and Regression Trees (CART) with various depths.

Task 2 (b)

Helpfulness prediction was a binary classification problem, and we used Linear SGD and Multinomial NB algorithms to complete the task.

Results

Task 2 (a)

We obtained impressive results when treating the problem, as a classification task. Below we see a good amount of AUC in the ROC curves for each label.

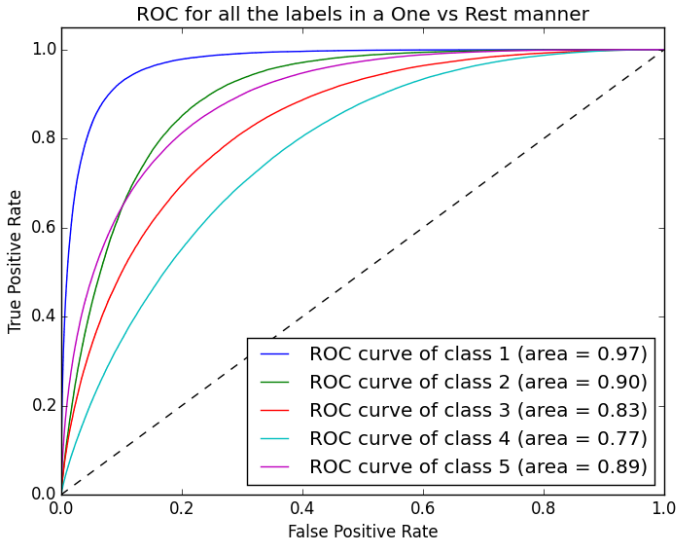

Fig 5 : ROC curves for each class label

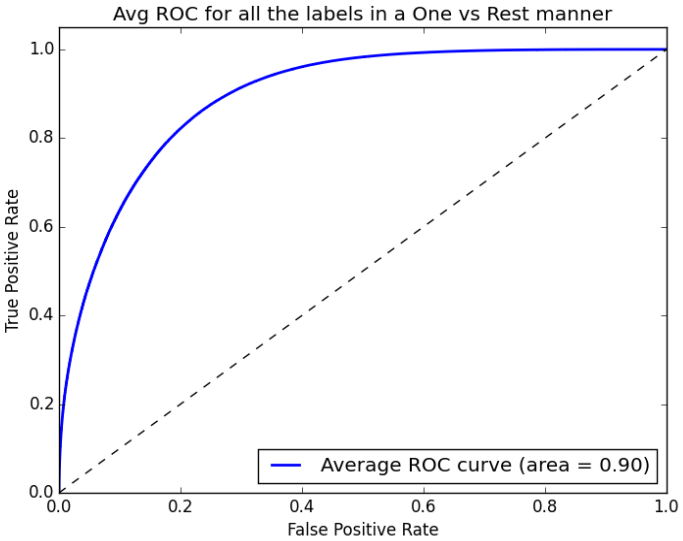

Fig 6 : Average ROC Curve for the task as a whole

CART performance on various evaluation metrics can be seen below. The metrics improve as the depth of the tree increases. However, training a tree with a depth of 15 was very time consuming.

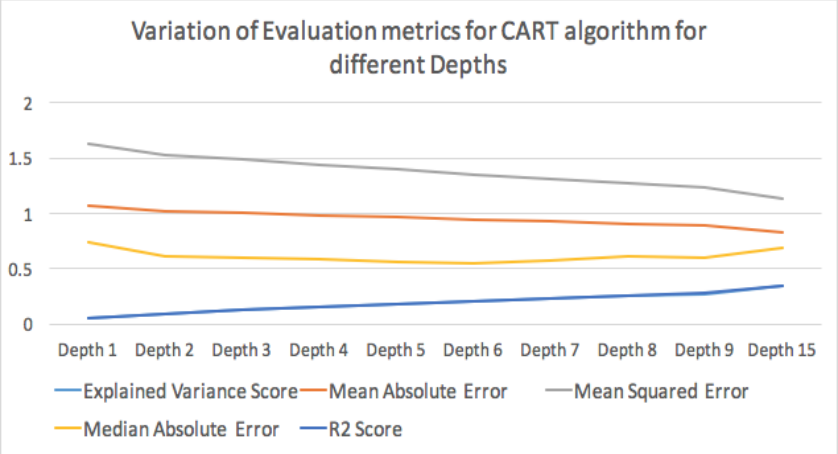

Fig 7: Change in evaluation metrics with increase in depth

Below are the results for CART algorithm with depth 15

Explained Variance Score	0.343
Mean Absolute Error	0.832
Mean Squared Error	1.13
Median Absolute Error	0.6957
R2 Score	0.343

Task 2 (b)

Both Multinomial NB and Linear SGD gave similar results when predicting the usefulness of a review based on it text. SGD gave marginally better results.

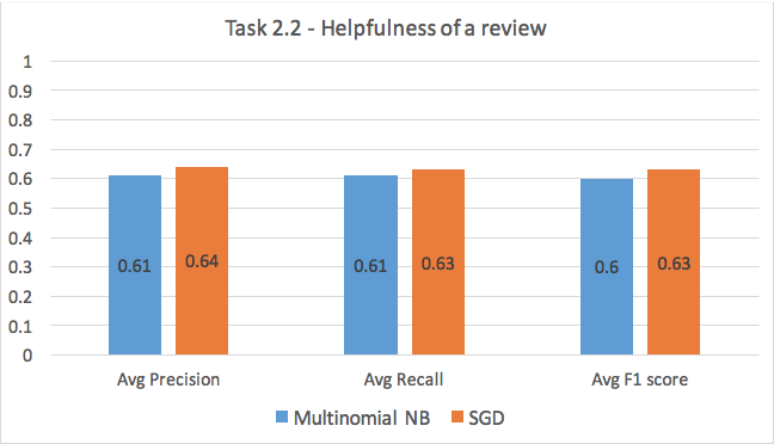

Fig 8: Performance comparison of Multinomial NB and SGD for Task 2b

The figure here shows the ROC curve for the helpfulness task, the results are not so impressive.

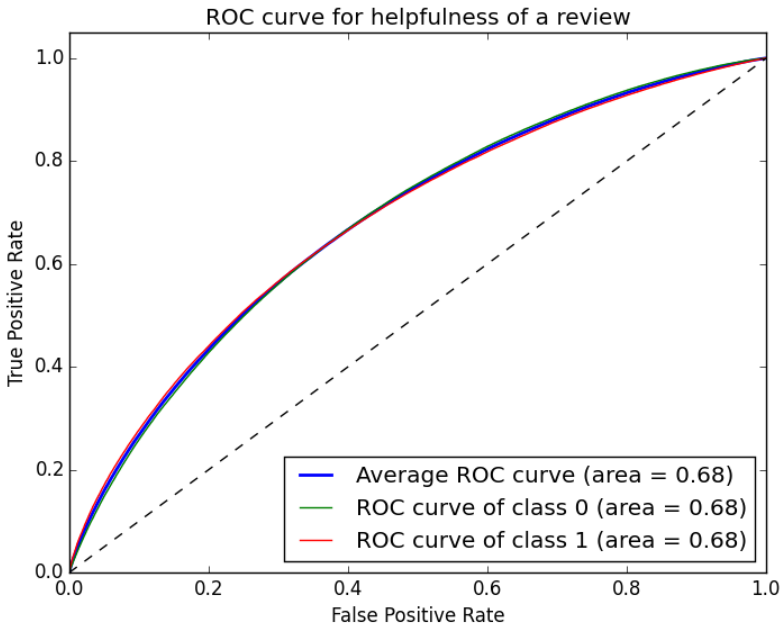

Fig 9 : ROC curve for SGD for Task 2b

Analysis of Task 2

For the rating prediction task, we added a custom feature in the data, ie: appending "not" to a negated word in an attempt to increase the performance of the classifiers, interestingly though, the performance of the classifier was degraded. We got an average precision of 0.6 and F1 score of 0.5. Both degraded from precision of 0.64 and F1 score of 0.52 of the original feature set. One reason for this may have been that there were not a lot of negated words in the feature space, also there could have been other features like positive and negative adjectives that describe the rating better. Ex: "good" vs "excellent" and "bad" vs "horrible".

Treating prediction of the rating as a classification task using Recursive Feature Elimination, with parameters set to remove 90% of the features to reduce the time taken to produce results, we get good results, may be because of the fact that the reviews contain adjectives that can be used to determine ratings decisively. This bolsters our conclusion in the previous paragraph. CART did not give actionable results with depths less than 15, however, the evaluation metrics kept getting better at each increased depth. Due to intractability on limited power machines, we were not able to produce results for depth greater than 15. CART may have outperformed Linear SVC if it were able to grow to its ideal depth.

One of the reasons we obtained good results is because of using Stratified Sampling of the data when splitting the data into training and test sets. This took care of the class imbalance problem, since, ratings 4 and 5 made up more than 50% of the data. Prediction of helpfulness with text data could have been better with proper tuning of the parameters.