ML1010 - Yelp Reviews

Machine Learning Pokémon: Durai Nachiappan, Iman Lau, Shabeeth Syed, Lijuan Yang, Mohammad Islam

Introduction

Our Jupyter Notebooks and code are available at https://github.com/stellarclass/ML1010-Yelp-Project.

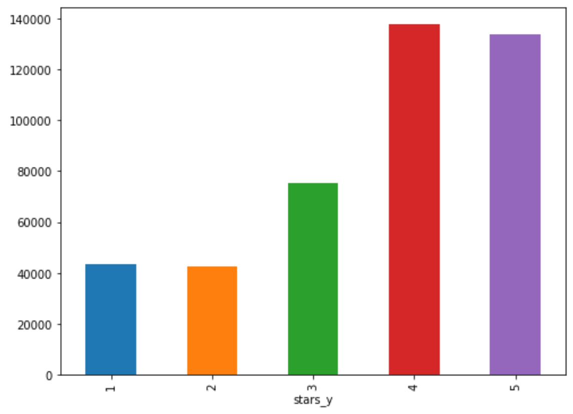
Our project uses data from Yelp, an online review site where users can rate various businesses. Most often, it is used for restaurants, although any business can be rated, from hotels to doctors. Yelp is widely used in North America but is available in numerous countries worldwide. This data is available through Kaggle, provided by Yelp.

Using natural language processing, we can take a large amount of unstructured review data and gather insights from the reviews. If one were to open a business in the city of Toronto, it would be useful to know what reviews tend to talk about. This would let a business owner know what to focus on when creating and running their business.

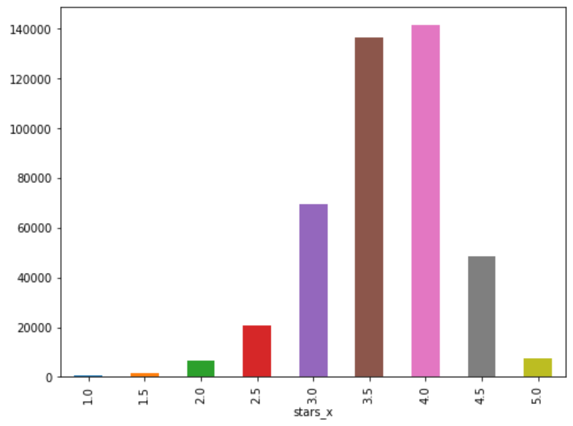
With that in mind, we created some models that would assist a Toronto business owner in this aspect. Our approaches were topic modelling and sentiment analysis. These machine learning approaches would help a business owner quickly find and understand relevant reviews in order to better their business or create a new one.

Exploratory Data Analysis

We briefly explored the data to see what it looked like. We saw that most reviews actually are 4 and 5 star reviews - almost twice as much as 1 - 3 star reviews combined.



We also can see that most businesses are about 3.5 to 4 stars.



This means that we could consider 3.5 to be "good."

Methodology

As mentioned in the introduction, we used topic modelling and sentiment analysis for our project. As this course focuses on Python, that was the language of choice. We used packages such as numpy, pandas, gensim, and scikit-learn.

Preprocessing

A fair amount of preprocessing was done to the dataset. Generally, natural language corpora require a great deal of preprocessing in order to be fed into machine learning models.

The Yelp dataset includes data from 11 metropolitan areas across 4 countries. As we only wanted to know about Toronto businesses, we needed to filter out non-Toronto data.

Due to memory limitations of our personal computers and the large size of the data, it was necessary to create a way to load the data into Python in stages. A script was written to read in the JSON file character by character, which also helped to removed data that was improperly formatted.

Once the data was written in, two pandas dataframes were joined together and then filtered for Toronto businesses.

After this, we could preprocess the Yelp review text to prepare it for modelling. The review text was normalize through removing stop words, forcing all words into lowercase, and removing non-alphanumeric characters. Then we could perform some feature engineering.

Feature Engineering

We used Bag of Words, Bag of n-grams (2-word phrases) and TF-IDF techniques as part of Feature engineering. These techniques helped us to identify the unique words in the reviews provided for Toronto's businesses.

In topic modelling, we created Bigrams and Trigrams. Gensim's Phrases model can build and implement the bigrams, trigrams, quadgrams and more. The two important arguments to Phrases are min_count and threshold. The higher the values of these parameters, the harder it is for words to be combined to bigrams. Once the bigrams model is ready, stopwords are removed and bigrams are created and lemmatized. The two main inputs to the LDA topic model are the dictionary and the corpus, which also were created.

Bag of Words was used in the sentiment analysis for the classifiers and ensemble methods. Neural networks are required to have the corpus converted into cleaned tokens and then encoded. Best performance happens when these tokens are also set to the same length, so shorter tokens are padded with 0's.

Topic Modelling

Latent Dirichlet Allocation (LDA) from Gensim package was used to perform Topic Modeling. Latent Dirichlet Allocation is the most common technique used in Topic Modeling, it is a generalized form of probabilistic latent semantic analysis (PLSA). In a nutshell, LDA considers each document as a collection of topics in a certain proportion, and each topic as a collection of keywords in a certain proportion. Once the algorithm is provided with the number of topics, all it does it to rearrange the topics distribution within the documents and keywords distribution within the topics to obtain a good composition of topic-keywords distribution. A topic is nothing but a collection of dominant keywords that are typical representatives. A Topic is identified by looking at the keywords.

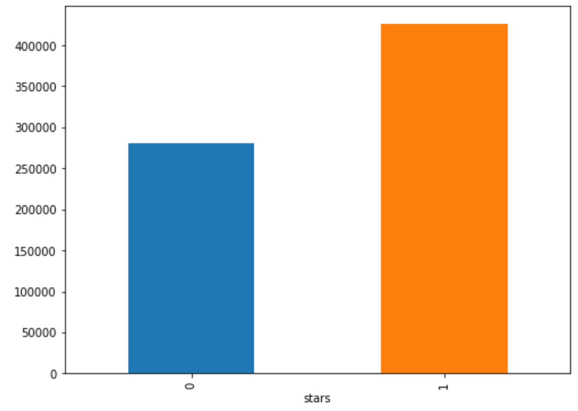
In addition to the corpus and dictionary,we need to provide the number of topics as well. Alpha and eta are hyperparameters that affect sparsity of the topics. According to the Gensim docs, both defaults to 1.0/num_topics prior. Chunksize is the number of documents to be used in each training chunk,and update_every determines how often the model parameters should be updated and passes is the total number of training passes.

Sentiment Analysis

A few different approaches were used to perform sentiment analysis. For standard classifiers and ensemble methods, we used and compared Naive Bayes, Logistic Regression, Stochastic Gradient Descent, Random Forest, and XGBoost.

Neural networks are systems that take input and process them through nodes and layers. The neural networks we tried, we used GLoVE word embeddings. We used the keras package for the neural networks.

In order to classify the reviews into positive or negative, reviews with a star rating of 3 or lower were considered negative with star ratings of 4 or 5 were positive. We can see the new distribution as such:



This is a roughly even distribution but is skewed slightly towards positive reviews.

Results

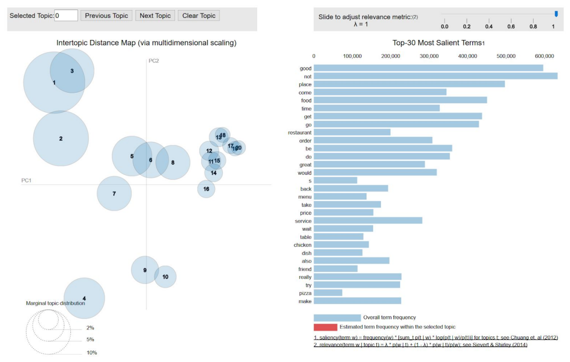
Topic Modelling

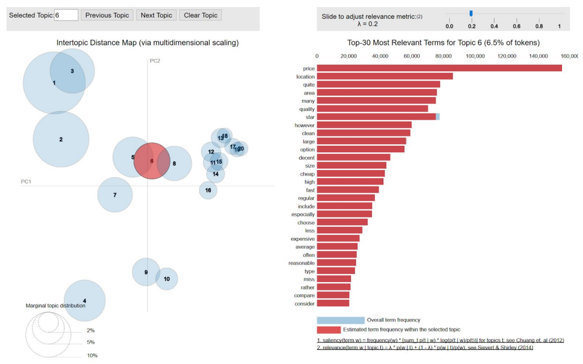
Model perplexity and topic coherence provide a convenient measure to judge how good a given topic model is. Topic Coherence is a measure used to evaluate topic models.Each such generated topic consists of words, and the topic coherence is applied to the top N words from the topic. It is defined as the average / median of the pairwise word-similarity scores of the words in the topic (e.g. PMI). A good model will generate coherent topics, and can be described by a short label, therefore this is what the topic coherence measure should capture

```
Perplexity:  -7.610246942214927

Coherence Score:
0.4172512043387064
```

We can also visualize the topics with an interactive chart (available in the Juptyer Notebooks).


Selected Topic: 0
Previous Topic
Next Topic
Clear Topic
Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
Marginal topic distribution
2%
5%
10%
Slide to adjust relevance metric:(2)
λ = 1
0.0
0.2
0.4
0.6
0.8
1
Top-30 Most Salient Terms1
0
100,000
200,000
300,000
400,000
500,000
600,000
good
not
place
come
food
time
get
go
restaurant
order
be
do
great
would
s
back
menu
take
price
service
wait
table
chicken
dish
also
friend
really
try
pizza
make
Overall term frequency
Estimated term frequency within the selected topic
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)


Selected Topic: 6
Previous Topic
Next Topic
Clear Topic
Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
Marginal topic distribution
2%
5%
10%
Slide to adjust relevance metric:(2)
λ = 0.2
0.0
0.2
0.4
0.6
0.8
1
Top-30 Most Relevant Terms for Topic 6 (6.5% of tokens)
0
20,000
40,000
60,000
80,000
100,000
120,000
140,000
160,000
price
location
quite
area
many
quality
star
however
clean
large
option
decent
size
cheap
high
fast
regular
include
especially
choose
less
expensive
average
often
reasonable
type
miss
rather
compare
consider
Overall term frequency
Estimated term frequency within the selected topic
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Sentiment Analysis

We can see various metrics of our sentiment analysis models in the following table:

Model Name	Accuracy	F1	Precision	Recall
Naive Bayes	0.8496	0.8817	0.8400	0.9278
Logistic Regression	0.8749	0.8980	0.8845	0.9119
Stochastic Gradient Descent				
Random Forest				

Unfortunately the remaining two models did not finish in time for this report.

We can see the accuracy of the neural networks in the following chart:

Neural Network Type	Accuracy
Convolutional Neural Network	0.39591462211969713
Long Short Term Modelr	0.39591462211969713
Recurrent Convolutional Neural Network	0.39591462211969713

As we can see, the neural networks don't have a noticeable improvement over regular classification systems. We can also see that there's a weird issue with the neural networks where the accuracy is the same for all the types that were attempted.

Discussion

We can see that overall, the most relevant topic for all reviews was "good." Since they are reviews, this makes a lot of sense - the most important thing in a review is to know whether or not a business was good. Many reviews on Yelp are restaurant/food based, and these also show up as common topic themes.

When we look closer into the topic clusters, for example topic 6, we can see that there's more concentration in different topics. In cluster 6, people are very concerned about price and location.

In the sentiment analysis, we can see that, without tuning, the standard classifiers perform admirably compared with untuned single-epoch neural networks.

Conclusion and Future Work

We can draw the following conclusions from our work:

- natural language processing is memory intensive
- Python is slow

For future work, we propose:

- better cleaning of data and stemming - some oddities showed up in the data and would benefit from more cleaning

- creating an interactive map to explore the topics and sentiments when attached to their respective businesses
- creating a dashboard/website where people can explore our interactive topic model chart
- fixing neural networks
- more layers and epochs in neural networks
- more hyperparameter tuning
- comparing sentiment analysis ROC curves

References

https://www.kaggle.com/yelp-dataset/yelp-dataset/home

http://blog.conceptnet.io/posts/2017/how-to-make-a-racist-ai-without-really-trying/

https://towardsdatascience.com/multi-class-text-classification-with-scikit-learn-12f1e60e0a9f

https://www.analyticsvidhya.com/blog/2018/04/a-comprehensive-guide-to-understand-and-implement-text-classification-in-python/

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

https://machinelearningmastery.com/prepare-text-data-deep-learning-keras/

https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/

https://rare-technologies.com/what-is-topic-coherence/

Appendix

Notes:

- Jupyter Notebooks were saved as PDFs and concatenated into this report as the appendix - they may or may not have titles but are named properly in the GitHub repo
- Some code is included that was ultimately not finished in time for this report (due to model run-time)