

Milestone II: Analyzing Customer Sentiments And Topics Of Food Delivery Platforms Through App Store Reviews

SIADS 694 & 695 (December 2021 - January 2022)

Mel Nguyen, Bojia Liu & Stanley Hung

Source Code: <https://github.com/mads-app-reviews-nlp/app-reviews-nlp>

Background

Food delivery apps have become an integral part of many people's daily lives. From the perspective of food delivery companies, understanding and catering to the needs of millions are no easy feat. At the same time, the food delivery business is extremely competitive, making the task to understand and measure a customer's "happiness meter" to the platform is even more imminent. In this project, we would like to leverage the power of natural language processing to achieve two goals. Firstly, how do customers feel about various food delivery platforms, and do these sentiments vary over time and locations? And secondly, what are the primary topics that customers write about on these reviews? Getting a low rating says very little about why that customer is deciding to give that rating, rather, we need to understand and mine the pain points that are driving the customers' decisions. And by understanding them, we can have a more insightful view of the strengths of different food delivery platforms, while also potentially identifying the pain points of each.

Data Sources

Our focus will be on food delivery platforms in Singapore & Australia. In each country, our team has curated a list of 3 to 4 food delivery apps, namely:

- Australia: DoorDash, UberEats, Grub, Deliveroo, and Menulog
- Singapore: foodpanda, Grab, Deliveroo

For each country, the dataset is constructed from the iOS and Android app stores for each respective country, and for the purpose of the milestone project, the ground truth labels will be based on customers' ratings. **In other words, customers whose ratings are high (above 3 stars) will be considered positive reviews, and vice versa reviews that are 1 or 2 stars will be considered negative.**

Below is an example of a customer review:



Ren Sack

★★★★★ January 14, 2022

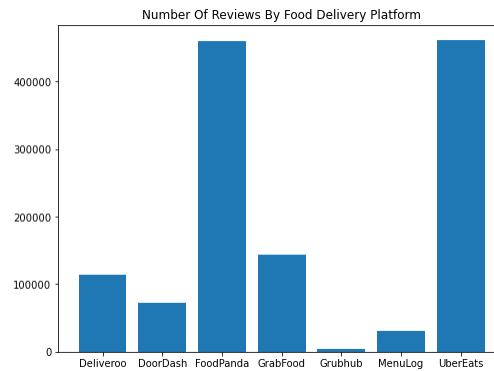


It was great once upon a time. Now, errr..... Payment error constantly. Been using the same credit card and spoke to the credit card company. According to them that they do get the charge but for some reason food panda decided to reverse the charges. I guess they don't want my money. Contacted support (I don't know why) to let them know the situation. No reply yet. Uninstalled.

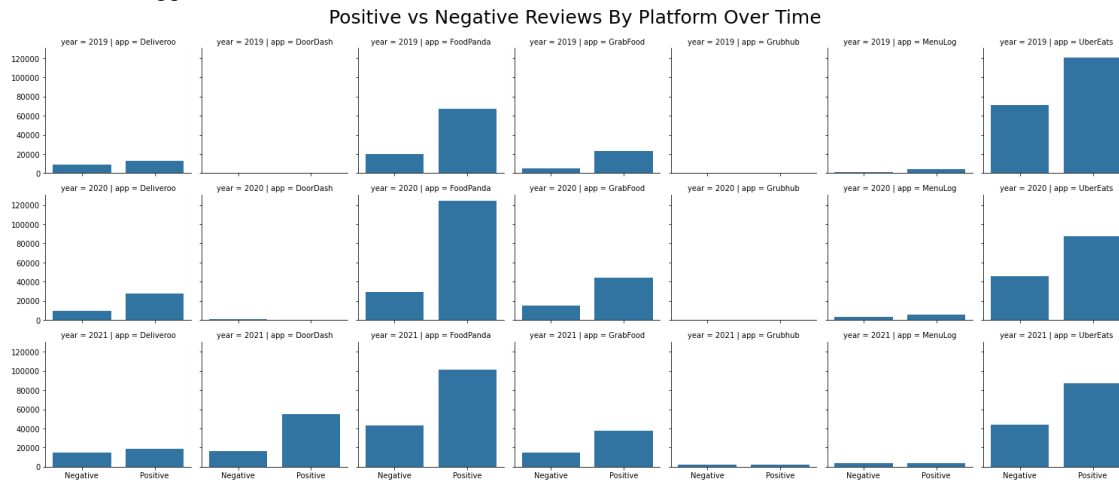
We use the `google_play_scraper` library to scrape the reviews from Google Store, and `app_store_scraper` to scrape the reviews from Apple Store. Both libraries allow specifying a `country` parameter to access the corresponding country's app store. However, in the raw data, we observe some duplicates for apps that are present in both countries (eg. Deliveroo operates in both Singapore and Australia). The impact is negligible, as the duplicates made up for less than 5% of the total reviews. The scraped reviews are then saved directly into pandas dataframe, which we subsequently separate into 2 Pickle (.pkl) dataset files, one for each country. Each dataset is comprised of the following information: when a review is made, the app/platform, rating, and the content of the review. In total, there are 659k reviews for Singapore, spanning from November 2013 to December 2021. Similarly, there are 626k reviews for Australia, spanning from September 2009 to December 2021.

The Landscape Of Reviews

Australia and Singapore are both relatively matured markets when it comes to food delivery platforms. Starting with just 1-2 players, both countries have seen massive growth in the need to order food and even groceries. The appearance of COVID-19 and subsequently nationwide lockdowns have spurred this growth even further, with food delivery being almost the sole alternative to dining out.

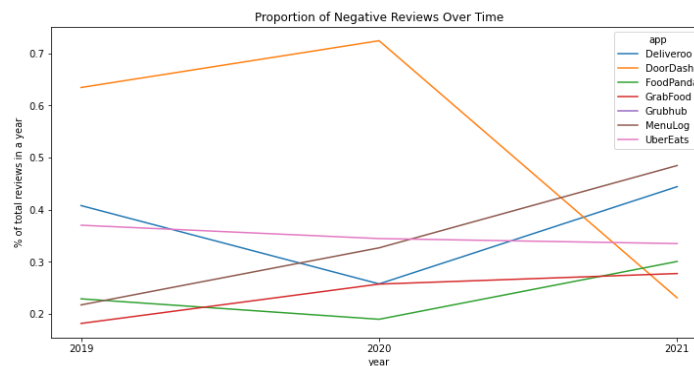


Foodpanda and UberEats are both well-established market leaders. Indeed, Foodpanda is one of the first food delivery services to operate in Singapore, while UberEats is one of the first in Australia. In the last 3 years, Singapore has seen an aggressive expansion of GrabFood, as well as Deliveroo. On the other hand, DoorDash, Grubhub, and Menulog have entered the Australian market, albeit much more recently -- GrubHub, for instance, only started in August 2021. Let us take a look at the breakdown of positive and negative over time of each app.



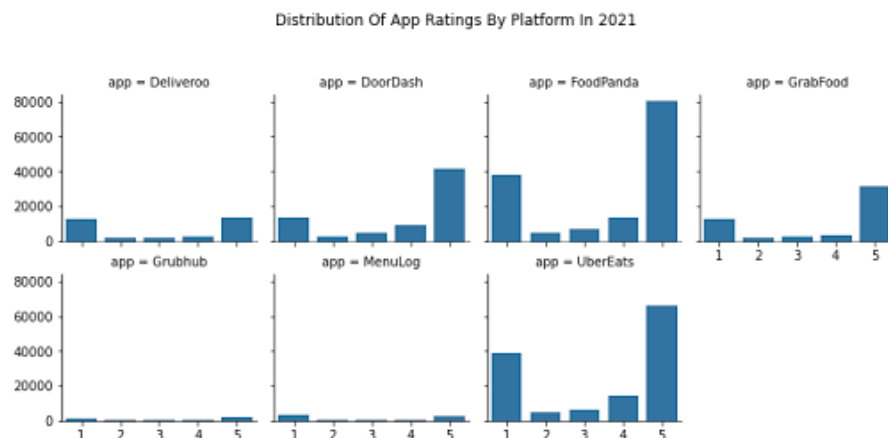
As soon as the reviews are broken down into positive and negative, we can see a few emerging patterns amongst the food delivery platforms. Notably, Foodpanda's number of positive reviews grew significantly between 2019 and 2020, but reduced between 2020 and 2021, replaced instead by more significant growth in negative reviews. UberEats, in contrast, saw a reduction in both positive and negative reviews over the same period. While this may not be a conclusive insight into the general popularity of each of these delivery platforms, it does show that the landscape of food reviews is by no means static -- instead, it is constantly changing as each food delivery platform develops new features or encounters new setbacks, as well as a shift in the general consumer appetite for food delivery as a whole. It's also worth pointing out that an app's time in the market is not a guarantee of an upward trend in positive reviews, as shown by both Foodpanda and UberEats above.

That brings us to the next point, which is, how has the general sentiment for each app changed over the years? To answer this question, we will look into the proportion of negative reviews for these platforms over time:



It is much clearer to see a sharp drop in the proportion of negative reviews for DoorDash, which is due to a significant increase in the volume of reviews, especially positive reviews in 2021. UberEats, interestingly, has seen the proportion of negative reviews trending down, as the overall volume of reviews decreases. Deliveroo and Foodpanda both saw an initial dip between 2019 and 2020 but subsequently rose back up in 2021, even exceeding 2019 levels. A product or customer experience team could further deep dive into

this by analyzing the changes that were rolled out to the app during this period, potentially coupling with the topic modeling results that would be covered in subsequent sections. Last but not least, we will take a more granular look into each platform's rating distribution in 2021.



Perhaps unsurprisingly, we see a highly polarized distribution of ratings, with reviews being either a 1 (worst) or a 5 (best). This could be due to the fact that users who are neither very unsatisfied nor very satisfied with an app would be less likely to write reviews than users who would take to the app store to write a complaint, or to express a very positive opinion about an app.

Sentiment Classification

Sentiment classification is the most appropriate supervised learning task for our reviews dataset. The applications of sentiment classification include, but are not limited to social network monitoring, estimating NPS (Net Promoter Score) score, evaluating product feedback, and streamlining customer support processes.

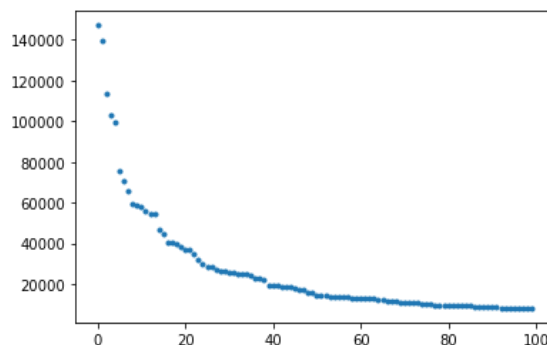
Our sentiment classification uses the reviews dataset documented above. We train 2 separate sets of models - one for Singapore and one for Australia, as each market could have its own nuances that should be captured separately. For each market, our dataset is divided into train and validation sets accordingly.

Naive Bayes & Logistic Regression

Both Naive Bayes and Logistic Regression are widely used, fast to compute, and simple to tune supervised learning algorithms, and hence they are ideal candidates as baseline models for our classification task. On the feature engineering side, both Bag Of Words and Word Embeddings methods are used to represent our dataset.

Bag Of Words (BoW)

A bag of words is constructed by first tokenizing the text data, and for this exercise, we first whitespace-split the sentences, then apply a regex to extract only alphanumeric tokens. This is, however, not the only way to tokenize a sentence. There could be more advanced methods, such as lemmatization, that could be applied to our dataset, but for the purpose of having a baseline model, we choose to keep the tokenization process relatively simple. Regardless of our approach, it is clear that the distribution of token frequencies follows Zipf's law:



This has two important implications. Firstly, we can remove many tokens that do not appear frequently, as they most likely would not impact predictive performance. Secondly, common tokens are most likely stopwords and should be removed as well. With this, we created a TF-IDF vectorizer that excludes English stopwords and has a minimum frequency of 500. We also include bigrams and unigrams into our vectorizer.

```
In [29]: vectorizer = TfidfVectorizer(stop_words='english', min_df=500, ngram_range=(1,2))
x_train_bow = vectorizer.fit_transform(X_train)
```

Once the vectorizer is ready and the train dataset is transformed, we started training our Naive Bayes and Logistic Regression classifiers.

The baseline results are shown below. Note that we have also run a few dummy classifiers (Uniform & Most Common), and confirmed that these algorithms outperformed these dummy classifiers.

Algorithm	Macro F1-Score	Accuracy
MultinomialNB	0.85	0.88
BernoulliNB	0.79	0.85
ComplementNB	0.83	0.86
Logistic Regression	0.85	0.87

The baseline results are quite promising, with both MultinomialNB and Logistic Regression achieving **0.85** on macro F1-score. This means that the bag-of-words approach is a rather solid approach for sentiment classification. It's also interesting to see that while MultinomialNB has a rather balanced number of false positives and false negatives, BernoulliNB and ComplementNB are different. BernoulliNB has a much higher number of false positives, while ComplementNB has a much higher number of false negatives.

Fine-tuning

For the BOW approach, fine-tuning the TF-IDF vectorizer could potentially improve the model performance. The table below summarizes various approaches to fine-tune the TF-IDF vectorizer on our best performing algorithm (Logistic Regression):

Approach	Impact On Logistic Regression F1-score
Reduce min_df to 100	Accuracy improved to 0.89 while F1 stays the same. Increased training time.
Skip stopwords removal	Accuracy improved to 0.9, and F1 improved to 0.88
Set max_df to 5000	No change in performance
Clip token frequency at 1 (using the 'binary' parameter)	No change in performance

Interestingly, omitting the stopwords removal step has resulted in the best performance improvement among other approaches. Reducing min_df in order to capture more tokens in the corpus also led to a bump in performance, but this is at the expense of a significantly longer training time.

With the TF-IDF tuning results, we also use this as the foundation to tune the 'C' parameter for logistic regression by performing a grid search over a range of C parameter values. The result, however, does not show a significant performance improvement with the tuned C parameter, suggesting that for this combination of feature engineering and supervised learning algorithms, tuning at the feature engineering stage would have a bigger impact on the final performance.

Word Embeddings

This section explores the use of word embeddings as feature extraction. We'll be working with dense representations of documents instead of the bag-of-words representations used earlier. To do this, we'll use the average (or mean) word vector of a document and classify from those representations. As a first step, we tokenize the reviews using the same regex logic. However, since we're going to be computing an average word vector, we remove stop words. Here, we'll use NLTK's list of English stop words. Since these words shouldn't affect our classification decision, we can remove them to avoid adding any noise they might cause.

Our word vectors are constructed using 4 different corpora - our entire text reviews corpus, as well as gensim's pre-trained corpora, namely wordvec-google-news-300, glove100, and glove300. The table below documents the performance result of each corpus when generating word embeddings and training them with Logistic Regression:

Corpus	F1-Score	Accuracy
Text Reviews Dataset	0.87	0.86
Google News 300	0.86	0.85
Glove 100	0.80	0.79
Glove 300	0.82	0.81

The performance overall is slightly underperforming as compared to BOW approach, but it is interesting to see that gensim's corpora perform almost just as well as using our dataset, especially the google-news-300 corpus. Having these texts represented as word vectors can also be useful for many other cases, including but not limited to topic modeling and document similarity. The promising performance on gensim's out-of-the-box corpus also means that companies who would like to set up NLP models without having access to a lot of training data can potentially leverage this approach.

SVM & Decision Trees, RandomForest

Support Vector Machine (SVM), and Random Forest are both simple supervised machine algorithms used for text classification purposes. In this section, we use these algorithms as well as the Decision Tree algorithm.

In the previous section, we focus on how performances vary when adapting different techniques - BoW and word embeddings. The focus of this section is to apply multiple ways of text pre-processing and compare the performance across methods and models. Only word embeddings with TD-IDF are used for this purpose.

Overall process

Pre-process review data with following methods respectively:

- Word lemmatization
- Stopwords removal
- Language detection

After which, splitting train, test datasets, vectorizing each dataset. Add columns to vectorized data as an additional feature when necessary, then fit each model with training data. Lastly, compare the F1-score of each model in different scenarios.

Baseline scenario

Feed vectorized training data directly to each model without any pre-processing.

Word Lemmatization

To reduce some noises caused by word forms, we want to group the inflected forms as a single word. The common way to do it is stemming and lemmatization. Both ways are tried out before we decide to use lemmatization. Examples shown below:

```
--- original ---
There isn't a platform for customers to leave their feedback Nor any responses when I send email.
--- stemming ---
there isnt a platform for custom to leav their feedback nor ani respons when i send email
--- lemmatisation ---
there isnt a platform for customer to leave their feedback nor any response when i send email
```

We can see that lemmatization gives more information on the context of words while stemming has some words truncated into incomplete syllables that do not even make sense. Thus we adopt the lemmatization method using NLTK building lemmatizer.

```
lem = nltk.stem.wordnet.WordNetLemmatizer()
```

Stopwords removal

Using NLTK's list of English stopwords.

Language detection

To filter out non-English reviews, language detection is used. In the previous section, it is done by SpaCy's *lang-detect* module. The problem with that is it takes a long time to run and has an unsatisfying detection result. For instance:

	review	rating	label	letter	lang
434	Driver deserves 6 stars & more.	5	positive	True	no
523	Good n excellent service.	5	positive	True	fr
638	餐厅接单后不出单，不能取消，一直在delay. 联系不上任何人，只能在线排队，体验非常非常差！	1	negative	True	zh-cn
688	Well done.	5	positive	True	it
706	Terrible	1	negative	True	ca

The detector mistakenly classifies some English reviews as other languages, thereby filtering out more reviews than it should have. To solve this issue, an alternative way of filtering out non-English words is used. In the UTF-8 encoding scheme (same for ASCII and other schemes), each character has a UTF-8 sequence number. All English alphabetic, digits and common punctuations, space, lie in a continuous range in the sequence. Knowing that range, filtering out non-English characters (like 'è') can be simply achieved by iterating through all characters in each string of review and comparing each character with the beginning and ending character of this range. If any character out of this range is detected, that review is taken as a non-English review.

The optimized language detector works fine, and the runtime has been drastically reduced. The non-English result is shown below:

😊😊😊😊😊	5	positive	True
餐很贵。	3	positive	True
方便快捷省钱。不赶时间做拼车很划算	5	positive	True
更新的版本还没尝试，之前的那个有时定位不准，造成过两次麻烦。	4	positive	True
搭车便宜多了，希望能够一直这样	5	positive	True

Feature Addition

So far we have only been using vectors from TF-IDF as features. To investigate if additional features will boost the performance, a new API is introduced. TextBlob is a Python library for processing textual data, which provides a simple API for diving into common natural language processing tasks such as sentiment analysis, classification, and more. We use the polarity API under the sentiment module. Polarity ranges from (-1,1). If polarity is >0, it is considered positive, <0 -is considered negative and ==0 is considered

neutral.

This polarity method alone has ~70% accuracy. We now use the polarity as one feature and concatenate that onto the TD-IDF vectorizer results.

Model Performance and Discussion

F1 score for each model in each scenario are shown below.

F1 Score	DecisionTree	RandomForest	SVC
Un-processed data	0.694	0.763	0.773
Lemmatization Only	0.698	0.767	0.778
Lemmatization and stopwords removal	0.723	0.739	0.737
Language detection	0.705	0.788	0.803
Add feature	0.745	0.806	0.824

The comparison indicates that lemmatization has basically no impact on model performance. Non-English review removal (i.e. language detection) barely increases the performance. As discussed before, it could be that non-English reviews have largely been removed already when we set `min_df` in the TF-IDF vectorizer to 500.

Adding features to original feature vectors does boost the performance.

However, one interesting finding is that removing stopwords has lowered the F1 score for Random Forest and SVC. See failure analysis for further discussion.

BERT

Deep Learning, or more specifically, BERT (Bidirectional Encoder Representations from Transformers) is increasingly gaining popularity in the NLP space. BERT-base consists of 12 transformer layers, each transformer layer takes in a list of token embeddings and produces the same number of embeddings with the same hidden size (or dimensions) on the output. The output of the final transformer layer is used as the feature of the sequence to feed a classifier.

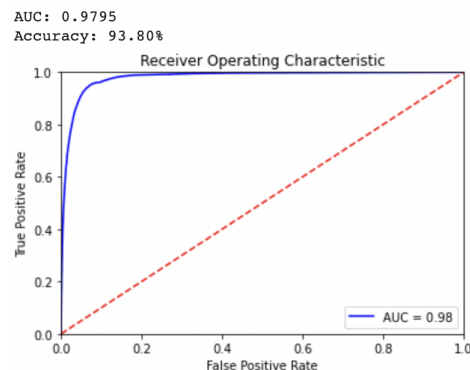
To fine-tune BERT to our reviews dataset, we start with tokenizing the text reviews, similar to previous approaches. The difference here is BERT has its own tokenizer, and thus instead of using regex, we invoke a BERT tokenizer from BERT's pre-trained *bert-base-uncased*. Note that BERT has a `max_length` of 512, which means longer reviews will need to be truncated; although this has not seemed to impact the final performance.

Once the tokenization is ready, we create an optimizer object to fine-tune the classifier later on. There are many parameters that can be fine-tuned, but we focus on two main parameters: batch size, and learning rate.

Finally, we train our BERT classifier for 2 epochs. In each epoch, we will train our model and evaluate its performance on the validation set through the following process:

- Load tokenized data
- Zero out gradients performed in the previous pass
- Perform a forward pass to compute logits and loss
- Perform a backward pass to compute gradients (`loss.backward()`)
- Clip the norm of the gradients to 1.0 to prevent exploding gradients
- Update the model's parameters & learning rate
- Compute loss and accuracy rate over the validation set

For the prediction step, the process is a lot simpler. We don't have to perform a backward pass; instead, we would only need to perform a forward pass and apply softmax to calculate probabilities.



The result is extremely promising. BERT outperformed our fine-tuned Logistic Regression, achieving an astonishing 0.98 accuracy and 0.93 macro F1-score. More importantly, this result is achieved by training on only 100k reviews, a much smaller dataset as compared to the original dataset that's trained on other algorithms. On the other hand, however, the training time also took significantly longer, with 1 epoch & 1 GPU taking about 1 hour to complete, as compared to other algorithms which only took a fraction of the time. Regardless, the result has demonstrated the potential of deep learning approaches on NLP tasks, including, but certainly not limited to sentiment analysis.

Failure Analysis

- The figure below shows a few examples of misclassifications under Logistic Regression:

```
In [40]: mis_class_logreg.sample(10).values

Out[40]: array([[ 'Good', 1, 0],
 [ 'Had a couple of glitches and got double charged but other than that fine',
   0, 1],
 [ 'Some resturents are spoiled the image of foodpanda. so Foodpanda should banned them otherwise people are betrayed by them..',
   0, 1],
 [ 'Rất chậm, kém trong xử lý vấn đề thẻ thanh toán của tôi bị lỗi',
   1, 0],
 [ 'Penghantarannya bagusss tapi sayang guna motor personal..semua sektor/service Penghantaran sepatutnya menyediakan kenderaan compan
   y..',
   0, 1],
 [ 'boring', 1, 0],
 [ 'No good', 1, 0],
 [ 'Food prices keep increase.', 0, 1],
 [ 'I have a question why the payment method via CREDIT/DEBIT CARD is not working?',
   0, 1],
 [ 'the ridera are very slpw 2 times the food arrived cold', 0, 1]],
 dtype=object)
```

We can see that our Singapore dataset contains many reviews that are not in English; however, this would not impact the model performance. To illustrate the impact of non-English reviews, we rerun the same algorithms on the dataset, but with non-English reviews filtered out, using SpaCy's *spacy_langdetect* module. While there seems to be a slight bump in F1-score for some algorithms, the increase in both F1-score and accuracy isn't significant, even though the language detector has found at least 30% of the reviews to be non-English. It's likely that many of the non-English tokens do not achieve the minimum frequency set during the TF-IDF stage, and get filtered out.

- For SVM & RandomForest algorithms, applying stopwords removal degraded the performance slightly. One possible explanation is that when removing some words from the list of common English stopwords, such as 'no' and 'not', the sentiment of a sentence can be completely changed. For example,

```
--- original ---
ordered food but never came. I do not appreciate this service.
--- remove stopwords ---
ordered food never came appreciate service
```

The original review which explicitly stated the disappointment towards this app has been twisted into a neutral, or even slightly positive tone.

- In some other misclassifications, we can see that certain reviews contain negation. For example, a customer may write “No good” instead of “Bad”. This is a potential downside of the Bag-of-words approach, where tokens are not considered in a particular context. Including bigrams and unigrams has mitigated some of these cases, but there are possibly more cases that are not captured.
- Last but not least, misclassifications could be due to a customer's way of ranking. This especially happens around the 3-star ratings, where a customer may not necessarily express a positive sentiment, but decides to give a high rating anyway. For instance:

```
It has been 2-3 weeks now I do not received any notifications on promotions etc. All setting for noti
fications is turn on. What else should I do?
```

This is a review in the test dataset that all three models fail to predict correctly. The actual label of this review is positive, however, by reading this, it is noticeable that this has a neutral to the negative tone. The problem behind this is that people sometimes express mixed feelings about an app yet give an acceptable rating generously, which creates a gap between the actual sentiment and rating. A properly labeled training dataset could help to mitigate this problem, although, in reality, it is challenging to create a domain-specific, labeled dataset without involving significant human effort.

Topic Modeling (Unsupervised Learning)

Over time, there were tons of reviews provided by the food delivery app user. As mentioned in the Data Source section, the number of reviews over the year is massive, there were a total of 659k reviews for Singapore and 626k reviews for Australia. When dealing with this amount of review, it is difficult to extract meaningful insights quickly and act on them. This is where topic modeling can offer a big help. With topic modeling, we will be able to extract key themes (topics) from all the reviews and assign the appropriate theme(s) to each topic. From there, we will be able to conduct some basic statistical analysis to see which theme is the dominant one within the review. As a result, this will help the app developers to understand the reviews quickly and start planning for the next step. (For example, resolving some issues if there is one, reporting key pain points to the company for improvement, etc.)

The unsupervised learning task used the same dataset as the supervised learning task. Please refer to the “Data Source” section for

more details about the dataset

Latent Dirichlet Association (LDA) and Non-Negative Matrix Factorization (NMF)

There are two common popular methods we can use for the topic modeling task, including Latent Dirichlet Association (LDA) and Non-Negative Matrix Factorization (NMF). For the project we used the LDA model due to a few reasons...

1. LDA assumes each document (review in our case) has multiple topics, while NMF calculates how well each document fits each topic, rather than assuming a document has multiple topics. For this project, a review could be related to multiple topics in reality.
2. LDA works better with longer texts, while NMF works best for shorter texts. In our dataset, approx. 14% of the reviews have more than 30 words. Therefore, we want to use a model that can model these reviews as well

In a nutshell, the analysis framework includes four steps, 1) Data pre-processing, 2) Vectorization, 3) model training and evaluation, 4) topic evaluation. Details below

1. Data pre-processing

To improve the performance as well as remove unnecessary tokens in the tokenization stage, we have done a few pre-processing steps on the test data using the Spacy library, detailed as follow

- Remove stop words and punctuation using Spacy default stop word list
- Remove common words/phrases (e.g the name of the food delivery app, emotional expression words like good, nice, ok, etc.)
- Lemmatization
- Lower casing all words

The overarching objective of the data pre-processing step is to remove words that are meaningless to the model in extracting topics (via stop words and common words removal), and also combine the words in the same token if they have the same meanings (achieve by lower casing and lemmatizing words)

2. Vectorization

Vectorization is a required tokenization step before we feed the data to LDA model training. Here TFIDF is used for the task. Compared to another vectorization approach – count vectorization, tfidf can evaluate how relevant a word is to a document in a collection of documents, which is certainly a better feature representation approach. The scikit-learn library is used to perform the tfidf vectorization on the review text data. Since we have used the spacy library to pre-process the text already, we will skip the tokenizer step here.

There are 2 key parameters we can tune in the tfidf vectorization stage – ngram_range and min_df.

- ngram_range – the default setting is (1,1) which means unigram only. We could use bigram and trigram by adjusting the parameter here. However, using bigram could dramatically increase the number of features in the tfidf matrix, which leads to a higher computational cost. The gains in performance probably won't outweigh the computational cost here. Therefore, we decided to use the default setting unigram only.
- Min_df – this parameter is used to ignore terms that have a document frequency strictly lower than the given min-df value. To remove rare/uncommon words, we set min_df to 100.

3. Model training and evaluation

The LDA model from Sickit-learn is used in model training. We have altered a few parameters to optimize the training process, including

- Setting learning_method to “online”. According to the official documentation, it is better to use “online” for better speed if the data size is large
- Setting n_jobs to 15 to implement multi-core processing to speed up the training process

One key parameter to tune in here is the n_components parameter which specifies the number of themes/topics we want to extract from the model. To find the best value, we started with the value of 2, evaluate the result and then adjust the parameter again until we find the best value. pyLDavis package will be used to evaluate the model performance. pyLDavis provide an intertopic distance map (via multidimensional scaling) to visualize the extracted topics. The rule of thumb is here is to find the output that has a max value of n_components but the topics are further apart from each other on the map.

It is also worth mentioning that we have tried to evaluate the model using log-likelihood and perplexity scores. These metrics were perceived as the most popular metrics is measuring the performance of the model where the log-likelihood higher the better and perplexity is lower the better. We found that the model with the lowest n_component always performs the best on these matrices. As a result, we decided to rely on the intertopic distance map and human judgment for evaluation.

Since we have a large dataset, training the model using the full data set will need approx. ~30 mins for a single run. Therefore, we decided to use 10% of the data to find the best n_components value first.

As a result, we landed at 8 topics for both Singapore reviews dataset and Australia reviews dataset. The pyLDavis intertopic distance map and the top 10 keywords per topic are shown below:

Topic modeling of Singapore reviews with n_components = 8



Topic 1	'order', 'food', 'time', 'helpful', 'service', 'customer', 'delivery', 'deliver', 'app', 'driver'
Topic 2	'app', 'update', 'use', 'order', 'location', 'payment', 'work', 'nt', 'try', 'grab'
Topic 3	'delivery', 'reliable', 'discount', 'restaurant', 'app', 'price', 'offer', 'expensive', 'food', 'need'
Topic 4	'service', 'love', 'fast', '👍', 'delivery', 'thank', 'far', 'job', 'food', 'perfect'
Topic 5	'app', 'easy', 'use', 'convenient', 'useful', 'like', 'work', 'quick', 'super', 'cool'
Topic 6	'grab', 'driver', 'slow', 'n', 'di', 'sangat', 'saya', 'dan', 'bisa', 'terbaik'
Topic 7	'awesome', 'bad', 'experience', 'friendly', 'user', 'app', 'service', 'quality', 'hot', 'polite'
Topic 8	'application', 'address', 'deal', 'app', 'install', 'load', 'log', 'smooth', 'problem', 'open'

Topic modeling of Australia reviews with n_components = 8



Topic 1	'order', 'food', 'time', 'deliver', 'delivery', 'cancel', 'driver', 'service', 'app', 'customer'
Topic 2	'order', 'delivery', 'app', 'offer', 'restaurant', 'food', 'option', 'nt', 'discount', 'need'
Topic 3	'app', 'easy', 'use', 'work', 'super', 'number', 'login', 'payment', 'useful', 'simple'
Topic 4	'far', 'friendly', 'like', 'user', 'perfect', 'happy', 'app', 'zomato', 'area', 'available'
Topic 5	'awesome', 'quick', 'convenient', '👍', 'thank', 'uber', 'easy', 'eat', 'fantastic', 'efficient'
Topic 6	'service', 'fast', 'food', 'delivery', 'app', 'reliable', 'superb', 'cool', 'quality', 'poor'
Topic 7	'bad', 'experience', 'customer', 'service', 'app', 'nt', 'support', 'issue', 'terrible', 'n'
Topic 8	'love', 'app', 'brilliant', 'helpful', 'pretty', 'wonderful', 'swiggy', 'gud', '👍', 'accurate'

4. Topic Evaluation

In addition to looking at the top 10 key terms in each topic, we also sampled a few reviews that highly relevant to each topic, to better understand the context and theme behind the topics. As a result, we assigned the following themes to the topics extracted from Singapore and Australia models.

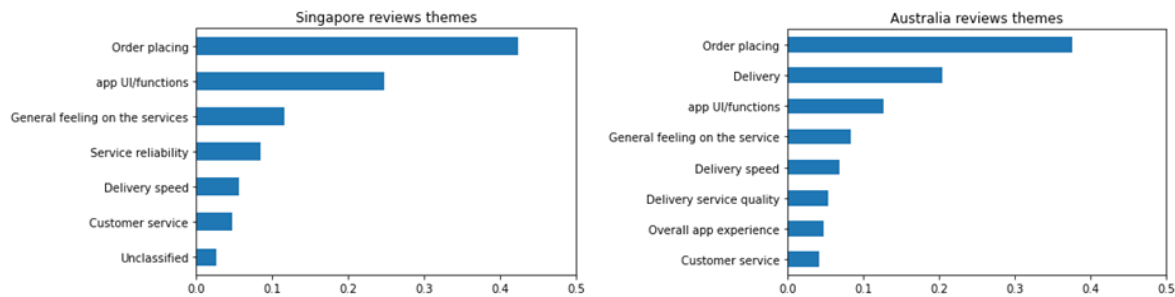
Singapore reviews		Australia reviews	
Topic 1	Order placing	Topic 1	Order placing
Topic 2	app UI/functions	Topic 2	Delivery
Topic 3	Service reliability	Topic 3	app UI/functions
Topic 4	General feeling on the services	Topic 4	Delivery service quality
Topic 5	app UI/functions	Topic 5	General feeling on the services
Topic 6	Delivery speed	Topic 6	Delivery speed
Topic 7	Customer service	Topic 7	Customer service
Topic 8	Unclassified	Topic 8	Overall app experience

Key Takeaway

The model will then be used to transform the review text data using the `lda_model.transform()` function. It will output an array with the size of {no. of review} x {no. of topic} where each review will be assigned a set of values that indicates how relevant it is to each topic. For example, the review below was assigned a set of values, and the highest value is 0.6859 which is in the topic_0 column, this means that the below review is most relevant to topic_0, and topic_0 is the dominant topic of this review

topic_0	topic_1	topic_2	topic_3	topic_4	topic_5	topic_6	topic_7	review
0.685900	0.084880	0.017569	0.017579	0.017769	0.017574	0.141129	0.017599	I've been a DoorDash user for a while now and ...

Below chart were created based on the dominant topic assignment approach as described above.



- In general, the food delivery app reviews from Singapore and Australia shared some commonalities where order placing is the most concerned topic in both reviews
- Interestingly, Singapore reviewers tend to have more interest/concern on the app overall UI/functions, whereas Australian users tend to be more interested/concerned on the delivery aspects in general.

Failure Analysis

During the topic modeling task, there are a few issues that we need to be aware of:

- Some topics are highly related to each other but are still identified as 2 separate topics in the model. For example, topic 2 and topic 5 in the Singapore review model. They are describing how good the app is but using a slightly different expression. Another example would be topic 5 and 6 in the Australia dataset. By looking at the keywords, these topics are highly relevant to each other. topic 5 has a keyword 'quick' and topic 6 has a keyword 'fast'. These 2 words are the same in the food delivery domain, but LDA model could not merge these words.
- We noticed that there is a considerable amount of non-English reviews in the dataset that we were not able to filter out because of no packages that can filter it out with a high accuracy level. Also, many non-English reviews contain English text which makes it difficult to identify it.
- We were not able to assign a theme to topic 8 in the Singapore reviews model, as the top 10 key terms there seem irrelevant to each other at all.

Discussions & Next Steps

- For supervised learning, feature engineering is important to achieve high performance. Our project shows that with a robust feature engineering process, we can achieve relatively high baseline performance.
 - Having good ground truth labels can be challenging, but is fundamental to a good classification task. As we can see in our failure analysis, many misclassifications happened because the customer ratings themselves are not good ground truth labels.
 - There exists a tradeoff between training time and model performance. Deep learning approaches have proven to result in better performance with less data thanks to the power of transfer learning but come at the cost of longer training time and more computational resources (GPU). This is an important consideration that should be evaluated for every NLP pipeline.
- The most challenging part of unsupervised learning is that there is no subjective measure to tell us how good the model is. It heavily relies on human judgment. The failure of using log-likelihood and perplexity to measure the model performance was a surprise for us because these metrics are perceived as the most popular evaluation metrics.
 - The topics extracted from the model look reasonable to us but there is still room for improvement. If time/resources are allowed, we would like to explore the gensim package which is a more robust NLP package than scikit-learn. In addition, the LDA2vec model may be able to help us to tackle the issue around reviewers using different words to represent the same meanings. The LDA2vec model leverages the power of word2vec as the feature representation approach.
- In terms of potential ethical issues, one possible issue that might arise is if we choose to adopt language detection as part of our feature processing. By filtering non-English reviews, we could be training models that are more biased towards English speakers. This is especially apparent for food delivery platforms that operate in countries where there are non-English speakers (e.g Malaysia).
 - Another ethical consideration for deploying these models would be the process of collecting and storing reviews data. Reviews data on both Google App Store and Apple Store could contain personal information such as a person's name and email address, and if these datasets are not stored and processed properly (e.g drop PII columns or restrict access), there could be a risk of exposing sensitive data.

Team Contribution

This project is a collaborative effort with equal participation from everyone, specifically:

- Mel: Supervised Learning (Naive Bayes & Regression), Deep Learning, Final Report
- Bojia: Supervised Learning (SVM & Decision Trees, RandomForest), Final Report
- Stanley: Unsupervised Learning (LDA & NMF), Final Report