Amazon Food Review Classification

Group Work

Contents

1. Data Exploration and Visualization

Exploratory Data Analysis (EDA) is a crucial part of the data modelling workflow as it is this step that leads to developing an understanding of the content and nature of the available dataset.

The following steps were undertaken as part of EDA w.r.t. this project. (KGP Talkie, 2020)

1. Once data was loaded, a check for missing values were conducted the result of which was negative.

2. The next part involved inspecting the reviews to understand what kind/extent of cleaning would need to be performed. The entirely English text was relatively clean apart from some special characters, URLs and html tags. It contained emoticons which would be replaced by their meaning in words.

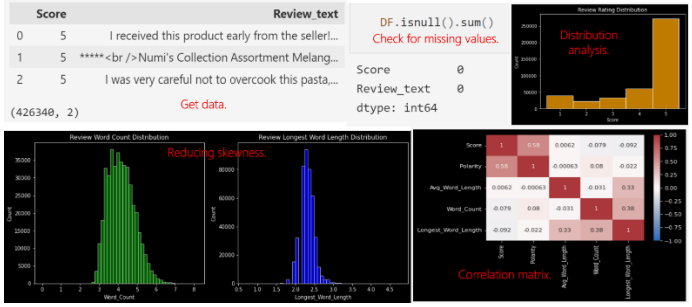
3. Then, cleaning of the text was conducted. Regular expressions returned matches w.r.t. text to be removed which would get replaced by the empty string. In this way, html tags, URLs and anything that's not an English alphabet were removed. Emoticons found in the text were replaced by their textual description obtained from the "emot" library. Contractions in the text (like "she'll") was expanded (like to "she will") using a dictionary of contraction expansions gathered from Marteen (2013). Then, words that were joined by hyphens and ones containing "*'s*" indicating belonging, were replaced with space and empty string. All text was made lowercase.

4. Often with text data, conducting processes like named entity recognition (NER)/length analysis can reveal new features in the data to be given as supplementary input to ML models. (M.D. Pietro, 2020). Here, length analysis was performed and average word length, word count and longest word length was determined. Moreover, polarity of the reviews as output by "vader" library was obtained as another feature. Correlation between review ratings and new features were inspected and it was found that while polarity had the highest correlation with rating, average word length had the lowest and hence, it was dropped.

5. Distribution of the data was then analyzed using histograms and it was found that data was highly unevenly distributed with rating 5 being much higher than others. Since ML models benefit from balanced data, it was decided to perform resampling of data later on.

6. The added features "polarity", "word count" and "longest word length" were found to have highly skewed distributions. So, a log transformation was applied to "word count" and "longest word length" features to reduce the skewness which also lead to a slight increase in correlation with review ratings. Polarity was left untouched as transforming it would result in undesirable reduction in its correlation with ratings.

7. Finally, it was identified that "word count" and "longest word length" were to be standardized later on, to conclude this part of the notebook.



2. Text Processing and Normalization

Text Processing

Since the data has undergone cleaning in the previous section, the only steps left to prepare the data for the normalization process was removing the stop words. Stop words do not extra value to the sentiment analysis and their removal reduces the number of features present to highlight only the important information in the reviews. A set of stop words have from NLTK's library is utilized to complete this step (Deepanshi, 2021).

Stemming or Lemmatizing?

Stemming normalizes text by trimming a word to its stem following a series of sequential steps based on fixed rules. The stems produces might not always be valid words; i.e. the stems can be meaningless. Lemmatization uses a dictionary to retrieve the root form of words. This ensure that the lemmas are semantically valid words. Although there are many algorithms that perform stemming, the Lancaster stemmer overstems the words so it was ruled out of use. The Porter stemmer and the Snowball stemmer in NTLK performed better than the Lancaster stemmer (Jabeen, 2018). However, as the common tradeoff with stemming is that the text of the reviews would after stemming be lexicographically invalid, lemmatization was chosen. Since our application deals with sentiment analysis, it was of great importance that words post-normalization are valid words with their meanings intact as the aim of this project is to be able to assign a label/score to a review based on the sentiment of the review (Srinidhi, 2020).

Why Use spaCy instead of NLTK's Lemmatizer?

One of the reasons to shift to spaCy's implementation of the lemmatizer was the flexibility of adding rules of how words are to be lemmatized. With WordNet's lemmatizer, words like "us" would be lemmatized to "u". Hence, maintaining the word as "us" post-lemmatization was necessary as "u" is an inaccurate normalization of "us". In addition, WordNet's lemmatizer did not lemmatize some nouns correctly. For example, the word "Vegas" would be recognized as a place, only when beginning with a capital letter. Since the data was made lowercase, "Vegas" became "vegas" causing the lemmatizer to treat it like any other noun thereby removing the "s" in "vegas" to return "vega". SpaCy's implementation contains a pipeline that tags part-of-speech (POS) of every word internally. This stage is especially important to lemmatization as the meaning of words changes according to the grammatical context, they are placed in.

SpaCy's implementation contains a pipeline that tags part-of-speech (POS) of every word internally. This stage is very important to lemmatization as the meaning of words change according to the grammatical context they are placed in.

One of the reasons to shift to spaCy's implementation of the lemmatizer was the flexibility of adding rules of how words are to be lemmatized. With WordNet's lemmatizer, words like "us" would be lemmatized to "u". Hence, maintaining the word as "us" post-lemmatization was necessary as "u" is an inaccurate normalization of "us". In addition, WordNet's lemmatizer did not lemmatize some nouns correctly. For example, the word "Vegas" would be recognized as a place, only when beginning with a capital letter. Since the data was made

lowercase, "Vegas" became "vegas" causing the lemmatizer to treat it like any other noun thereby removing the "s" in "vegas" to return "vega".
SpaCy's implementation contains a pipeline that tags part-of-speech (POS) of every word internally. This stage is very important to lemmatization as the meaning of words change according to the grammatical context they are placed in. For Example,

- "The object she held onto was very precious to her." The object in this sentence is a noun.
- "I object to ruining the surprise." Whereas, the object here is a verb.

Without functionality of picking up on POS, lemmatization will fail to process all the words correctly thereby failing to reduce feature space to the extent that is possible. The WordNet Lemmatizer does not come with POS tagging functionality; it has to built by users.

Lemmatization: Original Text Vs Cleaned Text

The lemmatization process was experimented with on both original reviews and cleaned reviews. This helped understand factors that affect lemmatization, since POS tagging largely depends on full sentences to tag each word correctly and the cleaning process removes stop words which may affect the grammatical structure of a sentence. The hypothesis was that cleaned text will be lemmatized differently. The results proved that the opposite has happened. The reviews that have been cleaned were lemmatized identically to the reviews that were not cleaned then got lemmatized. Moreover, cleaning the text meant any contractions present within, were appropriately expanded and lemmatized correctly. Thus, it was concluded that the better approach is to complete text processing prior to lemmatization.

Training Models

The best model for accuracy was the "trf" model as it lemmatized adverbs correctly. However, this model was very expensive, it lemmatized 100 reviews per minute, meaning that lemmatizing the entire dataset would take an estimate of 83 hours. Thus, to change the model to a speedier one, spaCy documentation recommends "sm" (spaCy, 2022b). There were a number of steps within the pipeline that were unnecessary, such as “ner” (Named Entity Recognizer) which recognizes and classifies named entities. The parser in the sm model was disabled as well and replaced with “sentencizer” after checking that the model with the parser enabled performed the same results when the parser was replaced with the sentencizer (TR517, 2018). They both deal with sentence segmentation using punctuation, but as the punctuation is removed during the cleaning process, it was not necessary to use them (spaCy, 2022a).

3. Vector Space Model and Feature Representation

Two different models have been used that stem from the bag of words model ie, the TFIDF (Term Frequency inverse document frequency) and the term frequency model. Due to the nature of the problem space, we have opted to use bigrams instead of unigrams. This is because a lot of the text is prefixed with certain negating words which express a positive sentiment with one word but a negative with another which might cause the final model to miscalculate if taken separately. That being said, not is usually considered as a stop word hence the restrictions in the stop words. (Silge, 2017) The features generated initially (word count, longest word length, and Polarity) were appended to the feature list generated for the TF and TFIDF models and the glove embedding generated per review, and this final data frame was used as input in the classifiers.

Resampling: This dataset was heavily imbalanced. To counter this issue resampling was taken advantage of. Just undersampling the data to the class that contained the least number of samples ended up taking out a lot of useful data and reduced the number of training samples to around 60,000. Only using oversampling to reach the majority class was a little too heavy on the RAM and increased the number of training samples to around 700,000. The best option was to combine both. The threshold was set to 30,000 and all the classes were forced to reach that threshold whether through oversampling or undersampling. This amounted to a training set of 150,000 samples. Although not ideal (loss of a lot of the samples in the majority class), due to the major difference in the number of samples per class, this seemed to be the most sensible way to get around the feat.

Near-miss and SMOTE or Synthetic Minority Oversampling Technique are efficient resampling algorithms for imbalanced datasets. Near-miss is used primarily for undersampling and the elimination is done randomly according to the classes provided by the sampling strategy (Bhoomika Madhukar,2020). SMOTE is an oversampling algorithm. Other oversampling techniques involve randomly duplicating the minority classes while SMOTE on the other hand selects neighbors in a feature space and synthesizes a new sample in between these neighbors. Therefore, SMOTE works with embeddings and not the sentences themselves. (Jason Brownlee, 2020) Initially, the Bag of Words (BOW) representation technique was used to extract features for our text classification problem and the vocabulary was represented using bigrams. However, two different methods were used for scoring the documents and these have been briefly explained as follows.

The Term Frequency (TF) strategy was first used for assigning numeric values per bigram for each review present in the given dataset. TF is defined as the ratio of the number of times a word appears in a review (the given document) to the total number of words in the review. (Panchal, 2019)

$$TF(w) = \frac{Number\ of\ times\ term\ w\ appears\ in\ a\ document}{Total\ number\ of\ terms\ in\ the\ document}$$

The CountVectorizer model (offered by sklearn) was used for transforming a document (review) into a vector based on the frequency of each term (bigram) that occurs in the document. Each individual frequency count acts as a feature (Tracyrenee, 2021). A matrix is generated in which the columns consist of each unique term present in the corpus and each of the rows contain each document (food reviews) present in the corpus. The frequency of the term in a certain document serves as an element in the matrix. (GeeksforGeeks, 2020)

Term Frequency Inverse Document Frequency (TFIDF) is a measure that speaks for the relevancy of a word concerning a collection of required documents. This measure is sound by multiplying the frequency of the n-gram over one document with how rare that word is across all the documents. This ensures that words that are very much common across the documents are not given a high score and this measure is useful in NLP tasks for scoring words. Google Colab was the primary IDE used for developing the feature representation models, and due to the limited availability of RAM on Colab, each of the Bag of Words (BOW) representations were limited to only the top 1000 features (ordered by term frequency) of the given text corpus. (MonkeyLearn Blog, 2019)

Global Vectors (GloVe) is a popular technique in NLP that is usually used for generating word embeddings. In GloVe, a large co-occurrence matrix is constructed in which the rows are represented by the count of each term and the occurrence frequency of this term in some context (Great Learning Team, 2020) serves as the column of the co-occurrence matrix. Additionally, the values of the co-occurrence matrix are normalized, and the log-bilinear function is applied on them. (Sciforce, 2018) The reviews are converted to vectors by using the following strategy. As the GloVe model produces 1 vector per word in the document, the sum of vectors of all the words in a document was computed to obtain a vector that represents each document.

4. Model Training, Selection and Hyperparameter Tuning and Evaluation

Support vector machines: Support vector machines or SVMs, are machine learning models with learning algorithms linked to them that examine data for regression or classification (Wikipedia Contributors, 2022). The major objective of this kind of learning method is to discover a hyperplane in a space that provides the maximum distance between both classes and that can classify data points. Support vector machines are

generally highly preferable due to their ability to provide significantly high accuracy with significantly less computation time. (Gandhi, 2018)
That being said for SVMs, the training time is directly proportional to the size of the dataset, and this was observed during testing. A common optimization technique in machine learning due to its speedy implementation is gradient descent. SGD or Stochastic Gradient Descent with a loss "hinge" is the optimized equivalent of the linear SVM. The way that this works is by iteratively updating a set of coefficients that were initialized with random values. was used to optimize this (Arliss, 2020). The best performance in this classifier was provided by the TFIDF model. (scikit-learn, 2022) Almost a 20% increase in the TFIDF model, as well as the glove model, was observed in the accuracy after the involvement of SGD. There was also a significant reduction in the training time in all the models.

Logistic Regression is a supervised learning classification algorithm that uses the sigmoid function to produce a probability prediction between 0 and 1. They are intended to be used for binary classification problems, although the scikit-learn implementation allows for multi-class classification. The hyperparameter "multi_class", is set by default to auto so that it detects the type of data, if it is a binary problem is fit to the model "ovr" is selected to allow for OneVsRest scheme to be applied. If it is a multi-class problem is fit to the model, then the mode changes to "multinomial" to appropriately classify the data (Hale, 2019).

Initially, Logistic Regression models did not perform well and so many hyper parameters were tested manually instead of running Grid Search as that was predicted to be time-consuming due to the large size of the dataset. Thus, various solvers were used to find the parameter weights that minimize the cost function (Hale, 2019). Penalties for each model were adjusted accordingly. Nevertheless, all the results produced were very identical to the default model. Therefore, the default model was used as it was the fastest one to run. The GloVe model performed the best for the Logistic Regression. This could be attributed to it having the most linearly separable data in comparison with the Term Frequency and TF-IDF models.

Naive Bayes is a probabilistic classifier based on the Bayes theorem and it is used for classification. It assumes that the features are independent of each other, i.e., one feature will have no effect on another feature and that they are identically distributed (Gandhi, 2018). Many types of the Naïve Bayes classifier were tested on but specifically the Multinomial classifier provided these best results, as it is suitable in classifying documents to a class label. In this case, it would classify a review to a rating between 1-5. It works with discrete features such as frequency of words in text classification, which is the Bag of Word approach discussed previously. This explains why, unsurprisingly, it performed the best for Term Frequencies model (Great Learning Team, 2020) (scikit-learn, 2022b).

The reason to run cross-validation is to assess the performance of a machine learning model on unseen data, in this case the testing data. K Folds technique was used as it ensures that every sample from the original data will appear in both the training and testing set.

The data will be split into K groups, and K-1 folds will be used to fit the model. In this case, we are splitting the data into 10 groups, where 9 groups will be used to fit/train the model and the validation of the model will happen with the remaining 1 group that will be used to test the model. (Lyashenko and Jha, 2022)

Using 10-fold cross validation, the accuracy across different types of data, i.e., data from TF, TF-IDF and GloVe embeddings, that are much higher than what the regular train/test has presented; it improved by ~23-26%. This increase can be explained by the data set being shuffled in the beginning and by having every sample (i.e., sample ranging across reviews of all ratings) appear so that the model is training fairly. (Lyashenko and Jha, 2022)

Grid Search: Grid Search is a technique used to find the best hyper-parameters for a model (parameters that aren't learned in the estimators) for the best cross-validation score. In this scenario, grid Search was run on the SGD-SVM classifiers for the TFIDF model as well as the glove model (this classifier gave the best results for these 2 models). (scikit-learn, 2022a) However, applying grid search did not help in increasing accuracy.

Concerning the models, the glove model and the TFIDF model did the best with the SGD model (optimized SVM) and the term frequency model did the best with the Multinomial Naive Bayes.

5. Modelling Text as a Sequence

NLP tasks are sequential modelling problems because meaning of sentences in human language is dependent on sequence (order) of words. Many ML models (ANN, CNN, etc) struggle to capture sequential nature of problems due to their lack of the ability to deal with time series data. This is where the Recurrent Neural Network (RNN) is different. (IBM Cloud Education, 2020), (OpenGenus, n.d.)

- RNNs have an internal state and can keep track of the state of the Neural Network (NN) at different time steps such that state at one time step may influence state/output in the next time step. This memory/ability to store network state across points in time, is why RNNs outperform other ML models when it comes to solving sequential modelling problems such as NLP tasks (next word prediction, names entity recognition, sentiment analysis, machine translation), music generation, speech recognition, etc.
- RNNs can be of different kinds depending on no. of inputs/outputs (one to one RNN, one to many RNN, many to one RNN, many to many RNN).

NLP problems are generally complex and may require an RNN with many hidden. But, because RNNs use output from a previous time step as input of the next time step along with maintaining hidden state, they follow slightly different backpropagation than feedforward NNs in that, errors at each time step gets summed over as time progresses. This makes RNNs vulnerable to exploding gradients (gradients become so large that the model becomes unstable) or vanishing gradients (gradients become too small and thus don't contribute to learning) problem. While these issues can be resolved by simply lowering the no. of hidden layers, this may not be ideal. Instead, advanced variations of RNNs (Bidirectional RNN (BRNN), Long Short Term Memory (LSTM) RNN, Gated Recurrent Unit (GRU)) exist which produce better results with fewer layers/faster.

BRNN: This kind of RNN uses data from prior as well as future time steps to influence outputs at any given time step.
LSTM: Cells in hidden layers of such networks have 3 gates (input gate, output gate, forget gate) that regulate flow of information such that relevant information be it from further back in time is retained while less important information may be forgotten. Such networks capture context well and thus are great for NLP tasks. In addition to solving the vanishing gradients problem, LSTMs also effectively solve the Short Term memory problem (information from earlier timesteps which may contain important context related knowledge has lesser/no influence in later time steps and gets replaced with recent state information even if it may be less important).
GRU: GRUs also solve the memory problem using hidden states and 2 gates (reset gate, update gate) instead of a cell state and 3 gates that is the case with LSTM.

Because sentiment analysis and text classification (being NLP problems) are often sequential/time-series problems, they are solved using Recurrent Neural Networks (RNNs) with much success. The LSTM variant of RNN is often preferred for text classification and sentiment because of its innate ability to better capture context of words in a document. (IBM Cloud Education, 2020)

Since in this CW, food reviews are to be classified into classes representing ratings which are reflective of someone's sentiment towards the food, it was decided that the LSTM RNN variation would be a good choice of classifier. The "many-to-one" kind of RNN was used since the aim is for the model to learn to output a single rating upon receiving multiple words in a review as input. An embedding layer was incorporated into the model that suitably embeds each input document into a vector of sub-vectors that correspond to each word in the review. The embedding layer provided by Keras can be trained to learn embedding based on given data. Another way to add the embedding layer is to use it with a provided embedding matrix with

words in the vocabulary mapped to vectors produced by a pre-trained model. Both approaches were explored.
The models were trained on 60% of the data, validated on 20% and tested on another 20% of the data. This split was possible due to the large size of the dataset.

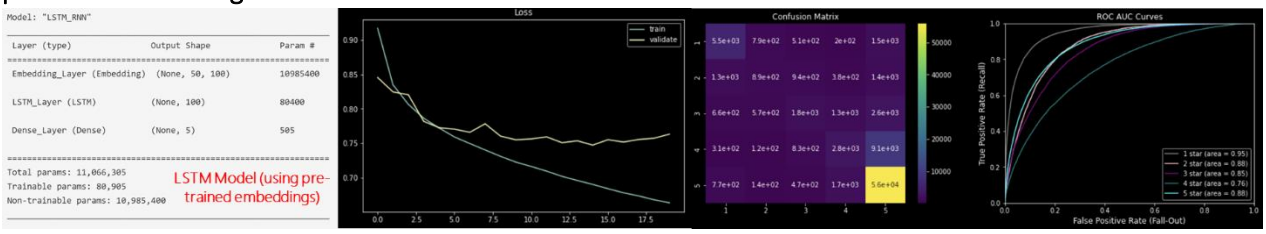
The following observations were made after training, validation, and testing.
(Using Pre-Trained GloVe Embeddings)

- The model achieved a fair accuracy of ~72%. Though there was some overfitting, it was not very high due to the 10% dropout incorporated into the model.
- From ROC AUC values, it was observed that the model achieved a similar degree of correctness in prediction across samples with different ratings. Judging from ROC curves alone, the model seems to have performed best with rating 1 samples since it correctly identified most (95%) of 1 rated sample with fewer misclassifications of samples with other ratings as having rating 1. Reasons like samples with rating 1 being much fewer in number than others combined with consequently less likely variation among such reviews may have contributed to the model exhibiting the observed result w.r.t reviews with rating 1.
- Considering Precision and Recall values whose trade-off is measured using the F score, it was noted that the model performed best when classifying samples with rating 5 (F = 0.86) and second best when classifying samples with rating 1 (F = 0.64), with ratings in between, seemingly confused the model a bit. This is understandable, because overwhelmingly positive and negative reviews are indeed easier to discern compared to reviews with ratings in between.

(With Custom Training of Embedding Layer)

- This model did better with training data achieving accuracy of ~82% but performs poorly with testing data with only ~70% accuracy. (lower bias, higher variance).
- This model without added dropout layers showed a significant amount of over-fitting. This was expected since training the embedding layer would make it adapt very well to the training set but not necessarily to new words the testing set as it may not be capturing meaning of the words like the GloVe embeddings do, in the limited number of epochs the simple model was trained for.

Overall, the LSTM model (using pre-trained GloVe embeddings), staying true to its reputation, showed best performance among all classifies so far w.r.t this NLP task.



Since RNNs tend to leave essential text related information out, popular variants of RNNs – LSTMs and GRUs are often preferred for text classification problems over regular RNNs. As an LSTM model was implemented for our text classification problem, it was decided to experiment with an additional variant of RNNs, i.e., the

GRU model.

The model generated the best results for the pre-defined GloVe word embeddings when compared to the locally trained embeddings, as observed in the two experiments involving the LSTM classifier. As a result, the GRU model's embedding layer produced its embeddings based on the pre-trained GloVe model.

The following observations were made for the GRU model:

- From our experiment conducted using the GRU model, it was observed that the model achieved an **accuracy of ~72%**. In the "Loss" graph and the difference between the computed train and test scores, it was evident that the model was subjected to some overfitting. However, the overfitting is limited due to the use of the dropout strategy while building the GRU.
- In the ROC AUC Curves plot, it is evident that the GRU model was able to distinguish between the classes – [1, 2, 3, 4, 5]. Additionally, from the same plot, it can be inferred that the **AUC for the "1 star rated**

reviews" ROC Curve (represented in gray color) is the highest amongst the remaining four classes. In the experiment conducted using the LSTM, a similar observation was made.

- In terms of the F1 score, the model calculated the best F1 score for samples with a rating of 5 **(F1 score = 0.86)**, whereas samples having a rating of 2, an **F1 score of 0.19** was generated. The F1 score values were as such due to the imbalanced dataset used as input to the GRU model. The samples having a rating of 2 had the lowest F1 score in comparison to the other four classes, since they served as the minority class in the imbalanced dataset. Due to the reason that most of the reviews had a rating of 5, the "5 star rated" samples had the highest F1 score.

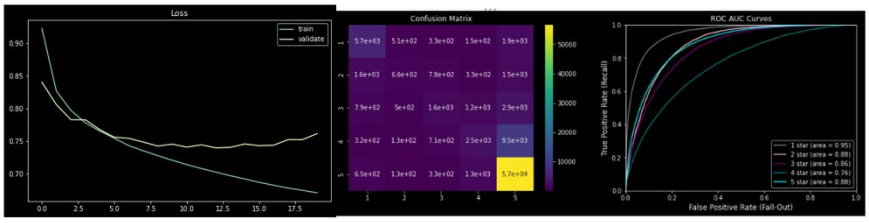
```
Model: "GRU_RNN"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Embedding_Layer (Embedding)  (None, 50, 100)          10985400

 GRU_Layer (GRU)             (None, 100)               60600

 Dense_Layer (Dense)         (None, 5)                 505

=================================================================
Total params: 11,046,505
Trainable params: 61,105
Non-trainable params: 10,985,400
_________________________________________________________________
```

GRU Model



LSTM vs. GRU:

After comparing the two RNN variants, the following observations were inferred:

- From the results generated, it can be concluded that the **LSTM performed slightly better** than the GRU model. As opposed to GRUs, LSTMs generally deal with larger sequences due to their ability of capturing the context of the data (V. Padia, 2020). Therefore, the LSTM gave a slightly higher accuracy than GRU.
- The **LSTM model took ~15 minutes** to complete the execution, while the **GRU model completed its execution in ~11 minutes**. This has been observed due to the following reason – In comparison to the LSTM model, the GRU has fewer gates and lesser tensor's operation (A. Jaiswal, 2022) which makes GRUs less complex and faster than LSTMs.

6. Topic Modelling of High and Low Ratings

Topic modelling refers to employing algorithms to extract possibly hidden themes/topics from a set of documents. Topics so identified hasten the process of understanding general content of documents for humans. Moreover, topics and associated words thus identified may reveal patterns/trends that can otherwise go unnoticed. Since for most NLP tasks, each word is considered a feature, there is often a large no. of features which make ML tasks complicated and resource consuming. Thus, feature/dimensionality reduction is necessary when modelling text for ML. Here too, topic modelling may be of aid, since it can reveal important/common themes based on which features may be filtered. (Manthiramoorthi, n.d.). Various kinds of topic modelling algorithms exist that follow both statistical as well as linear algebra-based language modelling approaches. Algorithms like LDA and LSA, that follow a probabilistic/statistical approach, use statistical tools to perform topic modelling and may do so based on probability distribution identified over word sequences (AI - All in One, 2016). Algorithms like NMMF that follow linear algebra-based approach, involve representing documents as matrices and factorizing them into matrices that capture relations between the document, its terms and topics. (A. Klos, 2020)

Topic analysis of 5 rated and 1 rated reviews were conducted to determine what food products are popular/loved as well as ones which people dislike. Topic modelling was done using a most popular

	5 Rated Reviews	1 Rated Reviews
Broad Topic 1	Tea, Coffee	Eating, Diet, Hunger, Food, Water
Broad Topic 2	Baking/Baked Products, Healthy, Snacks, Special Dietary Needs (Gluten Free, Whole Wheat, etc)	Inorganic Additives, Artificial Sweetness
Broad Topic 3	Pet Food	Stale Food, Meat

	5 Rated Reviews	1 Rated Reviews
Topic 1	Tea	Eating, Feeding, Hunger, Water
Topic 2	Baked/Baking, Goods, Gluten Free	Meat, Chicken, Leftover, Stale
Topic 3	Pet Food	Mixed, Bad Taste
Topic 4	Soothing, Anti-Allergy, Anti-Acidity	Bad Flavor
Topic 5	Butter, Nuts, Allergy	Digestive Issues, Fruit
Topic 6	Sleep, Relaxation	Coffee
Topic 7	Vegan, Organic	Hexane Additive, Toxic, Pet Food
Topic 8	Female Reproductive Health	Carcinogens
Topic 9	Mixed	Drug Traces
Topic 10	Cheap, Less Nutritional Value, Less Health Value	Stale, Faulty Packaging, Bad Service
Topic 11	Healthy, Snacks	Cost
Topic 12	Pets, Health	Poultry, Ailments
Topic 13	Coconut, Oils, Healthy	Mixed, Pork, Gelatin
Topic 14	Health, Medecine, Desserts	Mixed, Foul Flavor
Topic 15	Mixed, Kids	Foul Flavor
Topic 16	Drinks, Coffee, Herbal Tea	Artificial Sweetness, Bad Taste
Topic 17	Exercise, Diet, Body Weight	Spicy
Topic 18	Pet , Health	Pet Food
Topic 19	Mixed	Pet Food
Topic 20	Pregnancy	Valerian Tea

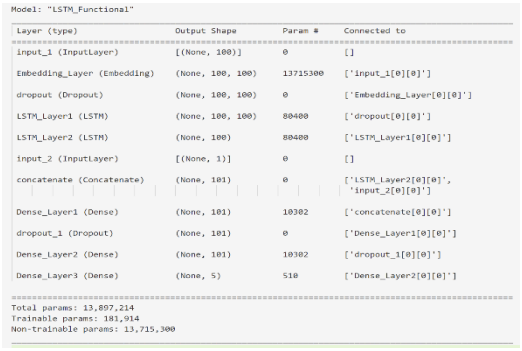
algorithm, Latent Dirichlet Allocation (LDA) to try and identify 20 topics w.r.t. positive/negative ratings. Topics thus identified were visualized using pyLDAvis. From the visualization, there seemed to be 3 broad topics within the 20 topics. These were also investigated. The 20 Topics as well as the 3 broad topics identified can be summarised as follows.

Overall conclusions drawn from topics identified after LDA using pyLDAvis was as follows.

- Food products that customers love to purchase online are tea, coffee, baked/baking goods, pet food (mostly dry food) and snacks.
- It can be observed that perishable foods such as meat (also wet pet food)/veggies were what was reported by some customers as stale/having foul odor. This reveals that online retailers may want to consider improving their food transport/storage facilities to keep perishable foods fresher longer. Adding additives to food to keep them fresher longer may not be the best strategy since customers today are very aware of food additives as evident from potentially harmful additives being the cause of a lot of foods being rated 1.
- It is to be noted that online buyers are largely health aware and highly appreciate healthy foods (organic, nutrient dense, vegan) and are fairly vocal when it comes to providing negative reviews when products lack nutrition or contain harmful ingredients even in the case of pet food. It is in best interest of online retailers to sell nutritious food items with natural quality flavors.
- Pet food seems to be amongst the most popular online food grade commodity thus, retailers may want to maintain more stock of such products. However, they must be maintained without them spoiling as pet owners are critical about feeding their fur babies quality food. This is especially critical since pets (cats, dogs) are known to have sensitive stomachs as confirmed from the many gastrointestinal issues reported as a consequence of stale or contaminated pet food.
- An interesting observation was that foods/ingredients that soothe gastrointestinal ailments, lessen sleeplessness/anxiety or helps manage difficulties w.r.t. female reproductive system is highly in demand despite there being few complaints regarding the taste of such products such as “valerian tea” being foul. Online retailers may thus be interested in expanding their inventory w.r.t. such food products since it is understandable that buyers with such ailments would prefer to order online as compared to physically purchase items under limitations that ailments may impose.
- New mothers (likely due to their busy schedule) form a significantly large section of online food shoppers (food for baby/toddler) and highly appreciate good nutritious products for kids.
- An interesting observation was that “sugar rich” and “junk” food appeared amongst both 5 rated as well as 1 rated review. This is likely because the taste of processed foods with high sugar, fat and salt while not healthy, is still highly palatable. Thus, the task of striking a balance between “junk” and “nutritious” food surfaces as a challenge that retailers must take on. A good idea may be to make tastier nutritious food options available.
- Some of the negative reviews also seem to be about customer service (like packaging), which could be a point of improvement that the online store could investigate.
- Finally, w.r.t. price of food, the views are conflicting. While some positive reviews hint at good value for money, it cannot be ignored that many negative reviews point to overpriced low-quality purchases as the reason for assigning a low rating. It is suggestive that retailers may be trying to provide fair prices as they should. However, they may also want to bear in mind that from analysis of positive vs negative reviews, it can be inferred that value for money is just as important as availability of cheaper products. This can be confirmed from the fact that while generally, a cheap product is highly appreciated (from positive reviews), the appreciation is received if and only if the food product meets minimum food grade quality in terms of safely, nutrition and taste as any food product (even cheap ones) that do not meet minimum standards are deemed a wastful expenditure/mistake no matter how low its price was (from negative reviews).

7. Kaggle Submission

Processing/experiments associated with final model. The submitted model is an LSTM variation of the RNN model. It has an embeding layer that generates embeddings based on outputs by a pretrained GloVe model. The LSTM model accepts a document represented as 100 dimension vector of word positions. This vector passes through the embedding layer which outputs a 100 dimension vector after embedding such that meaning of the words are captured. Then this vector passes through a dropout layer and then through 2 LSTM layers with 100 units each. The output of the LSTM layer is concatenated with the polarity feature value to produce a 101 dimension vector which is passed to 2 dense layers with 101 units and "relu" activation such that there is a dropout layer between the 2 dense layers. The output now passed through the final dense layer with 5 (no. of classes) units and "softmax" activation. The model is implemented using keras functional API and is compiled with categorical cross entropy as loss function and adam as optimizer. The model was trained for 40 epochs. The architecture of the model is as given below.



Model: "LSTM_Functional"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 100)]	0	[]
Embedding_Layer (Embedding)	(None, 100, 100)	13715300	['input_1[0][0]']
dropout (Dropout)	(None, 100, 100)	0	['Embedding_Layer[0][0]']
LSTM_Layer1 (LSTM)	(None, 100, 100)	80400	['dropout[0][0]']
LSTM_Layer2 (LSTM)	(None, 100)	80400	['LSTM_Layer1[0][0]']
input_2 (InputLayer)	[(None, 1)]	0	[]
concatenate (Concatenate)	(None, 101)	0	['LSTM_Layer2[0][0]', 'input_2[0][0]']
Dense_Layer1 (Dense)	(None, 101)	10302	['concatenate[0][0]']
dropout_1 (Dropout)	(None, 101)	0	['Dense_Layer1[0][0]']
Dense_Layer2 (Dense)	(None, 101)	10302	['dropout_1[0][0]']
Dense_Layer3 (Dense)	(None, 5)	510	['Dense_Layer2[0][0]']

Total params: 13,897,214
Trainable params: 181,914
Non-trainable params: 13,715,300

Note: It was observed while entering model predictions in the Kaggle competition that models when trained on data that's not resampled led to better results sometimes. This is expected to be because even though combining undersampling with oversampling is a widely accepted practice, in this case the large disparity in frequency distribution between reviews rated 5 and others lead to under sampling removing a large amount of 5 rated samples while oversampling through the amplification of lower rated reviews may have amplified noise and distorted the distribution of data in the train set making it too different from the test set. The key learning here was that in cases where data naturally follows uneven distribution (like here, people generally tend to give more positive reviews than negative) such that minority classes are not critical (in case of heart diseases for example), it is more ideal to preserve some unevenness of distribution to produce most accurate and natural predictions.

8. Conclusion

Overall, this coursework has been an exciting opportunity to apply theoretical knowledge on a real-world NLP task. The insights gained from topic modelling seemed like real useful inferences that online retailers would benefit from. Moreover, the experience of exploring roughly every step of the model deployment workflow lifecycle all the while collaborating as a team, though challenging, has been a rewarding experience. The Kaggle competition was a fun motivator!

References

A

- Abhishek Jaiswal. (2022) *"Tutorial on RNN | LSTM |GRU with Implementation",* Analytics Vidhya. Link: https://www.analyticsvidhya.com/blog/2022/01/tutorial-on-rnn-lstm-gru-with-implementation/ [Last Accessed: 27 Mar. 2022]
- Akash Panchal. (2019) *"NLP — Text Summarization using NLTK: TF-IDF Algorithm",* Medium. Link: https://towardsdatascience.com/text-summarization-using-tf-idf-e64a0644ace3 [Last Accessed: 27 Mar. 2022]
- Alex Klos. (2020) "Topic modeling: LDA vs. NMF for newbies". Link: https://alexklos.ca/blog/natural-language-processing-lda-vs-nmf-for-newbies/ [Last Accessed: 24 Mar. 2022]
- Artificial Intelligence - All in One. (9 Apr 2016) "Lecture 17 — Probabilistic Topic Models Overview of Statistical Language Models - Part 1 | UIUC", YouTube Video. Link: https://www.youtube.com/watch?v=znTOwpBHlbE [Last Accessed: 24 Mar. 2022]
- Arliss, W. (2020). *Solving the SVM: Stochastic Gradient Descent and Hinge Loss | Towards Data Science*. [online] Medium. Link: https://towardsdatascience.com/solving-svm-stochastic-gradient-descent-and-hinge-loss-8e8b4dd91f5b [Accessed 27 Mar. 2022].

B

- Bhoomika Madhukar (2020). *Using Near-Miss Algorithm For Imbalanced Datasets*. [online] Analytics India Magazine. Link: https://analyticsindiamag.com/using-near-miss-algorithm-for-imbalanced-datasets/ [Accessed 27 Mar. 2022].

D

- Deepanshi (2021). *Text Preprocessing NLP | Text Preprocessing in NLP with Python codes.* [online] Analytics Vidhya. Link: https://www.analyticsvidhya.com/blog/2021/06/text-preprocessing-in-nlp-with-python-codes/ [Accessed 27 Mar. 2022].

G

- Gandhi, R. (2018). *Naive Bayes Classifier - Towards Data Science.* [online] Medium. Link: https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c [Accessed 27 Mar. 2022].
- Gandhi, R. (2018). *Support Vector Machine — Introduction to Machine Learning Algorithms*. [online] Medium. Link: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 [Accessed 25 Mar. 2022].
- GeeksforGeeks. (2020) *"Using CountVectorizer to Extracting Features from Text",* GeeksforGeeks. Link: https://www.geeksforgeeks.org/using-countvectorizer-to-extracting-features-from-text/ [Last Accessed: 27 Mar. 2022]
- Great Learning Team (2020). *Multinomial Naive Bayes Explained*. [online] GreatLearning Blog: Free Resources what Matters to shape your Career! Link: https://www.mygreatlearning.com/blog/multinomial-naive-bayes-explained/ [Accessed 26 Mar. 2022].
- Great Learning Team. (2020) *"What is Word Embedding | Word2Vec | GloVe"*, GreatLearning Blog. Link: https://www.mygreatlearning.com/blog/word-embedding/ [Last Accessed: 27 Mar. 2022]

H

- Hale, J. (2019). *Don't Sweat the Solver Stuff - Towards Data Science.* [online] Medium. Link: https://towardsdatascience.com/dont-sweat-the-solver-stuff-aea7cddc3451 [Accessed 27 Mar. 2022].

I

- IBM Cloud Education. (14 Sep 2020) *"Recurrent Neural Networks"*, ibm.com. Link: https://www.ibm.com/cloud/learn/recurrent-neural-networks#toc-variant-rn-2xvhb_yi. [Last Accessed: 23 Mar. 2022]

J

- Jabeen, H. (2018). *Stemming and Lemmatization in Python.* [online] DataCamp Community. Link: https://www.datacamp.com/community/tutorials/stemming-lemmatization-python
- Jason Brownlee. (2020) *"SMOTE for Imbalanced Classification with Python*", Machine Learning Mastery. Link: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/ [Last Accessed 27 Mar. 2022]

K

- KGP Talkie. (16 May 2020) *"Complete Exploratory Data Analysis (EDA) on Text Data in Python | Text Data Visualization in Python"*, YouTube Video. Link: https://www.youtube.com/watch?v=HVBk2Ge_Q98. [Last Accessed: 1 Mar. 2022]

L

Lyashenko, V. and Jha, A. (2022*). Cross-Validation in Machine Learning: How to Do It Right - neptune.ai.* [online] Neptune Blog. Link: https://neptune.ai/blog/cross-validation-in-machine-learning-how-to-do-it-right [Accessed 27 Mar. 2022].

M

- Maarten (2013). *Expanding English language contractions in Python.* [online] Stack Overflow. Available at: https://stackoverflow.com/questions/19790188/expanding-english-language-contractions-in-python [Accessed 28 Mar. 2022].
- Mauro Di Pietro. (2020) *"Text Analysis & Feature Engineering with NLP"*, towardsdatascience.com. Link: https://towardsdatascience.com/text-analysis-feature-engineering-with-nlp-502d6ea9225d [Last Accessed: 27 Mar. 2022]
- Murugesh Manthiramoorthi. (n.d.) *"Topic Modelling Techniques in NLP",* OpenGenus IQ: Computing Expertise & Legacy. Link: https://iq.opengenus.org/topic-modelling-techniques/ [Last Accessed: 24 Mar. 2022]
- MonkeyLearn Blog. (2019). *Understanding TF-ID: A Simple Introduction*. [online] Link: https://monkeylearn.com/blog/what-is-tf-idf/ [Accessed 27 Mar. 2022].

O

- OpenGenus. (n.d.) *"Types of RNN (Recurrent Neural Network)".* Link: https://iq.opengenus.org/types-of-rnn/ [Last Accessed: 23 Mar. 2022]

S

- Sciforce. (2018) *"Word Vectors in Natural Language Processing: Global Vectors (GloVe)",* Medium. Link: https://medium.com/sciforce/word-vectors-in-natural-language-processing-global-vectors-glove-51339db89639 [Last Accessed: 27 Mar 2022]
- scikit-learn. (2022a). *sklearn.model_selection.GridSearchCV*. [online] Link: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html [Accessed 27 Mar. 2022].

- scikit-learn. (2022b). *sklearn.naive_bayes.MultinomialNB*. [online] Link: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html [Accessed 27 Mar. 2022].

- scikit-learn. (2022c). *1.5. Stochastic Gradient Descent*. [online] Link: https://scikit-learn.org/stable/modules/sgd.html [Accessed 27 Mar. 2022].

- Silge, J. (2017). *4 Relationships between words: n-grams and correlations | Text Mining with R*. [online] Tidytextmining.com. Link: https://www.tidytextmining.com/ngrams.html [Accessed 27 Mar. 2022].

- spaCy. (2022a). *Sentencizer*. [online] Link: https://spacy.io/api/sentencizer [Accessed 27 Mar. 2022].

- spaCy. (2022b). *spaCy 101: Everything you need to know*. [online] Link: https://spacy.io/usage/spacy-101 [Accessed 27 Mar. 2022].

- Srinidhi, S. (2020). *Lemmatization in Natural Language Processing (NLP) and Machine Learning.* [online] Medium. Link: https://towardsdatascience.com/lemmatization-in-natural-language-processing-nlp-and-machine-learning-a4416f69a7b6 [Accessed 27 Mar. 2022].

T

- Tracyrenee. (2021) *"How to count occurance of words using sklearn's CountVectorizer",* Medium. Link: https://medium.com/geekculture/how-to-count-occurance-of-words-using-sklearns-countvectorizer-a9a65815b1e6 [Last Accessed: 27 Mar. 2022]
- TR517 (2018). *How to speed up spaCy lemmatization?* [online] Stack Overflow. Link: https://stackoverflow.com/questions/51372724/how-to-speed-up-spacy-lemmatization [Accessed 27 Mar. 2022].

V

- Vivek Padia. (2020) "How to count occurance of words using sklearn's CountVectorizer", Medium. Link: https://medium.com/aubergine-solutions/scratching-surface-of-rnn-gru-and-lstm-with-example-of-sentiment-analysis-8dd4e748d426 [Last Accessed: 27 Mar. 2022]

W

- Wikipedia Contributors (2022). *Support-vector machine*. [online] Wikipedia. Link: https://en.wikipedia.org/wiki/Support-vector_machine [Accessed 25 Mar. 2022].