Machine Learning Project

Latent Dirichlet Allocation(LDA) and Non Negative Matrix Factorization (NMF) for Topic Modeling

Team Members: Teresa Chu (B00690148)
Trupti Bhagoorkar (B00820305)
Joseph E Rivas (B00358083)

Abstract

Topic modeling is used in machine learning and natural language processing. Topic modeling is a statistical model which is used to analyze large text data in order to extract topics such as customer reviews on movies or products, news stories, emails of customer complaints. The outcome of the topic modeling is to understand what people are talking about and to understand their opinion or problems. In this project, our team plans to discover the topics and opinions provided by the text from product reviews using Latent Dirichlet Allocation (LDA) and Non Negative Matrix Factorization (NMF) models.

Dataset

- **Consumer Reviews on Amazon Products**: more than 5000 reviews on Amazon products centered mostly around Amazon electronics like the kindle or echo.
- **Food Reviews** : more than 500,000 reviews on food products bought from Amazon that span a period of more than 10 years.

Models

- **Latent Dirichlet Allocation (LDA) :** LDA uses an unsupervised approach for finding and observing a bunch of words. This model assumes that documents are produced from a mixture of topics and those topics then generate words based on their probability distribution.
- **Non Negative Matrix Factorization (NMF):** NMF is an unsupervised technique that factorizes high dimensional vectors into a lower dimensional.

Experiment and Findings

Dataset 1 using LDA model

For the amazon technological product reviews, first we performed data(text) processing on the dataset. For that we used nltk and spacy libraries to remove stopwords, punctuations, new lines and special characters. After the tokenization process and data cleaning we created bigrams(two words frequently occurring together) and trigrams(three words frequently occurring together). For this whole process we used python's Gensim package. The two main inputs to the LDA topic

model are the dictionary and the corpus(collection of documents). Now we have everything required to train the LDA model. In addition to the corpus and dictionary, you need to provide the number of topics. The LDA model is built with 10 different topics where each topic is a combination of keywords and each keyword contributes a certain weightage to the topic. As shown below,

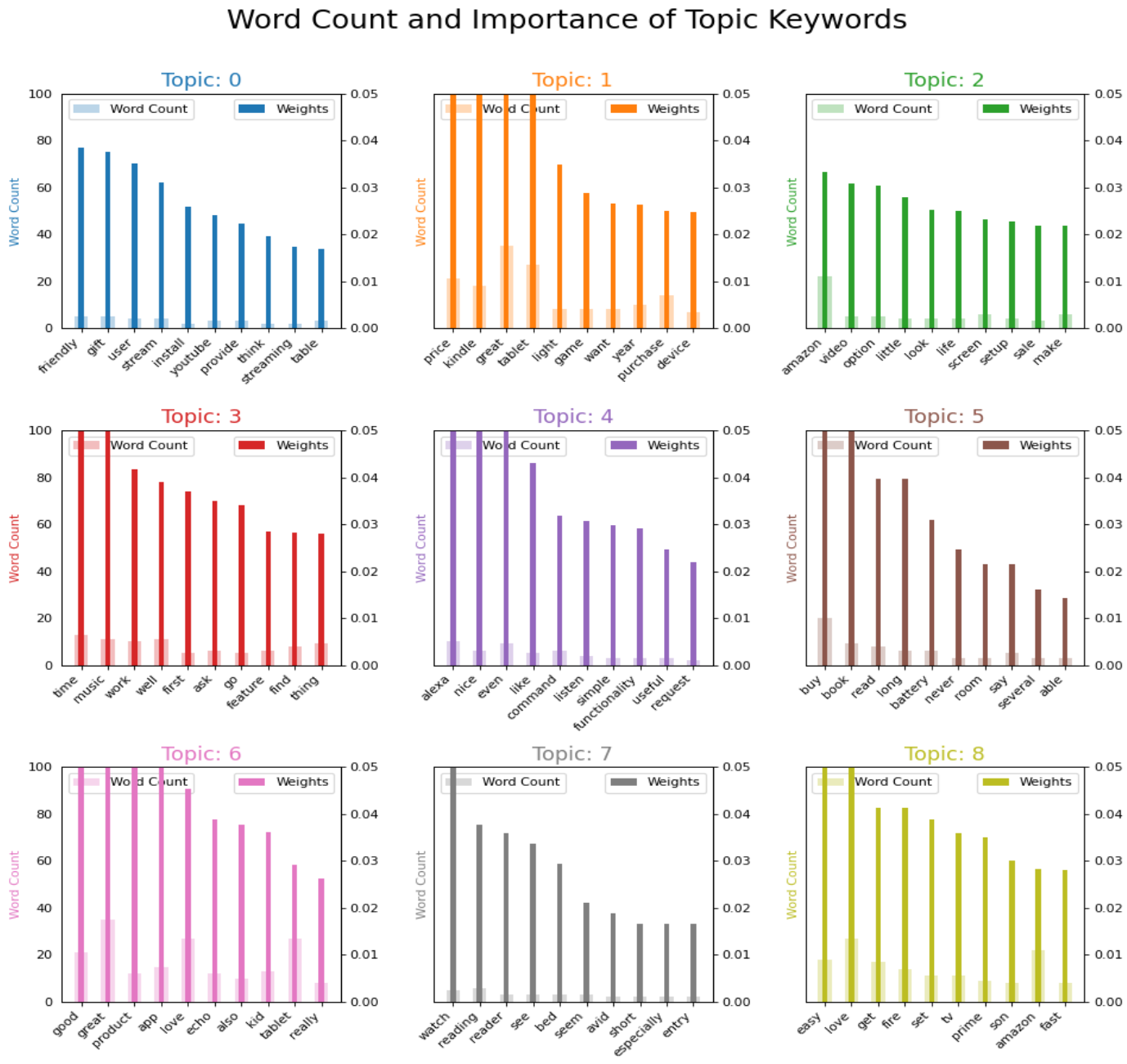
```
['convenient', 'readers', 'online', 'works', 'quickly', 'great']
[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1)]
[(0,
  '0.021*"love" + 0.020*"echo" + 0.016*"tablet" + 0.012*"great" + '
  '0.011*"alexa" + 0.010*"product" + 0.010*"better" + 0.010*"display" + '
  '0.010*"best" + 0.010*"name"'),
 (1,
  '0.020*"tablet" + 0.020*"great" + 0.016*"easy" + 0.013*"fun" + '
  '0.013*"options" + 0.013*"price" + 0.013*"games" + 0.013*"can" + '
  '0.013*"unit" + 0.010*"one"'),
 (2,
  '0.016*"model" + 0.015*"like" + 0.013*"lighting" + 0.012*"reading" + '
  '0.011*"also" + 0.010*"price" + 0.010*"previous" + 0.009*"amazing" + '
  '0.008*"amazon" + 0.008*"best"'),
 (3,
  '0.018*"great" + 0.015*"works" + 0.013*"tablet" + 0.012*"go" + 0.010*"set" + '
  '0.010*"cost" + 0.009*"echo" + 0.009*"easy" + 0.009*"speaker" + '
  '0.008*"gift"'),
 (4,
  '0.028*"tablet" + 0.016*"kids" + 0.015*"amazon" + 0.015*"can" + '
  '0.011*"great" + 0.011*"good" + 0.011*"control" + 0.011*"children" + '
  '0.006*"everything" + 0.006*"tablets"'),
 (5,
  '0.043*"great" + 0.024*"tablet" + 0.016*"can" + 0.015*"easy" + '
  '0.014*"product" + 0.013*"little" + 0.010*"amazon" + 0.010*"loves" + '
  '0.009*"quality" + 0.009*"fire"'),
 (6,
  '0.019*"love" + 0.016*"tv" + 0.014*"faster" + 0.014*"kindle" + '
  '0.013*"tablet" + 0.013*"amazon" + 0.012*"can" + 0.012*"fast" + '
  '0.012*"really" + 0.010*"best"'),
 (7,
  '0.018*"kindle" + 0.012*"worth" + 0.012*"light" + 0.012*"smart" + '
  '0.012*"gadget" + 0.010*"bought" + 0.010*"portable" + 0.009*"money" + '
  '0.009*"got" + 0.008*"thing"'),
 (8,
  '0.023*"great" + 0.015*"amazon" + 0.015*"bought" + 0.014*"can" + '
  '0.012*"much" + 0.012*"tablet" + 0.010*"buy" + 0.010*"kindle" + 0.009*"fire" '
  '+ 0.009*"easy"'),
 (9,
  '0.013*"got" + 0.011*"perfect" + 0.011*"reading" + 0.010*"everything" + '
  '0.010*"well" + 0.009*"great" + 0.009*"love" + 0.009*"slow" + 0.009*"also" + '
  '0.008*"speed"')]
```

Result of LDA model for Amazon Tech Dataset

We wanted to cover additional aspects of the LDA model and it's usage. As such, not only did we find the 10 dominant topics for this dataset, we also discovered what percentage level these topics encompass as shown below.

	Document_No	Dominant_Topic	Topic_Perc_Contrib	\
0	0	8.0	0.8714	
1	1	3.0	0.9331	
2	2	2.0	0.9100	
3	3	1.0	0.9308	
4	4	8.0	0.9150	
5	5	6.0	0.9308	
6	6	1.0	0.9526	
7	7	1.0	0.9357	
8	8	3.0	0.8473	
9	9	6.0	0.9550	

To visualize the data in a graphical format we used matplotlib library. When it comes to the keywords in a topic, it is important to know the weight or value of each keyword. Along with that we have plotted how frequently the words have appeared in the documents.



Graphical representation of LDA model Result

Dataset 2 using LDA model

For the food dataset, we used an alternative approach to processing the data. Instead, we used a file containing common English stopwords. Given that this file was not as extensive as the nltk stopword library, our team further processed the data by removing words with length less than or equal to 2. That way it could remove unnecessary words in the reviews in the dataset. Like in the amazon tech product, we also removed punctuations, new lines and other special characters using regular expressions to match and remove these symbols.

In the food dataset, we explored several parameters in the LDA model by using the sklearn module in order to train our LDA model to get the best number of components and the learning decay. We ran through the LDA model over a range of 10-30 topics and a learning decay over a range of 0.5-0.9. Using the sklearn LDA model, we learned that the best number of topics was 10 while the best metric for the learning decay was 0.5. This was a result of this model attaining the best log likelihood score and model perplexity score where we wanted the log likelihood score to be maximized and the perplexity score was minimized.

```
Best Model's Params:  {'learning_decay': 0.5, 'n_components': 10}
Best Log Likelihood Score:  -33.04350303032485
Model Perplexity:  217.26711781201942
```

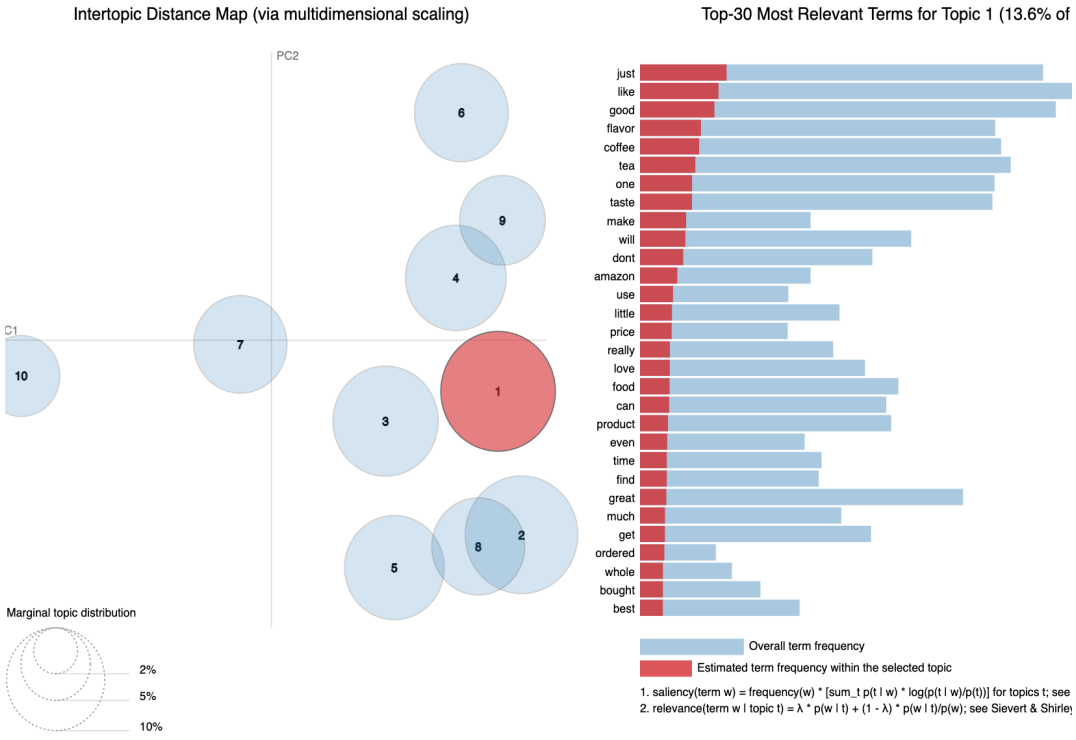
Finding Model parameters based on sklearn LDA module

It is for this reason that when using the gensim LDA model in order to be able to graph the results with the pyLDAvis module, we used 10 topics. To avoid miscellaneous words from affecting the LDA model, we filtered out words that had length less than or equal to 2 in addition to removing stopwords. That way, we would be able to prevent preposition words from appearing in the reviews which would have little contribution to the LDA model created. In addition, our team also filtered out reviews that had less than 60 words to avoid short reviews that had little substance.That way our LDA model would be able to more accurately reflect on keywords appearing in the 10 topics.

After running the LDA model and finding 10 topics over a sample within the dataset and central keywords within these 10 topics, it was interesting to learn that a majority of the terms in these topics used words like “good”, “like” or “great” indicating that most of these reviews were favorable to the product. Another noticeable feature learned while using topic modeling was certain keywords appearing in the topic like “cats”, “dogs” and “kids”. This may indicate that the audience for the food products provided in the dataset lies with an unexpected demographic. Therefore, not only can we gain a better understanding of the user experience of these products but this could also potentially help sellers understand which audience they need to cater to.

```
[(0,
  '0.011*"good" + 0.010*"food" + 0.008*"just" + 0.007*"one" + 0.007*"great" + '
  '0.007*"like" + 0.007*"can" + 0.006*"better" + 0.006*"love" + '
  '0.006*"really"'),
 (1,
  '0.011*"just" + 0.010*"like" + 0.010*"good" + 0.008*"flavor" + '
  '0.008*"coffee" + 0.007*"tea" + 0.007*"one" + 0.007*"taste" + 0.006*"make" + '
  '0.006*"will"'),
 (2,
  '0.011*"like" + 0.008*"one" + 0.007*"can" + 0.007*"just" + 0.005*"made" + '
  '0.005*"much" + 0.005*"ive" + 0.005*"flavor" + 0.004*"little" + 0.004*"get"'),
 (3,
  '0.009*"just" + 0.007*"food" + 0.007*"dog" + 0.006*"bag" + 0.006*"product" + '
  '0.005*"will" + 0.005*"like" + 0.005*"good" + 0.005*"flavor" + 0.005*"one"'),
 (4,
  '0.009*"flavor" + 0.007*"one" + 0.006*"great" + 0.006*"coffee" + '
  '0.006*"like" + 0.006*"get" + 0.006*"good" + 0.006*"taste" + 0.005*"can" + '
  '0.005*"love"'),
 (5,
  '0.017*"like" + 0.013*"tea" + 0.011*"great" + 0.011*"coffee" + 0.009*"taste" '
  '+ 0.008*"one" + 0.007*"just" + 0.005*"much" + 0.005*"used" + 0.005*"good"'),
 (6,
  '0.011*"coffee" + 0.009*"great" + 0.009*"like" + 0.007*"tea" + 0.007*"taste" '
  '+ 0.006*"good" + 0.006*"just" + 0.006*"flavor" + 0.005*"time" + '
  '0.005*"food"'),
 (7,
  '0.012*"good" + 0.010*"like" + 0.008*"product" + 0.007*"flavor" + '
  '0.007*"coffee" + 0.006*"just" + 0.006*"taste" + 0.006*"will" + 0.005*"one" '
  '+ 0.005*"food"'),
 (8,
  '0.009*"flavor" + 0.008*"like" + 0.008*"good" + 0.006*"taste" + 0.006*"tea" '
  '+ 0.006*"little" + 0.005*"get" + 0.005*"one" + 0.005*"product" + '
  '0.005*"will"'),
 (9,
  '0.010*"like" + 0.008*"just" + 0.008*"tea" + 0.007*"get" + 0.007*"taste" + '
  '0.007*"one" + 0.006*"great" + 0.005*"chocolate" + 0.005*"tried" + '
  '0.005*"much"')]
```

Result of LDA model for Food Dataset



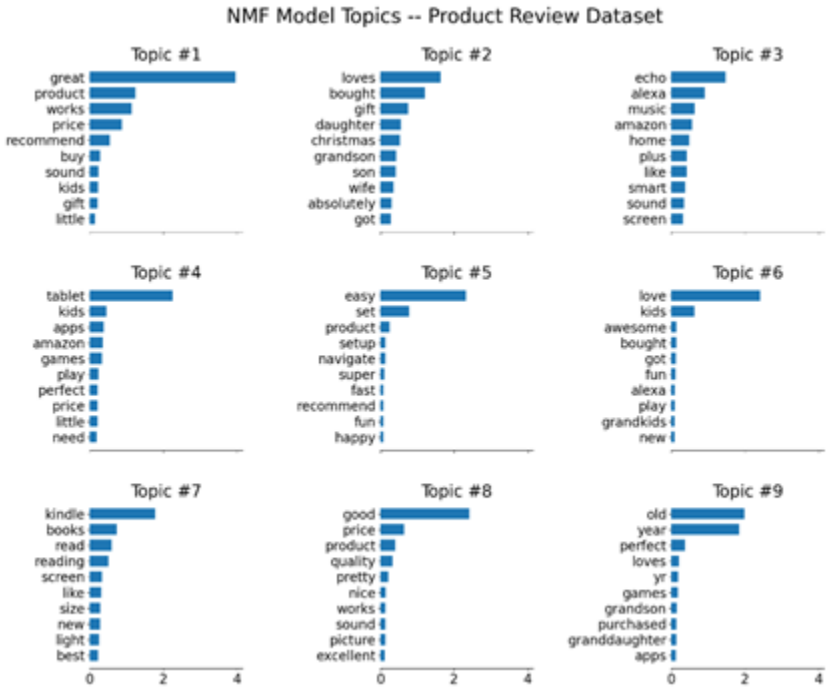
10 Topics and the Top 30 Relevant Terms for each Topic using pyLDavis

Non Negative Matrix Factorization

Similarly to the LDA model, the data was preprocessed using regex and NLTK stopwords. Punctuation and special characters were also removed. The *gensim.corpora* class was used to create a dictionary from the text. The final *corpus* resulted in a list of lists, with each sublist containing the text for a specific product review. The *sentences* variable was used to join all the separated words into sentences again for input into the NMF model.

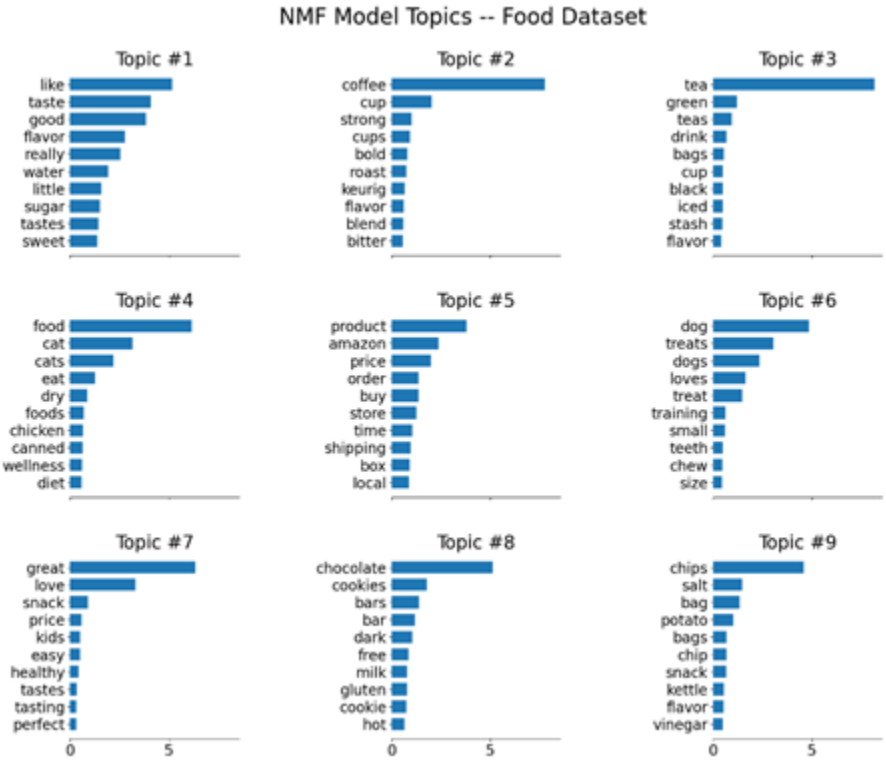
For the NMF model, the *TfidfVectorizer* class was applied to the text. This is used to convert the raw document data into a matrix of term frequency-inverse document frequency (TF-IDF) features. These feature vectors are basically used to identify how important each word is to a specific body of text. In this case, the TF-IDF vectors indicate which words are important in each review.

With the data preprocessed, the NMF class is initialized. The NMF class is imported from *sklearn.preprocessing* and is initialized with the *n_components* parameter set to the global variable *NUM_TOPICS*, which in our implementation was selected to be 9. This will tell the model to extract a total of 9 topics from the corpus. The second NMF class parameter is the *init="nnsvd"* parameter. NNSVD stands for nonnegative double singular value decomposition; this parameter helps with handling sparse matrices from the TF-IDF feature vectors. The text data is passed in to the NMF class to fit the data using the *fit()* function. To get the specific words for each topic, the *get_feature_names()* function (from the *TfidfVectorizer* class) is called. To visualize the topics and words, the *plot_words()* function was created. This function takes in the NMF model, feature names, and number of top words (defaulted to 10) in order to generate plots. This functionality was modified from the official scikit-learn documentation. Another helper function *get_NMF_topics()* is used to simply get the text output of the topics and top words in a csv file. The topics and top words for the Amazon product reviews dataset can be observed below.



It is very interesting to note the word clustering for each topic. For example, Topic #3 appears to group words related to describe technology. Such words include: “smart”, “sound”, “screen”, “alexa”, which are all part of technology devices. Topic #1 appears to describe how good a product is using words like “recommend”, “buy”, “great”. Topic #7 looks like it is related to reading or reading devices using words like “kindle”, “read” and “books”. Interestingly, Topic #9 appears to signify older buyers – words like “grandson”, “granddaughter”, “old” seem to describe products that grandparents may have purchased for their grandchildren.

The NMF model was generated again, this time with the food reviews dataset. The processing was conducted in the same manner, with the only change being the dataset. Since this dataset was much larger, the entire program took several minutes longer to complete. The results for the topic modeling using the food review dataset are shown below.



The topic and word groupings for this dataset were much more interesting. The topics and top words in each topic are much more distinct from one another. For example, Topic #2 is clearly centered on coffee products, with the top words including “coffee”, “cup”, “strong”, and “roast”. Topic #3 is centered around tea products. Topics #4 and #6 are focused on cats and dogs, respectively. Topic #8 describes food containing chocolate; Topic #9 describes potato chips. These topics are incredibly observable and it is easy to see how each group is structured thanks to the top words.

One thing to note is the comparison between the topic modeling using LDA and the topic modeling using NMF. There were many similarities between both models. For example the LDA and NMF models both produced clusters around coffee and tea, and also around dogs and cats. Both models produced distinct clusters, however, both models also included different top words within their clusters. This difference is likely due to the fact that both models used slightly different preprocessing steps. For example, the NMF did not use spacy for lemmatization. We found this process very time consuming when using the spacy package, especially if using a larger dataset like the food reviews, which was nearly 300mb in size. Both models did use NLTK, regex, and gensim for initial preprocessing and to create the text corpus for input into the respective models.

Conclusion

From these experiments we discover how to extract good quality topics that are clear, segregated and meaningful. This depends on the quality of text preprocessing and the strategy of finding the optimal number of topics. In addition to this, it is highly important to understand the dataset on which various techniques are applied.

Contribution

Teresa Chu : Data Processing, Model Evaluation(LDA), Data Visualization, Report, PPT
Trupti Bhagoorkar : Data Processing(NLTK), Model testing(LDA), Data visualization, Report
Joseph E Rivas : Data Processing(NLTK), Model Testing(NMF), Data Visualization, Report, PPT

Github Link: https://github.com/tbhagoo1/Machine-Learning-Project

References

1. https://en.wikipedia.org/wiki/Topic_model
2. https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation
3. https://www.nltk.org/
4. https://pypi.org/project/gensim/
5. https://towardsdatascience.com/end-to-end-topic-modeling-in-python-latent-dirichlet-allocation-lda-35ce4ed6b3e0
6. https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf
7. https://radimrehurek.com/gensim/models/ldamodel.html
8. https://medium.com/analytics-vidhya/topic-modeling-using-gensim-lda-in-python-48eaa2344920
9. https://medium.com/pew-research-center-decoded/making-sense-of-topic-models-953a5e42854e
10. https://scikit-learn.org/stable/auto_examples/applications/plot_topics_extraction_with_nmf_lda.html