Indian Institute of Technology Delhi

MTL782 DATA MINING

Customer Support on Twitter

Mridul Ahuja
2018MT10626
Dept.: Mathematics

Kartikeya Rai
2018MT60811
Dept.: Mathematics

Facilitator
Niladri Chatterjee
Professor
Department of Mathematics
IIT Delhi

July 24, 2021

Contents

1 Introduction and Motivation

Among all forms of data available for analysis, Natural Language datasets provide the deepest insights into human behaviour and thus there has been a great push for innovation in the field of *Natural Language Processing (NLP)*. But many of the datasets in use are not composed of the modern day English actually used in conversations. The Customer Support on Twitter dataset can effectively bridge this gap and is an enormous, modern compilation of tweets and responses which can greatly help in studying customer support practices for industry leaders and their impact. It can also prove to be useful for testing and generation of more innovative data mining and natural language understanding techniques.

It has major advantages over other conversational datasets, such as:

- *Variety*: Customers belong to several diverse demographics and come from a much broader segment.
- *Specificity*: Unlike unconstrained conversational datasets, in this corpus, consumers contact customer support only for specific problems, thus making it much more focused with a relatively small manifold of problems.
- *Condensed*: Owing to the character limit (*280 characters*) imposed on each tweet, the responses contain shorter description of problems and solutions which makes data processing relatively faster.

As many as thirty-three percent of Americans say they'll consider switching companies after just a single instance of poor service. This goes to show the rising importance of customer service and how much it is valued in the modern world. But how can one deliver great customer service when one's teams are bombarded with more data than ever before? In this day and age, customer service can break or make a company and simply having a great product is no longer the only basis for a company's success - constantly delivering stellar customer service is extremely vital in order to stand out from the crowd.

2 Past Work and its Relevance to our Problem

We conducted an extensive literature review to learn more about the field and also the state-of-the-art data mining and NLP techniques. The fundamental techniques we reviewed are:

1. **Sentiment Analysis:** Its main aim is to detect the person's mood, behavior and opinion from text documents. With the expanded use of social networking sites, sentiment analysis techniques have started to use these sites' public data to do sentiment analysis studies in different sociological areas, such as politics, sociology, economy and finance.
   Various Classification Algorithms have been developed for gauging the sentiment and polarity of a given text. They can be broadly grouped into two categories:
   - **Supervised Methods**: Naive Bayes, Maximum Entropy, Support Vector Machine (SVM) Classifiers
   - **Lexical** or **Unsupervised Methods** - Lexicon, Dictionary or Corpus based techniques

   The aim while performing twitter sentiment analysis is classifies the tweets in different sentiment classes accurately. In this field of research, various techniques have evolved, which come up with methods to train a model and then test it to check its effectiveness. However, performing sentiment analysis is challenging on twitter tweets due to limited tweet size (140 characters[1]), use of imperfect language or 'slang' words, and the variety in user's expression of sentiments. Also, the different features provided by Twitter such as allowing hashtags, URLs, etc also make the processing of data harder.
2. **Topic Modelling:** It is the task of identifying which underlying concepts are discussed within a collection of documents, and determining which topics each document is addressing. Essentially, it is an unsupervised technique that's capable of scanning a set of documents, detecting word and phrase patterns within them, and automatically clustering word groups and similar expressions that best characterize the documents. This type of modelling has many applications; for example, topic models may be used for information retrieval (IR), to identify influential individuals on a social media platform and so on.

[1] This figure corresponds to the time period to which the tweets from our dataset belong. Presently the limit is 280 characters.

The conventional methods used for topic modelling are probabilistic latent semantic analysis (pLSA) and Latent Dirichlet Allocation (LDA). These are models that are frequently used to capture the set of latent "topics" over documents within a corpus. But these usually work well on documents with a few hundred words and may not give satisfactory results on shorter length texts such as tweets.
However, automatically detecting topics from tweets could be extremely useful for companies responding to a plethora of queries on a daily basis and could streamline their customer support efforts.

3. **Learning Word Embeddings:** A word embedding is a *low-dimensional, dense* and *real valued vector representation* of a word. Embedding of a word captures both its syntactic and semantic aspects and they have been successfully used in many NLP tasks. It is a powerful tool widely used in NLP tasks, including semantic analysis, information retrieval, dependency parsing, question answering and machine translation. However they are usually generated from a large text corpus, whereas tweets are short, noisy and have unique lexical and semantic features that are different from other types of text.

3 Our Scheme

In what follows we describe the dataset we worked on, and discuss the data preprocessing and cleaning methods employed. Then we detail the different approaches adopted by us in order to tackle the various problems that we worked on. Next we describe the novel aspects of our scheme and finally present our results.

3.1 Description of the Data

The *Customer Support on Twitter* dataset is a large corpus of tweets and replies authored and made available by Stuart Axelbrooke at Thought Vector. Every included conversation has at least one request from a consumer and at least one corresponding response from a company. The dataset is a CSV file with about 2.81 million rows and 7 columns. Each row corresponds to a tweet, while the columns describe various attributes of the tweets. The different column fields are as follows:

(a) `tweet_id` - A unique, anonymized ID for the tweet; referenced by the `response_tweet_id` and `in_response_to_tweet_id` fields

(b) `author_id` - A unique, anonymized ID for each user; @*s* in the dataset have been replaced with their associated anonymized user ID

(c) `inbound` - A binary field that indicates whether a tweet is "inbound" to a company doing customer support on Twitter

(d) `created_at` - Date and time when the tweet was sent

(e) `text` - Tweet content with personal information masked; sensitive information like phone numbers and email addresses have been replaced with mask values like `__email__`

(f) `response_tweet_id` - IDs of tweets that are responses to the current tweet (comma separated)

(g) `in_response_to_tweet_id` - ID of the tweet that the current tweet is in response to, if any.

	tweet_id	author_id	inbound	created_at	text	response_tweet_id	in_response_to_tweet_id
0	1	sprintcare	False	Tue Oct 31 22:10:47 +0000 2017	@115712 I understand. I would like to assist y...	2	3.0
1	2	115712	True	Tue Oct 31 22:11:45 +0000 2017	@sprintcare and how do you propose we do that	NaN	1.0
2	3	115712	True	Tue Oct 31 22:08:27 +0000 2017	@sprintcare I have sent several private messag...	1	4.0
3	4	sprintcare	False	Tue Oct 31 21:54:49 +0000 2017	@115712 Please send us a Private Message so th...	3	5.0
4	5	115712	True	Tue Oct 31 21:49:35 +0000 2017	@sprintcare I did.	4	6.0

Figure 1: A sample view of the data

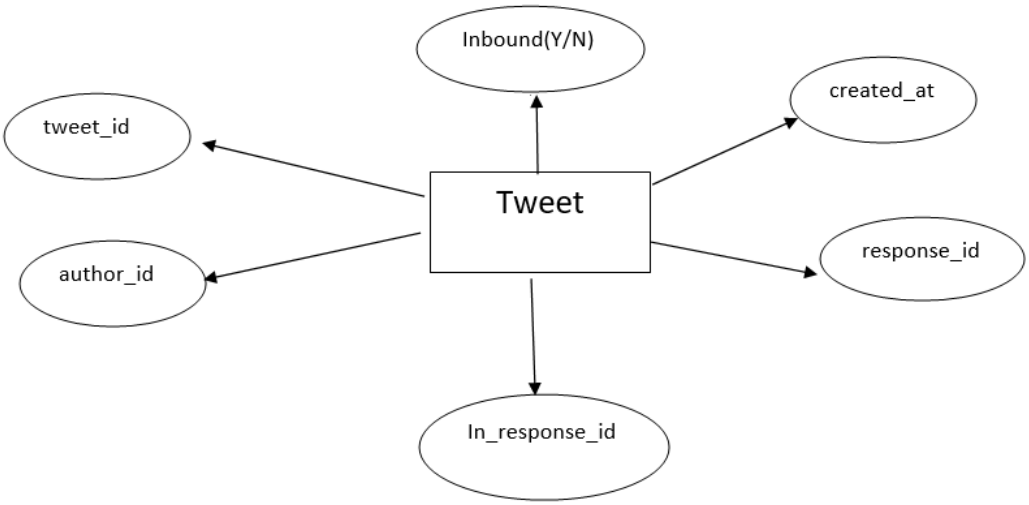

Figure 2: The Entity Relationship (ER) Diagram of our dataset

3.2 Data Preprocessing

Data preprocessing is an essential part of data mining. Data collection or generation methods are often loosely controlled and may result in impossible values, out of range values or even missing data. Analyzing data that has not gone through proper cleansing and processing may lead to misleading results and false conclusions. Below we describe the various techniques that we employed to clean and process the data before any further analysis.

(a) **Removal of missing values** The `response_tweet_id` and `in_response_to_tweet_id` were found to contain a lot of missing values. We discarded these columns for most of our work.

(b) **Lower casing** All the text was lower cased so that different instances of the same word like "WORD", "Word", "word" etc are treated the same way. This reduces the duplication of words which aids text featurisation techniques such as *CountVectorizer*, *TfidfVectorizer* etc in finding the correct counts. It must be noted that indiscriminate use of this technique may lead to the loss of data. We do not employ this method while performing sentiment analysis as upper case words convey strong emotions and lower casing would lead to a loss in valuable information.

(c) **Removal of punctuations** This is a common preprocessing technique that is applied so that instances like "hurray" and "hurray!" are treated the same. The following punctuations were removed: ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [\\ ] ^ _ { | } ~ `.

(d) **Removal of stop words** Stop words are a set of commonly used words in any language. For example, "the", "is" and "and" are stop words in English. Usually stop words do not provide valuable information and are removed, allowing applications to focus on the important words instead.

(e) **Removal of frequent words** Frequently occurring words such as "us", "get", "please" etc are almost devoid of meaning, provide very little semantic content, and thus were removed. Further because these words have high counts, most scoring functions are rewarded for predicting their counts, more than they are for the correct predictions of other words. We note that this may cause loss of data for the sentiment analysis use case.

(f) **Removal of very rare words** Rare words are likely to be the names of persons, places etc and are expected to have little bearing on the meaning of a tweet. Here the objective is to remove such words. This is very similar to the previous technique, but there is another aspect - due to the rare occurrences of these words, their association with others is dominated by noise and presents problems in learning.

(g) **Stemming** Stemming is the process of reducing inflected words to their base or root form. Stemming effectively modifies all words that have the same meaning but differ in grammatical form by reducing them to a simpler base word. We used NLTK's implementation of the Porter stemming algorithm.

(h) **Lemmatization** Lemmatization is the algorithmic process of determining the lemma (dictionary form) of a word based on its intended meaning. This is closely related to stemming. The difference is that a stemmer operates on a single word without knowledge of the surrounding context, while a lemmatizer looks at the surrounding context. Further the lemma given by a lemmatizer always belongs to the language. This need not be true in stemming. We used the *WordNetLemmatizer* from NLTK.

(i) **Removal of emojis and emoticons** Some datasets have a huge number of emojis or emoticons which one might wish to remove before further processing. We used regular expressions to search for and remove all emojis and emoticons that were present. Since emojis and emoticons are used to express powerful emotions, we remark that this must not be done in case of sentiment analysis.

(j) **Conversion of emoticons and emojis to words** As mentioned above, removal of emojis and emoticons causes loss of valuable emotion related data, and it is not recommended while performing sentiment analysis. For this use case, we use dictionaries of emojis and emoticons to replace their instances by a textual counterpart that conveys a similar meaning.

(k) **Removal of URLs, HTML tags, phone nos., email IDs etc** Such information is usually quite specific and no additional semantic information can be gleaned from it. We used regular expressions to search for each of these types of expressions and remove them.

(l) **Chat word conversion** The objective of this step is to convert commonly used slang to its counterpart in proper English, e.g.: "ASAP" is converted to "As Soon As Possible". Since tweets have a fairly low character limit, the usage of slang chat words is quite common, making this an especially important preprocessing step for Twitter data.

(m) **Spelling Correction** This is a very common step in text processing as some techniques require one to correct the spellings before applying them, e.g.: identification and removal of stop words. We used the Python package *pyspellchecker* for spelling correction.

(n) **Removal of non-english words** Some tweets may contain words of a language different from English. In this case there are typically two choices - translate the word, or remove it altogether. We have adopted the latter strategy.

3.3 Techniques used

We explored and experimented with different techniques and models for the different tasks that we attempted. We now describe these in some detail.

3.3.1 Word2vec

A word embedding denotes a representation of words, typically as a real valued vector, that is used for language processing tasks. The encoding is such that words with closer vector representatives (with respect to a norm on the vector space) are expected to be closer in meaning. Word2vec is a combination of two-layer neural network based models that are used to produce these word embeddings. The word2vec algorithm accepts a text corpus as input and outputs a vector representation for each word in the corpus.

The algorithm can utilize two different model architectures and thus is available in two flavors, namely CBOW (continuous bag-of-words) and continuous skip-gram. In the CBOW architecture, the model tries to predict the current word from a window of surrounding context words, whereas in the continuous skip-gram architecture the current word is used to predict the surrounding window of context words. We have made use of the CBOW architecture made available by `gensim` in our implementation.

The size of the context window is an adjustable hyperparameter that determines the number of surrounding words that are to be included as context words. For example, consider the simple sentence, "the quick brown fox jumps over the lazy dog", with a window of size 2. Here one gets training samples like ([the, brown], quick), ([quick, fox], brown), ([the, dog], lazy) etc. We used a window size of 2 in our implementation.

In the training phase, a word2vec model may rely on either hierarchical softmax or negative sampling for learning the optimal weights. In the former approach, the softmax function is used as the last activation function of the neural network. It takes an input $\mathbf{z}$ and normalizes it into a probability

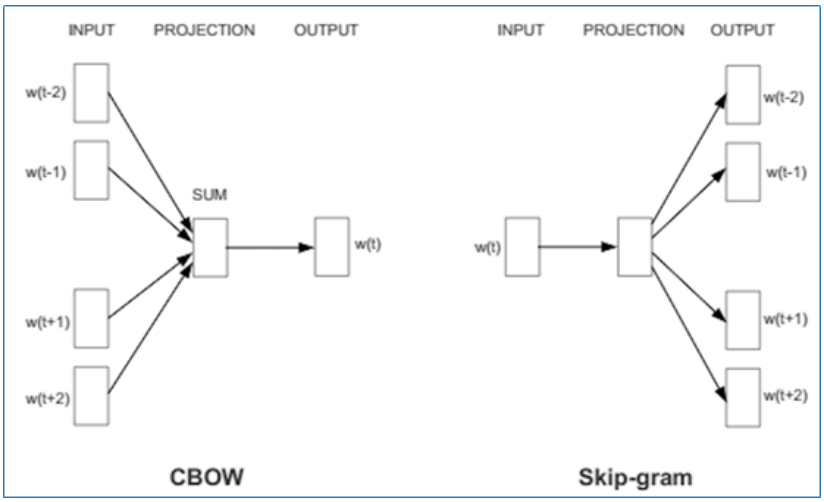

Figure 3: The difference between the CBOW and skip-gram architectures

distribution by applying the softmax function to it. The standard softmax function $\sigma : \mathbb{R}^K \to [0,1]^K$ is defined as

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K.$$

We have used a neural network that uses vectors of dimensionality 300 for the words. This combined with a large vocabulary of words, found across 2.8 million tweets, results in huge weight matrices for the layers of the neural network. The conventional softmax based approach would require updating all of these weights for each of the millions of training samples.

We have, instead, used the negative sampling approach which addresses this by having each training sample only modify a small percentage of the weights, rather than all of them. The weights that are updated correspond to a random selection of `negative` (hyperparameter) words among those that are negative, that is have a 0, in the one-hot output vector of the training sample. (The weights for our "positive" word are also updated.) These "negative" samples are selected using a unigram distribution, where more frequent words are more likely to be selected. In our implementation, we used `negative` = 2.

3.3.2 Topic Modelling using Latent Dirichlet Allocation (LDA)

LDA is a form of unsupervised learning techniques that views the entire corpus as an unordered bag of words and operates on the fundamental assumption that each document (in our case, the tweets) are formed by first choosing a particular number of topics and then choosing particular words from each of those topics. Building on this assumption, LDA models the generation of documents within a document space by employing the following process:

(a) A mixture of **k** topics, θ, is sampled from a Dirichlet prior, which is parameterized by α

(b) A topic $\mathbf{z}_n$ is sampled from the multinomial distribution, $\mathbf{p}(\theta; \alpha)$, which models $\mathbf{p}(\mathbf{z}_n = i|\theta)$;

(c) A word, $\mathbf{w}_n$, is then sampled (given the topic $\mathbf{z}_n$) via the multinomial distribution $\mathbf{p}(\mathbf{w}|\mathbf{z}_n)$;.

This distribution can be symmetric or unsymmetric where the former implies that each of the chosen topic is evenly distributed throughout the document whereas the latter can be used when certain topics are favoured over the others during the generation.

Given a corpus of M documents $\mathbf{D} = \{w_1, ..., w_M\}$, we can make use of the EM algorithm to converge upon the parameters of the distribution and our LDA model.

`gensim` provides an implementation of LDA, however, this gives poor results over shorter documents (such as tweets) and for this reason we introduced novelty by concatenating tweets, using different priors for different topics, etc. to try and get better results (has been discussed later). We used a set of 20,000 tweets at a time and tried to find most frequent, coherent topics in each and identified the keywords associated with these particular topics along with the importance of each keyword for the

topic. We identified a total of 8 such topics by forming the bigram, trigram models on the lemmatized, preprocessed tweets.

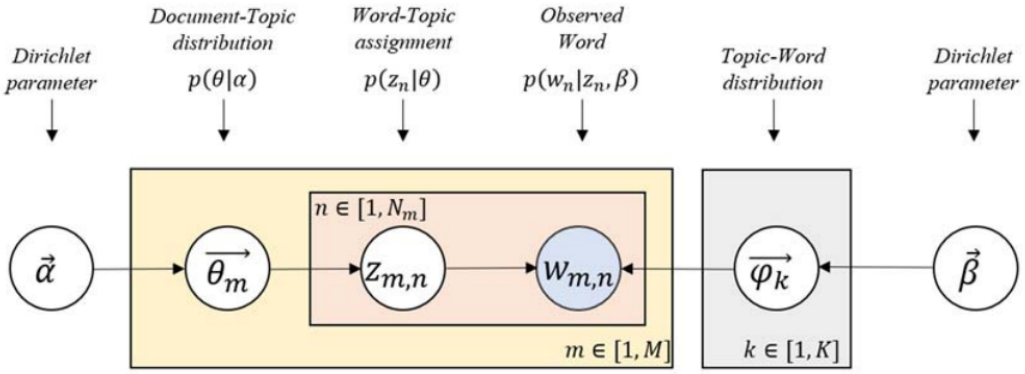

Figure 4: Working of LDA

3.3.3 Sentiment Analysis

Sentiment analysis, also known as opinion mining, is a data mining technique that aimts to detect and quantify positive or negative sentiment in text. The analyzed data quantifies the general public's sentiments or reactions toward certain products, people or ideas and reveal the contextual polarity of the information. It is extremely important in the modern age as it aids businesses in better understanding the overall opinions of their customers.

Sentiment analysis primarily uses methods from natural language processing and machine learning to automatically determine the emotional tone behind conversations. Sentiment analysis algorithms broadly fall into one of the following three classes:

- **Rule-based:** such systems perform sentiment analysis based on a set of manually crafter, predetermined rules.
- **Automatic:** these systems rely on machine learning techniques to learn from data.
- **Hybrid:** this involves combining both rule-based and automatic approaches.

Our work falls into the first category. We used the rule based tool VADER (Valence Aware Dictionary for sEntiment Reasoning) from NLTK. VADER is optimized for social media data and yields good results when using data from sites like Twitter and Facebook. It is a model that is sensitive to both polarity (positive/negative) and intensity (strength) of emotion. It relies on a dictionary that maps lexical features to emotion intensities, known as sentiment scores. The net sentiment score of a text can then be obtained by summing up the intensities of all the words in the text.

3.4 Novel Aspects

3.4.1 Phrase based Topic Modelling

Topic models can help us get a deeper understanding of an extremely large corpus but they suffer from not being very human understandable as these are individual words clubbed together devoid of their original context. In this case it can be better to pick phrases that might be able to convery a more meaningful expression of the tweets.

Following this logic, we tried to compare the frequency of occurrence of pairs compared to their frequency of occurrence individually (so as to not penalize words that occur rarely overall) and tried to find meaningful phrases that are human understandable.

We were able to pick upon quite a few of these phrases that were scattered throughout the tweets and which individually in a topic would not have conveyed much intuition but together as a phrase could be much more human understandable.

Some of these were -

- `private message -> private_message`
- `power cycle -> power_cycle`
- `laughing out loud -> laughing_out_loud`
- `estimate delivery data -> estimate_delivery_date`
- `as soon as possible -> as_soon_as_possible`
- `lightroom classic -> lightroom_classic` (when run separately on Adobe customer tweets)
- `burrito bowl -> burrito_bowl` (when run separately on Chipotle customer tweets)
- `black ops -> black_ops` (when run separately on Playstation customer tweets)

We can see that these just as a list would have provided us much lesser information but when clubbed together along with a different topic modeling technique such as LDA, it can provide deeper insights.

3.4.2 Biterm Topic Modelling (BTM)

Extending the idea of the Phrase based topic modelling, we felt we could use a 'bi-term' model, which explicitly models the generation of unordered pairings in the documents, on the basis that topics are groups of correlated words and this correlation can be understood better by looking at co-occurrences.

This is different from LDA, as it doesn't assume that each document is a collection of several topics but rather that the bi-terms as a whole are a collection of several topics throughout the corpus as opposed to being restricted to a single document (which can help with shorter texts such as our tweets).

Since, this is a new model, it has no implementation in a standard library and we tried to build the model in $C++$ and then using a wrapper in R. This allowed us to test the model efficiently, however, even though we expected it to perform better than LDA, we found very similar results which could be because of either modified use of LDA by us, which was making it perform better than the usual standard, or that our implementation lacks certain details which needs to be explored as a part of the future work.

3.4.3 Analysis of Customer Sentiments

We identified the top brands by volume of tweets and found the average customer sentiment using inbound tweets. This allows us to quantify how happy customers are when they first interact with customer support. We obtained the average customer sentiment scores across the top 20 brands (by volume). This allowed us to identify those firms among the top 20, with whom the customers are the happiest, and also those firms which incite the most negative sentiments among customers. Using the results obtained, we examine (qualitatively) if there is any correlation between a company's perception (sentiment score) and how it is in its line of work.

We grouped the tweets by the company they are addressed to, and obtained the average sentiment value for each customer that addressed that company. Another possible measure of a consumer's satisfaction is the lowest sentiment score that a tweet originating from him/her obtained. These can be used by companies to monitor customer sentiment and track satisfaction levels over time. In cases where the customer sentiment goes below a certain threshold, the company can then reach out to the concerned customer to resolve issues. Herein, we also propose a system where a firm keeps track of the sentiment scores of incoming tweets. This then helps to gauge the severity of requests, and enables the classification of tweets on the basis of their severity. The firm can then assign a greater priority to the more aggrieved customers, thereby achieving quicker responses in urgent cases.

3.5 Results obtained

Our primary focus has been on the following: topic modelling of twitter support discussion using latent Dirichlet allocation (LDA), phrase based modelling, word2vec modelling; comparison of customer sentiments across the top brands and some related tasks. In this section, we discuss the results obtained.

3.5.1 Exploratory Data Analysis

We present some key characteristics of the dataset in the form of graphs. Some other observations are also recorded.

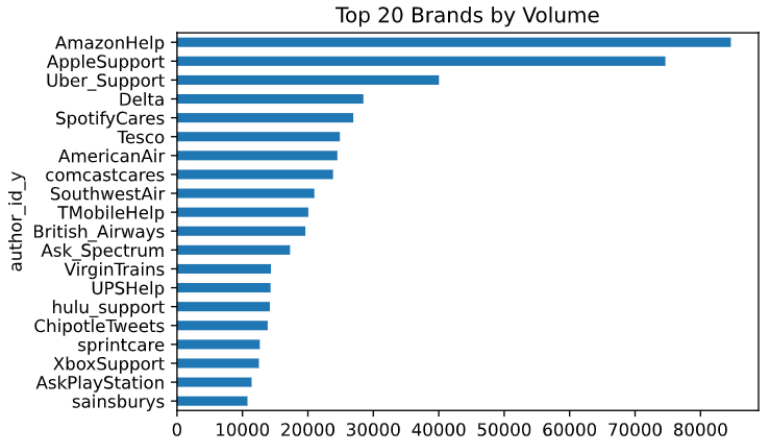

Figure 5: Caption

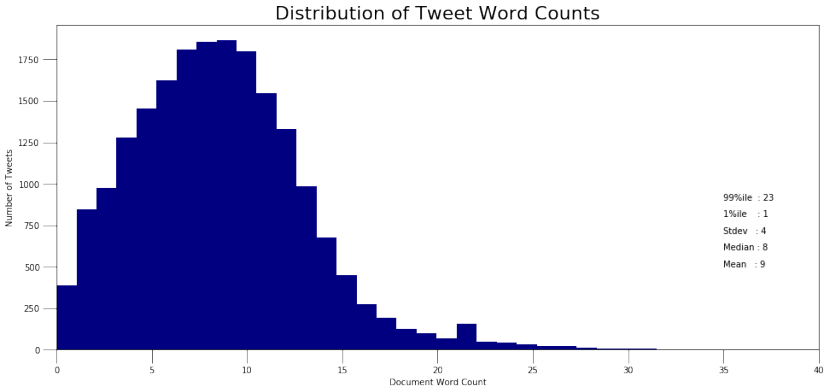

Figure 6: Distribution of Tweet Word Counts

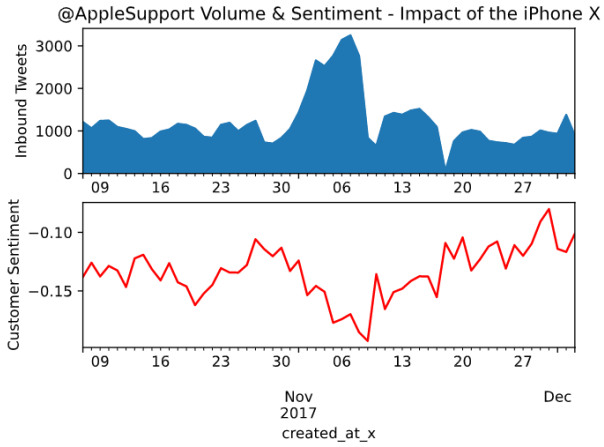

Figure 7: The timeline corresponds to the period in which Apple released iPhone X.

From figure (7) above, one can see that around the date of the release (3 November 2017), the number of inbound tweets to *AppleSupport* sharply rises and there is an overall improvement in customer sentiment.

3.5.2 Word2vec based topic modelling

The *Word2Vec* model from the `gensim` library was trained using the given data for 30 epochs with the CBOW architecture and window size= 2 . We present the results obtained. First we queried the model for words similar to a given word.

```
w2v_model.wv.most_similar(positive=["apple"])
```

```
[('115858', 0.6577697992324829),
 ('applesupport', 0.6511053442955017),
 ('iphone', 0.604471743106842),
 ('115858_applesupport', 0.5964983105659485),
 ('116333', 0.5798109769821167),
 ('iphones', 0.5659255385398865),
 ('applenews', 0.5264140963554382),
 ('phone', 0.5186710357666016),
 ('iphone8', 0.514138400554657),
 ('macbook', 0.513713538646698)]
```

Figure 8: Some similar words are “applesupport”, “iphones” and “phone” which is in agreement with what one might expect.

```
w2v_model.wv.most_similar(positive=["amazonhelp"], topn=20)
```

```
[('amazon', 0.7557227611541748),
 ('115821', 0.7298078536987305),
 ('115850', 0.679477334022522),
 ('115830', 0.6760198473930359),
 ('prime', 0.6248518228530884),
 ('upshelp', 0.5856618881225586),
 ('amazonhelp_115851', 0.5742184519767761),
 ('askebay', 0.5686715841293335),
 ('115851', 0.5574002265930176),
 ('115850_115821', 0.5562988519668579),
 ('prime_member', 0.5489098429679871),
 ('118919', 0.5376055240631104),
 ('amazonprime', 0.5321245789527893),
 ('116090', 0.5310828685760498),
 ('115850_115851', 0.517478346824646),
 ('deliver', 0.5165422558784485),
 ('guaranteed_delivery', 0.5123353600502014),
 ('amazon_prime', 0.5122708082199097),
 ('115830_115851', 0.5092859268188477),
 ('marked_deliver', 0.4995267391204834)]
```

Figure 9: The similar words “amazon”, “prime”, “deliver” etc are as expected.

```
w2v_model.wv.most_similar(positive=["game"], topn=20)
```

```
[('xbox', 0.6837126016616821),
 ('ww2', 0.6638185977935791),
 ('xbox_one', 0.6496511697769165),
 ('cod_ww2', 0.6450698375701904),
 ('play_game', 0.6271839737892151),
 ('play', 0.614237904548645),
 ('ps4', 0.6111302375793457),
 ('atviassist', 0.5983550548553467),
 ('multiplayer', 0.5900417566299438),
 ('xboxsupport', 0.5822678804397583),
 ('xb1', 0.5749655365943909),
 ('codww2', 0.5715525150299072),
 ('player', 0.5660527348518372),
 ('halo_5', 0.5651882886886597),
 ('destiny_2', 0.5601149797439575),
 ('115766', 0.5577322244644165),
 ('gameplay', 0.5574690103530884),
 ('cod_wwii', 0.5529870390892029),
 ('fortnite', 0.5520461797714233),
 ('console', 0.5491738319396973)]
```

Figure 10: The similar words “xbox”, “play”, “ps4” etc are as expected.

Next we queried the model for the similarity between two given words. We list some of the words we tried.

```
w2v_model.wv.similarity("apple", 'iphone')
```

```
0.6044717
```

Figure 11: As expected, there is a high similarity between “apple” and “iphone”.

```
w2v_model.wv.similarity("britishairways", 'americanair')
```

```
0.72780585
```

Figure 12: Both companies are airlines and have a high similarity.

```
w2v_model.wv.similarity("chipotletweets", 'burrito')
```

```
0.67098397
```

Figure 13: Both items are part of the Mexican cuisine and are often consumed together.

Finally we tried querying the model for analogies. We passed two related words and a third word to the function with the expected output being a word such that the relation between the passed related words, and the third word and this word is similar. The following results were obtained.

```
w2v_model.wv.most_similar(positive=["apple", "amazonhelp"], negative=["amazon"], topn=3)
```

```
[('applesupport', 0.7136973142623901),
 ('115858', 0.6176860332489014),
 ('115858_applesupport', 0.6052395105361938)]
```

Figure 14: “amazon” is to “amazonhelp” as “apple” is to “applesupport”.

```
w2v_model.wv.most_similar(positive=["chipotle", "apple"], negative=["iphone"], topn=5)
```

```
[('chipotletweets', 0.6921877861022949),
 ('mcdonalds', 0.5467450618743896),
 ('jackbox', 0.5261733531951904),
 ('arbys', 0.5249221324920654),
 ('chipotletweets_chipotle', 0.5180568695068359)]
```

Figure 15: “iphone” is to “apple” as “chipotle” is to “chipotletweets”.

```
w2v_model.wv.most_similar(positive=["xbox", "song"], negative=["spotify"], topn=3)
```

```
[('game', 0.6415499448776245),
 ('xbox_one', 0.5659031867980957),
 ('xboxsupport', 0.5278393626213074)]
```

Figure 16: “spotify” is to a “song” as “xbox” is to a “game”.

3.5.3 Topic Modelling: LDA, Phrase based and Bi-term models

Using the LDA and biterm topic models, we were able to identify over 9 topics and the associated keywords and their probabilities.

```
[(0,
  '0.205*"number" + 0.124*"back" + 0.065*"hello" + 0.062*"give" + '
  '0.057*"follow" + 0.047*"pay" + 0.044*"internet" + 0.039*"year" + '
  '0.033*"home" + 0.029*"tweet"'),
 (1,
  '0.223*"store" + 0.113*"im" + 0.084*"come" + 0.063*"hey" + 0.055*"even" + '
  '0.054*"find" + 0.042*"already" + 0.037*"week" + 0.032*"ill" + 0.030*"open"'),
 (2,
  '0.202*"account" + 0.202*"send" + 0.160*"use" + 0.088*"link" + '
  '0.031*"access" + 0.028*"sprintcare" + 0.027*"sign" + 0.022*"someone" + '
  '0.018*"extra" + 0.014*"food"'),
 (3,
  '0.142*"service" + 0.130*"phone" + 0.093*"customer" + 0.076*"happy" + '
  '0.060*"askspectrum" + 0.057*"apology" + 0.054*"great" + 0.042*"name" + '
  '0.041*"bad" + 0.040*"apologize"'),
 (4,
  '0.207*"location" + 0.000*"team" + 0.000*"leadership" + 0.000*"visit" + '
  '0.000*"feedback" + 0.000*"always" + 0.000*"available" + 0.000*"let" + '
  '0.000*"kindly" + 0.000*"close"'),
 (5,
  '0.247*"issue" + 0.130*"assist" + 0.073*"line" + 0.065*"experience" + '
  '0.058*"long" + 0.044*"people" + 0.033*"click" + 0.024*"kind" + 0.021*"top" '
  '+ 0.018*"profile"'),
 (6,
  '0.237*"thank" + 0.135*"youre" + 0.110*"information" + 0.061*"date" + '
  '0.054*"local" + 0.049*"resolve" + 0.037*"ever" + 0.021*"correct" + '
  '0.018*"quickly" + 0.018*"consider"'),
 (7,
  '0.177*"order" + 0.108*"contact" + 0.085*"dont" + 0.072*"update" + '
  '0.068*"info" + 0.061*"keep" + 0.057*"support" + 0.044*"free" + 0.044*"chat" '
  '+ 0.037*"never"'),
 (8,
  '0.129*"call" + 0.125*"still" + 0.080*"today" + 0.075*"app" + 0.064*"sure" + '
  '0.055*"card" + 0.043*"charge" + 0.042*"fix" + 0.035*"something" + '
  '0.030*"help"'),
```

Figure 17: Topics and their Keywords along with their Importance

Next we show some results obtained using phrase based modelling.

```
['direct', 'message', 'email', 'address', 'follow']
['direct_message', 'email', 'address', 'follow']
['spotifycares', 'use', 'macbook', 'pro', 'el', 'capitan', 'late', 'spotify', 'update', 'iphone', 'late', 'softwarespotify',
'update']
['spotifycares', 'use', 'macbook_pro', 'el', 'capitan', 'late', 'spotify', 'update', 'iphone', 'late', 'softwarespotify', 'up
date']
```

Figure 18: Example of Phrase detection (1)

```
['thank', 'reach', 'flight', 'attendant', 'assistance', 'tmt']
['thank', 'reach', 'flight_attendant', 'assistance', 'tmt']
['dear', 'cute', 'flight', 'attendant', 'delta', 'boardjust', 'sayin']
['dear', 'cute', 'flight_attendant', 'delta', 'boardjust', 'sayin']
```

Figure 19: Example of Phrase detection (2)

```
['sprintcare', 'sent', 'several', 'private', 'message', 'one', 'respond', 'usual']
['sprintcare', 'sent', 'several', 'private_message', 'one', 'respond', 'usual']
['send', 'private', 'message', 'assist', 'click', 'message', 'top', 'profile']
['send', 'private_message', 'assist', 'click', 'message', 'top', 'profile']
['send', 'private', 'message', 'gain', 'detail', 'account']
['send', 'private_message', 'gain', 'detail', 'account']
```

Figure 20: Example of Phrase detection (3)

Analyzing the results of this topic modeling can provide great insights for industrial use as it can help the support staff to label items quickly and thus increase the efficiency of dealing with such queries.

But understanding just these numbers and words in this format can prove to be challenging and thus, we provide different ways of analyzing these results below.

	Tweet_No	Dominant_Topic	Topic_Perc_Contrib	Keywords	Text
0	0	10.0	0.1506	time, would, see, trouble, understand, arrive, nice, amazonhelp, let, check	[understand, would, assist, would, need, private, secure, link, assist]
1	1	2.0	0.1098	account, send, use, link, access, sprintcare, sign, someone, extra, food	[sprintcare, propose]
2	2	9.0	0.2913	get, one, try, message, target, also, sent, thats, respond, definitely	[sprintcare, sent, several, private, message, one, respond, usual]
3	3	5.0	0.1871	issue, assist, line, experience, long, people, click, kind, top, profile	[send, private, message, assist, click, message, top, profile]
4	4	2.0	0.1098	account, send, use, link, access, sprintcare, sign, someone, extra, food	[sprintcare]
5	5	9.0	0.1378	get, one, try, message, target, also, sent, thats, respond, definitely	[send, private, message, gain, detail, account]
6	6	3.0	0.2074	service, phone, customer, happy, askspectrum, apology, great, name, bad, apologize	[sprintcare, bad, customer, service]
7	7	17.0	0.2298	look, httpstco, hear, able, anything, stop, shoot, rule, kc, amazonhelp	[sadden, hear, shoot, look, kc]
8	8	14.0	0.2449	make, detail, change, way, thing, point, na, hopefully, whole, family	[sprintcare, gon, na, magically, change, connectivity, whole, family]
9	9	19.0	0.1460	day, wed, since, month, much, every, forward, direct, per, picture	[understand, concern, wed, send, direct, message, assist, aa]
10	10	2.0	0.1413	account, send, use, link, access, sprintcare, sign, someone, extra, food	[sprintcare, since, sign, yousince, day]
11	11	5.0	0.1536	issue, assist, line, experience, long, people, click, kind, top, profile	[wed, definitely, work, long, experience, issue, aa]
12	12	8.0	0.2985	call, still, today, app, sure, card, charge, fix, something, help	[lie, great, connection, bar, lte, still, load, something, smh]
13	13	2.0	0.2474	account, send, use, link, access, sprintcare, sign, someone, extra, food	[send, private, message, send, link, access, account, fr]
14	14	7.0	0.1836	order, contact, dont, update, info, keep, support, free, chat, never	[whenever, contact, customer, support, tell, shortcode, enable, account, never, year, ive, try, ...]

Figure 21: Keywords for the Identified Topic of Selected Tweets

	Topic_Num	Topic_Perc_Contrib	Keywords	Representative Text
0	0.0	0.3080	number, back, hello, give, follow, pay, internet, year, home, tweet	[verizonsupport, friend, without, internet, need, play, videogames, together, skill, diminish, e...
1	1.0	0.3164	store, im, come, hey, even, find, already, week, ill, open	[hey, chipotletweets, wan, na, come, mammoth, ill, least, eat, week, promise]
2	2.0	0.2474	account, send, use, link, access, sprintcare, sign, someone, extra, food	[amazonhelp, look, signinform, say, dont, account, cant, access, account, skip, sign, use]
3	3.0	0.3451	service, phone, customer, happy, askspectrum, apology, great, name, bad, apologize	[yo, askspectrum, customer, service, rep, super, nice, imma, start, trippin, service, go]
4	5.0	0.2267	issue, assist, line, experience, long, people, click, kind, top, profile	[chipotletweets, excellent, service, tonight, plenty, people, line, go, fast, everyone, kind]
5	6.0	0.2414	thank, youre, information, date, local, resolve, ever, correct, quickly, consider	[chipotletweets, thank, chipotletweets, resolve, issue, quickly, best, fanforlife]
6	7.0	0.3119	order, contact, dont, update, info, keep, support, free, chat, never	[info, share, moment, feel, free, keep, eye, blog, news, update]
7	8.0	0.3048	call, still, today, app, sure, card, charge, fix, something, help	[frustrate, chipotletweets, order, dinner, saturday, use, app, order, wrong, charge, credit, car...
8	9.0	0.2913	get, one, try, message, target, also, sent, thats, respond, definitely	[sprintcare, sent, several, private, message, one, respond, usual]
9	10.0	0.1902	time, would, see, trouble, understand, arrive, nice, amazonhelp, let, check	[see, uptime, hr, minute, see, downstream, channel, range, nearly, spec, well, see, report, upst...
10	12.0	0.2500	work, address, provide, receive, ive, concern, wrong, version, didnt, mean	[mean, boorito, basically, adult, version, halloween, becky]
11	13.0	0.2493	go, area, part, office, particular, department, amazonhelp, asktarget, let, check	[department, part, corporate, office, youre, particular, area, go, format, unawa, httpstcop, xcm...
12	14.0	0.2449	make, detail, change, way, thing, point, na, hopefully, whole, family	[sprintcare, gon, na, magically, change, connectivity, whole, family]
13	15.0	0.2823	know, connect, wifi, minute, nothing, refer, cut, router, wireless, verizonsupport	[verizonsupport, know, router, downstairs, wifi, nothing, connect, ethernet]
14	16.0	0.2342	well, email, need, want, could, share, via, id, do, form	[well, yet, whats, customer, support, email, communicate, via, email]

Figure 22: Representative Tweets for the Keywords of the Identified Topics

Some of the usually employed metrics to evaluate topic models are the perplexity and coherence score. Perplexity is a measure of how well the model performs on held out data and basically shows how

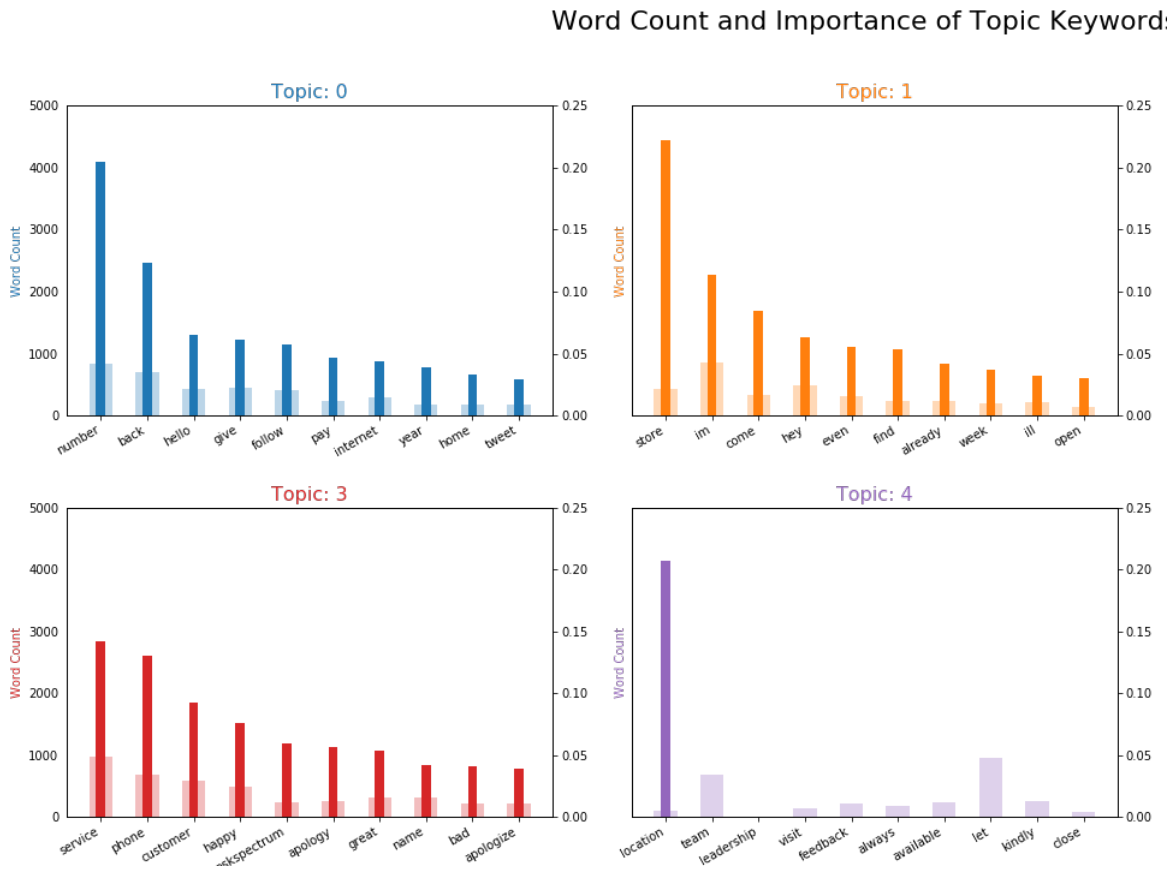

Figure 23: Word Count and Importance of Keywords for 4 of the Identified Topics

Figure 24: Word Clouds for 4 of the topics

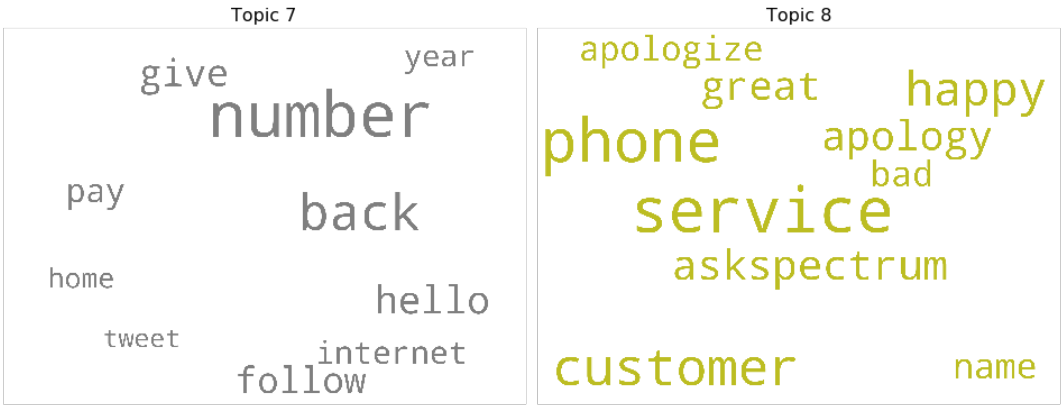

Figure 25: Word Clouds for Topic 7 and 8

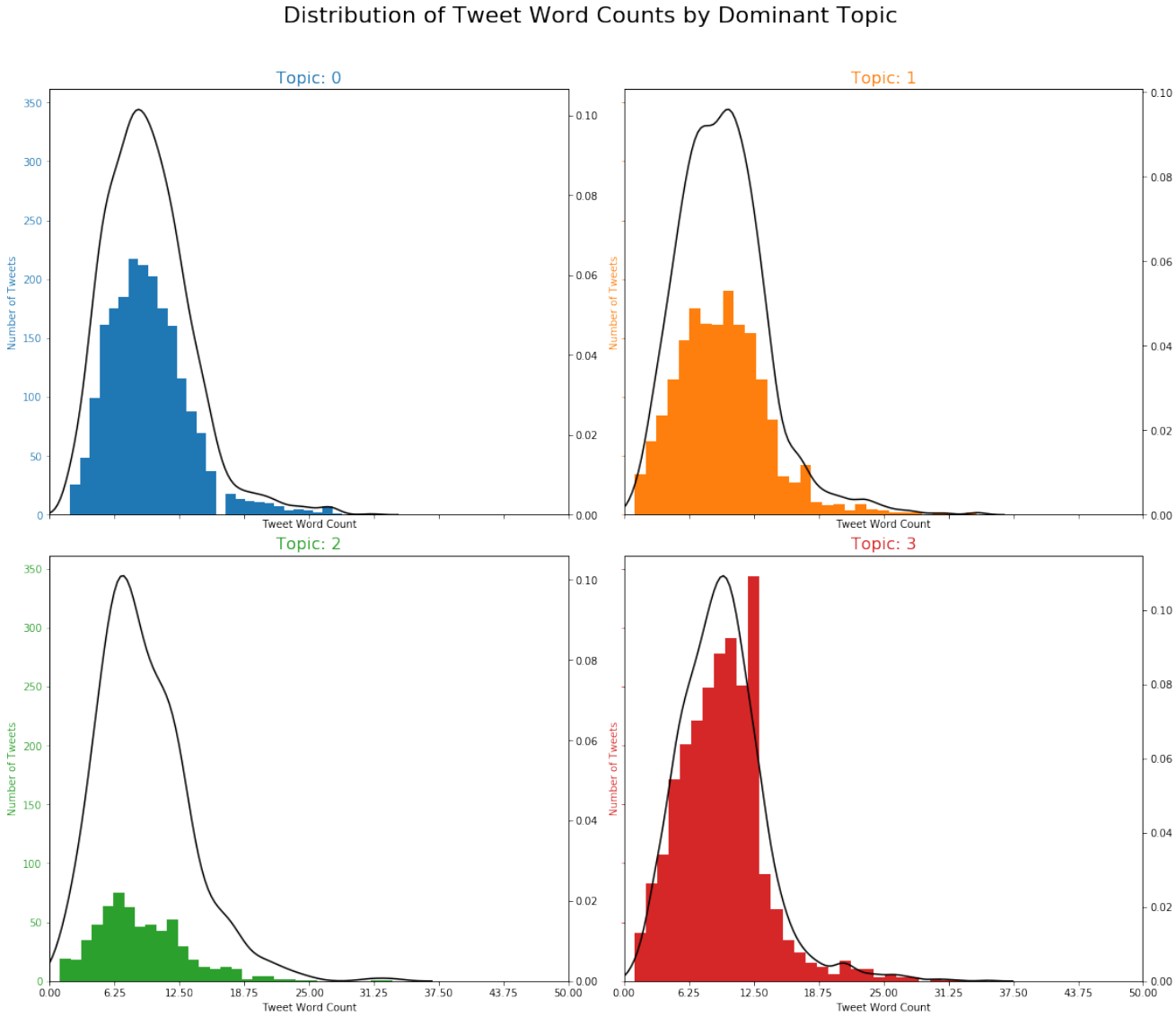

Figure 26: Distribution of Tweet Word Counts by Dominant Topic

'surprised' the model is on seeing new information. Thus the lower the perplexity the better and the perplexity score for our model turned out to be `-20.9388`. The coherence score signifies the semantic

similiarity between the identified keywords for the topics These measurements help distinguish between topics that are semantically interpretable topics and topics that are artifacts of statistical inference. The coherence score obtained by our model is `0.447` which is still fairly large for a corpus of short tweets.

3.5.4 Sentiment Analysis

Here we present some of the results obtained from sentiment analysis. We observed that the sentiment of SouthwestAir's customers is the highest, while the customers of AppleSupport have the lowest sentiment.

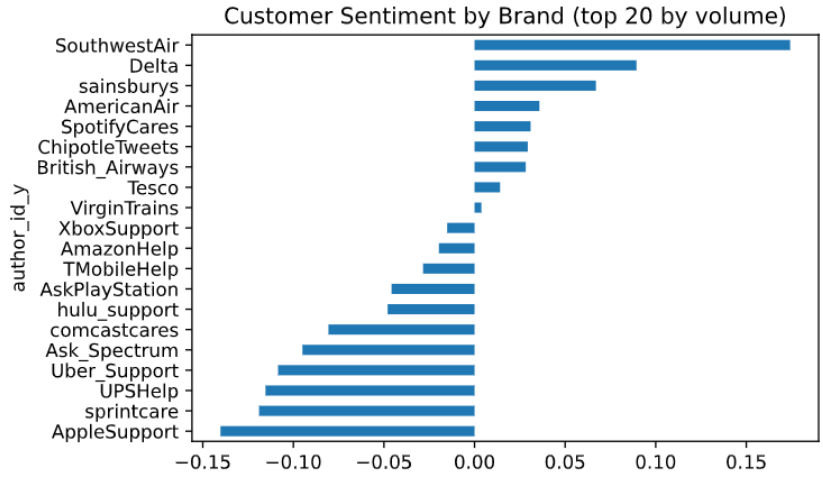

Figure 27: The graph shows the average sentiments of the customers for the top 20 brands by volume.

Finally we show the customers of sprintcare that lie below the `avg_threshold=-0.7` for average sentiment score and `min_threshold=-0.8` for the minimum sentiment score.

	author_id	sentiment
0	157821	-0.9487
1	157432	-0.8722
2	128518	-0.8402
3	139643	-0.8264
4	136463	-0.8126
5	147658	-0.8122
6	126203	-0.8020
7	154080	-0.8020
8	167686	-0.8011
9	160374	-0.7734
10	134203	-0.7698
11	163442	-0.7562
12	152916	-0.7562
13	157819	-0.7545
14	175888	-0.7381
15	166623	-0.7303
16	130362	-0.7269
17	167673	-0.7243
18	136459	-0.7166

Figure 28: Average sentiment score

	author_id	sentiment
0	157821	-0.9487
1	136460	-0.9301
2	150834	-0.9221
3	160379	-0.9211
4	173005	-0.8910
5	123402	-0.8778
6	157432	-0.8722
7	148370	-0.8689
8	166617	-0.8553
9	160370	-0.8514
10	150245	-0.8481
11	128518	-0.8402
12	136459	-0.8338
13	151439	-0.8316
14	139643	-0.8264
15	141500	-0.8246
16	167668	-0.8126
17	136463	-0.8126
18	147658	-0.8122
19	126203	-0.8020
20	166622	-0.8020
21	154080	-0.8020
22	160397	-0.8016
23	167686	-0.8011

Figure 29: Minimum sentiment score

4 Conclusions and future work

Here we see that with such a large corpus, there can be innumerable ways of dissecting the data, trying to understand it and exploit it in different manners. In our attempt to do it most efficiently we tried a number of these approaches and feel that for the purposes of topic modeling and identifying which topic a tweet could be related to LDA coupled with phrase based detection gives us the dual benefit of identifying keywords while retaining their original context.

For a bird's eye view of the different relationships, similarities and analogies in the data set we found the word2vec model trained by us to be incredibly powerful as it could present similarities between different entities and express human understandable relationships in the form of the similarity scores, with a surprisingly high accuracy.

For future work we believe that a few directions worth exploring in terms of the technique used would be the Biterm model and how its parameters relate to the distributions of the biterms over the entire corpus, whether assuming these Dirichlet distribution over individual or concatenated tweets may perform better or whether it can be used with different unsupervised techniques such as a Gaussian Mixture Model. More focus is needed not only on its conceptualization but also in the implementation as we could trying to implement it along with the libraries present in Python to use its entire functionality which might have been subdued by using it in R.

Also, in terms of the different problems that can be explored, one could look at the time at the which these tweets are responded to as compared to their incoming time and then the frequency of tweets for that particular request so as to get a sense if and when requests go 'stale' i.e. a time after which either the query is resolved by the customer themselves or more dangerously for the company, losing of brand loyalty and their consumer base.

One can also try to identify new problems and potential pain points for customers by analyzing the

company specific tweets and what usually leads to problems for customers from their present products. This could help them design new lines of products by incorporating this sort of feedback or eliminating these problems in further software updates/version of their products.

5 References

[1] Lovins, Julie Beth. "Development of a Stemming Algorithm." Mechanical Translation and Computational Linguistics 11 (June 1968). http://chuvyr.ru/MT-1968-Lovins.pdf.

[2] NeelShah18. Emot. Python, 2020. https://github.com/NeelShah18/emot/blob/master/emot/emo_unicode.py.

[3] slowkow. Remove-Emoji. Python, 2018. https://gist.github.com/slowkow/7a7f61f495e3dbb7e3d767f97bd7304b.

[4] rishabhverma17. SMS/Message_Slang_Translator. Python, 2018. https://github.com/NeelShah18/emot/blob/master/emot/emo_unicode.py.

[5] Manning, Christopher D.; Raghavan, Prabhakar; Schütze, Hinrich. "Spelling Correction." In Introduction to Information Retrieval. Cambridge University Press, 2008. https://nlp.stanford.edu/IR-book/html/htmledition/spelling-correction-1.html.

[6] Manning, Christopher D.; Raghavan, Prabhakar; Schütze, Hinrich. "Stemming and Lemmatization." In Introduction to Information Retrieval. Cambridge University Press, 2008. https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html.

[7] Ali, Zafar. "A Simple Word2vec Tutorial," January 6, 2019. https://medium.com/@zafaralibagh6/a-simple-word2vec-tutorial-61e64e38a6a1.

[8] McCormink, Chris. "Word2Vec Tutorial Part 2 - Negative Sampling," January 11, 2017. http://mccormickml.com/2017/01/11/word2vec-tutorial-part-2-negative-sampling/.

[9] MonkeyLearn. "Sentiment Analysis," 2020. https://monkeylearn.com/sentiment-analysis/.

[10] Beri, Aditya. "SENTIMENT ANALYSIS USING VADER," May 27, 2020. https://towardsdatascience.com/sentimental-analysis-using-vader-a3415fef7664.

[11] AFAF5. "Sentiment Analysis: VADER or TextBlob?," January 6, 2021. https://www.analyticsvidhya.com/blog/2021/01/sentiment-analysis-vader-or-textblob/.

[12] Yan, Qiushi. "Latent Dirichlet Allocation." In Notes for "Text Mining with R: A Tidy Approach," n.d. https://bookdown.org/Maxine/tidy-text-mining/latent-dirichlet-allocation.html.

[13] Blei, David M.; Ng, Andrew Y.; Jordan, Michael I. "Latent Dirichlet Allocation." Journal of Machine Learning Research 3 (March 1, 2003): 993–1022.

[14] Doll, Tyler. "LDA Topic Modeling: An Explanation," June 25, 2018. https://towardsdatascience.com/lda-topic-modeling-an-explanation-e184c90aadcd.