Text Analytics Final Project

Creating Filters on Reddit Comments

Nimeelitha Akkiraju | Keerthi Bojja | Pranidhi Prabhat | Michael Wang | Mona Yao

Dec 2, 2020

Project Report

Table of Contents

Problem Statement

The data set comprises of Reddit's comments for different brands of cars. For a new user visiting the pages, the content is overwhelming at the first sight. If he has a priority list for certain features in the automobile, he might not be able to reach the most valuable comments given the number of comments.

Solution

To create a better user experience for these users, we planned to create filters of the most important topics under play in each of the brands comments, so that the user can filter out the comments related to these important topics.

Related Work

Amazon has the feature of filters on its reviews. So, for example, if a user is looking for a certain quality in the product such as 'look' , he can just click on the 'look' or 'appearance' filter and the comments that talk about the look or appearance of the product will appear, providing the user more valuable comments to read. We are drawing example from this case and are trying to create the filters for the Reddit comments.

Approach & Methodology

The first step would be to look for the keywords in the text corpus and use them to create topics. As there are various methods to do the same, determine the best approach to get the most relevant set of keywords. Hence, explore different methods to finalize which methodology works the best and produces the most relevant filters.

The methodology involved was to first create a corpus of stop words that was relevant to the dataset. And then perform the following methods on the dataset :

Topic Modeling with Latent Dirichlet Allocation

Word2Vec – Create embeddings and use KMeans algorithm on the embeddings to create clusters

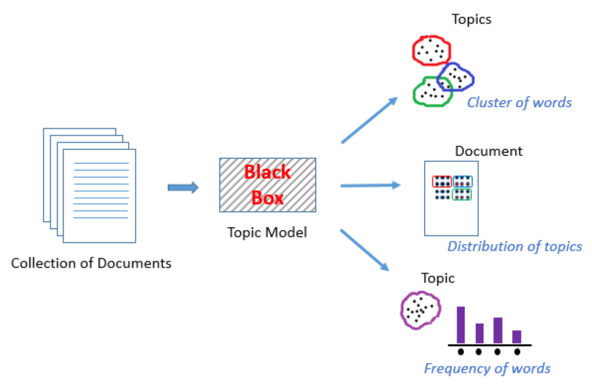
BERT – Use BERT methodology to create topic modeling using the UMAP and DBSCAN to establish the number of topics and then extract the number of topics.

Topic Modeling

What is a topic model?

A Topic Model can be defined as an unsupervised technique to discover topics across various text documents. These topics are abstract in nature, i.e., words which are related to each other form a topic. Similarly, there can be multiple topics in an individual document. For the time being, let's understand a topic model as a black box, as illustrated in the below figure:

This black box (topic model) forms clusters of similar and related words which are called topics. These topics have a certain distribution in a document, and every topic is defined by the proportion of different words it contains.



Topic modeling helps in exploring large amounts of text data, finding clusters of words, similarity between documents, and discovering abstract topics. To provide a solution to the problem we identified, we have tried a couple of approaches to implement Topic Modeling on the Reddit Data Comments

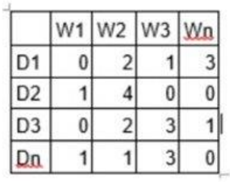
Topic Modeling with LDA

Latent Dirichlet Allocation is the most popular topic modeling technique. It assumes documents are produced from a mixture of topics. Those topics then generate words based on their probability distribution.

Documents ------> Topics ------> Words

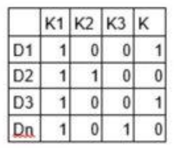
Given a dataset of documents, LDA backtracks and tries to figure out what topics would create those documents in the first place.

LDA is a matrix factorization technique. In vector space, any corpus (collection of documents) can be represented as a document-word matrix

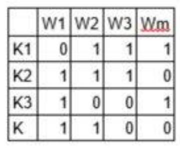


	W1	W2	W3	Wn
D1	0	2	1	3
D2	1	4	0	0
D3	0	2	3	1
Dn	1	1	3	0

LDA converts this Document-Term Matrix into two lower dimensional matrices - **Document – topic** and **Topic – word matrices.**



	K1	K2	K3	K
D1	1	0	0	1
D2	1	1	0	0
D3	1	0	0	1
Dn	1	0	1	0



	W1	W2	W3	Wm
K1	0	1	1	1
K2	1	1	1	0
K3	1	0	0	1
K	1	1	0	0

The main aim of LDA is to use sampling techniques in order to improve these matrices. It Iterates through each word “w” for each document “d” and tries to adjust the current topic – word assignment with a new assignment. After a number of iterations, a steady state is achieved where the document topic and topic term distributions are fairly good. This is the convergence point of LDA

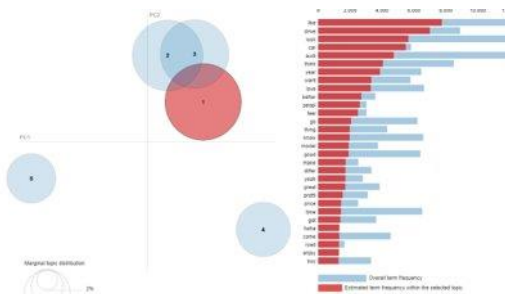
Text Preprocessing:

- Remove punctuation
- Lower casing
    - Convert a word to lower case
- Stop words removal
    - English
    - Adding words of your choice
- Stemming
    - Transform a word to its root form
    - Chops off the derivational affixes
- Lemmatization
    - Reduces the words to a word existing in the language

    - Uses morphological analysis
- Tokenization
    - Paragraph, sentence, words

Results

Below is topic modeling result for Audi brand



There is an overlap in topics and repeat in words among various topics. This can be overcome by other methods below.

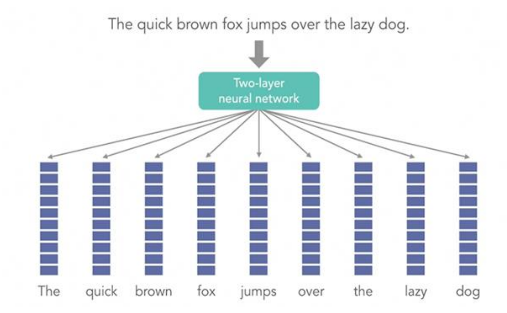
Topic Modeling with Word2Vec

Text Preprocessing:

Same steps have been followed for LDA & Word2Vec to keep our data cleaning consistent. We then passed the cleaned & tokenized text data into the Word2Vec model

Text Representation:

Word2Vec is a statistical method for efficiently learning a standalone word embedding from a text corpus. In simple terms, it is a two-layer neural network "that accepts a text corpus as an input, "and it returns a set of vectors", as word embeddings



It was developed by Tomas Mikolov, et al. at Google in 2013 as a response to make the neural-network-based training of the embedding more efficient and since then has become the de facto standard for developing pre-trained word embedding.

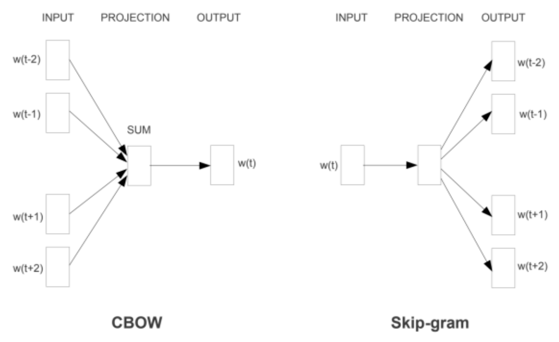
Additionally, the work involved analysis of the learned vectors and the exploration of vector math on the representations of words. For example, that subtracting the "man-ness" from "King" and adding "women-ness" results in the word "Queen", capturing the analogy "king is to queen as man is to woman".

Two different learning models were introduced that can be used as part of the word2vec approach to learn the word embedding; they are:

- Continuous Bag-of-Words, or CBOW model.
- Continuous Skip-Gram Model.

The CBOW model learns the embedding by predicting the current word based on its context.

The continuous skip-gram model learns by predicting the surrounding words given a current word.



Both models are focused on learning about words given their local usage context, where the context is defined by a window of neighboring words. This window is a configurable parameter of the model.

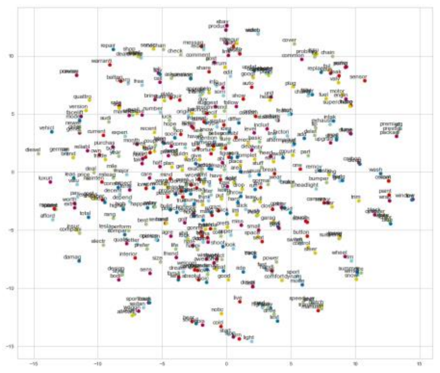
The size of the sliding window has a strong effect on the resulting vector similarities. Large windows tend to produce more topical similarities, while smaller windows tend to produce more functional and syntactic similarities.

The key benefit of the approach is that high-quality word embeddings can be learned efficiently (low space and time complexity), allowing larger embeddings to be learned (more dimensions) from much larger corpora of text (billions of words).

We have used CBOW to train our model. And below are the results:

Visualize Word2Vec using TSNE

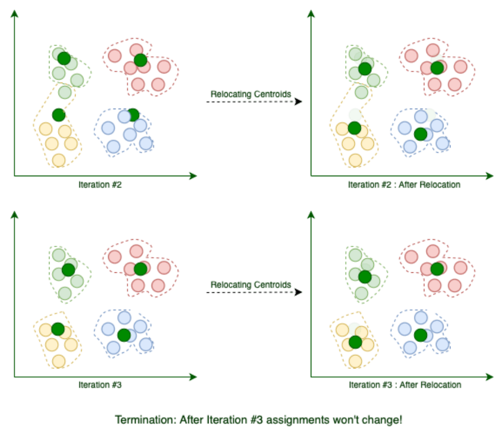
t-SNE is a tool to visualize high-dimensional data. It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. t-SNE has a cost function that is not convex, i.e. with different initializations we can get different results.



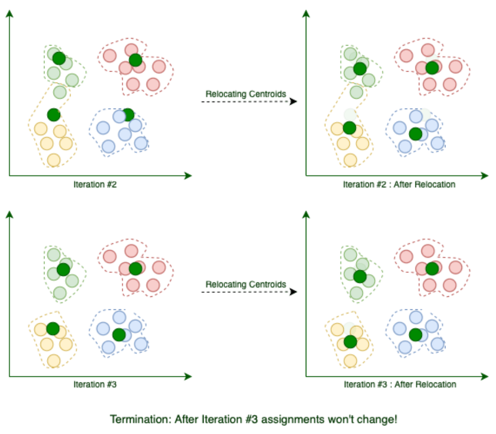
The above figure is a visualization of word embeddings of our wor2vec model. The words seem to be sparse and overlapping each other. But these word vectors capture the meanings to the extent that you can construct word analogies, like "king" is to "man" as "queen" is to "woman".

Now, after the words have been converted to list of vectors, we then passed those vectors as an input to K-Means algorithm to cluster the documents.

K-Means Clustering:


Relocating Centroids
Iteration #2
Iteration #2 : After Relocation
Relocating Centroids
Iteration #3
Iteration #3 : After Relocation
Termination: After Iteration #3 assignments won't change!

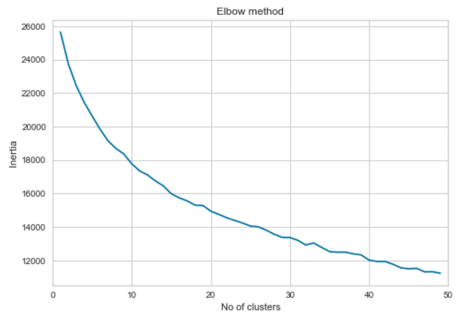
The K-means clustering technique basically starts with k arbitrary chosen means which are called the centroids, then it allocates each vector to the cluster with the closest mean. It then recalculates the means of each cluster as the centroid of the vectors in the cluster. This process repeats until the clusters are stabilized around it’s closest centroid.


Relocating Centroids
Iteration #2
Iteration #2 : After Relocation
Relocating Centroids
Iteration #3
Iteration #3 : After Relocation
Termination: After Iteration #3 assignments won't change!

The output of K-Means clustering algorithm shown in the below plot. We chose the optimal number of clusters to be 35 based on the elbow method, and also considered the Silhouette Score to arrive at this number

Elbow Method:

In the below graph, the cluster value where there is a decrease in inertia value becomes constant can be chosen as the right cluster value for our data.



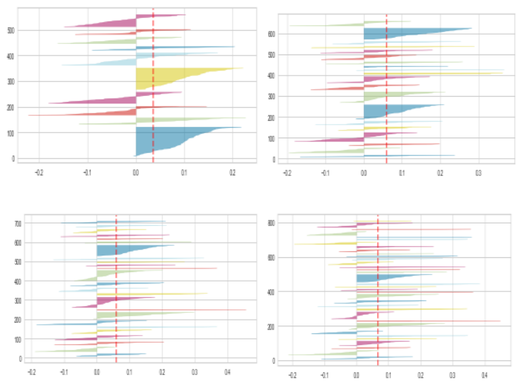
Here, our optimal cluster value seems to be between 30 to 40 clusters. To reduce the range further, we have considered another method called Silhouette Score.

Silhouette Score:

This Score helps evaluate the quality of clustering done using the K-means algorithm.

Silhouette Score	Meaning
1	Clusters are dense and nicely separated
0	Clusters are overlapping
<0	Samples might have got assigned to the wrong clusters

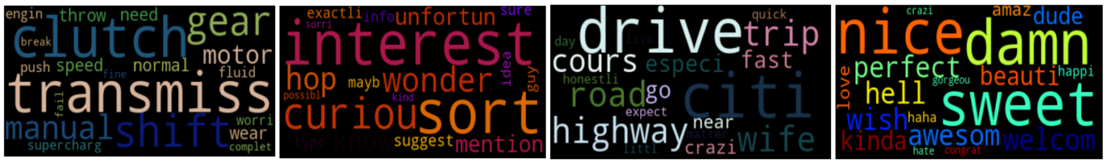
The below graph here are Silhouette plots for 10, 20, 25 & 35 clusters and the score above 0.05 for each of them, 35 clusters having the highest as 0.068.



As we can see in the above figures, 10 & 20 clusters have a lot of fluctuations in cluster sizes, whereas 25 & 35 have lesser fluctuations comparatively. Also, 35 clusters is optimal compared to 25 because the fluctuation in size of clusters is similar. The thickness of each cluster in the silhouette plot is also one of the deciding factors to select the optimal number of clusters

Word Cloud Visualization of Clusters:

Finally, below are the word clouds for couple of clusters with top 20 words that have been generated by K-Means. As we can see, each cluster has different set of words associated to it and are neatly clustered.



Topic Modeling with Transformers & BERT

This section will briefly explain what “transformers” is and how it helps with analyzing text data. In addition, I will illustrate the differences between BERT and other models such as Word2Vec.

Transformers is a library that provides general-purpose architectures such as BERT, GPT-2, DistilBERT for Natural Language Understanding and Natural Language Generation.

BERT is one of the models that are available in the transformer library. BERT stands for Bidirectional Encoder Representations from Transformers) and as its name suggests, it read the text sequence all at once from both directions instead of just one, which allows it to understand the meaning of a specific word based on its given context. An example would be, suppose we have two lines of text each contains the word ‘bank’: “Kids were playing alongside a riverbank” and “The man went into the bank to deposit a large check”. In word2Vec, the model will not differentiate the meaning of these two “banks”, while BERT can do so because it was trained with masks on certain words and it was designed to distinguish different meanings of the same words based on the context.

BERT is powerful because it was pre-trained on BookCorpus, which covers text data from 11,038 unpublished books and the entire English Wikipedia. This creates billions of words and text for BERT to learn the contextual meanings of a specific word.

For our project, we leveraged a package named ‘Sentence-Transformer’ and used BERT to perform sentence embedding. Our objective was to convert raw text data into vectors in high dimensions and once we have the vectors available, we can cluster them into different topics using UMAP. Because we are working with 6 distinct brands of cars, we are interested in examining the topic differences between these brands. Clustering text sequences will help us achieve our goal and give us a general idea of what people talk the most under each Subreddit.

Data

In this part of the project, since we couldn't get access to the GPU, we have to run the code locally, we only used 10,000 data points in each file. What we did is we randomly selected 10,000 text body and used them to come up with the topics.

Workflow

The whole workflow can be divided into three steps, word embedding, clustering and topic creation.

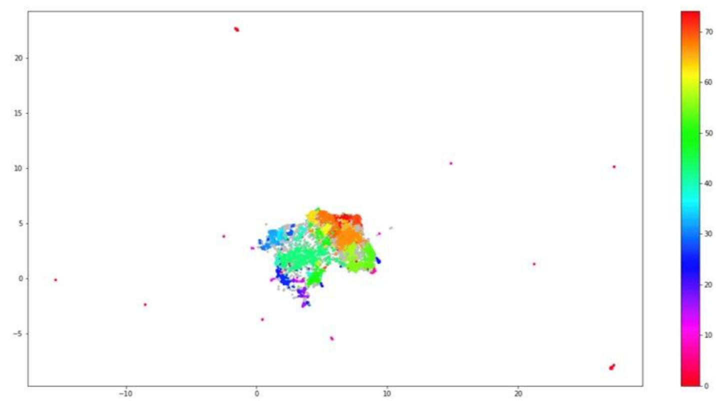
In the word embedding part, we broke down the text body into sentences, and fed these sentences to BERT. We specifically selected sentence transformer, which is a pre-trained model in BERT. We chose this model because it has shown to be very high quality and typically work well for document-level embeddings.

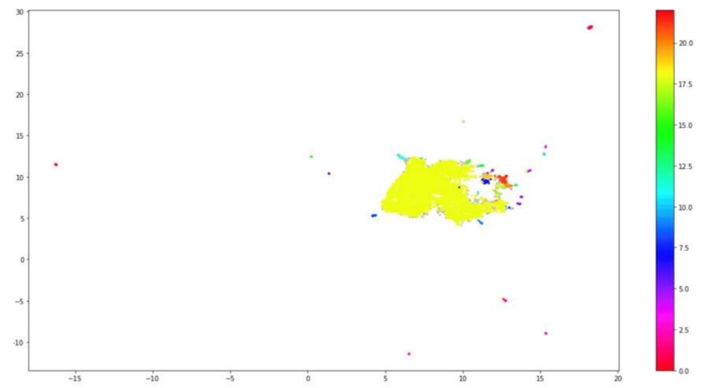
After we have the word embeddings, we can try to cluster them out. However, since a lot of clustering model deal with high dimensional data poorly, we use UMAP to reduce the dimensionality first. After we reduced the dimensionality of the data, we used HDBSCAN to do the clustering. HDBSCAN is a density-based algorithm that works quite well with UMAP, since UMAP maintains a lot of local structure even in lower-dimensional space. In addition, HDBSCAN doesn't force data points to clusters as it considers them outliers.

The last step is topic creation. In this step, we used class-based TF-IDF. The difference between this method and normal TFIDF is that, in normal TFIDF, we compare the importance of word in different documents. However, in class-based TF-IDF, we treat all documents in one group and apply TF-IDF on it. Then, the result will give us the word importance in one topic.

After we got the word importance, we can use the top words with the highest importance to represent the topic, and finally we can get our topics.

Below are some representative illustrations of the cluster plots, we can see some brand topics spread relatively evenly like what showed in the upper plot, which means the topic clusters contains similar number of words. But for others like the plot below shows, the largest topic contains a large amount of words, and the rest of topics share the rest words.





Analyzing Results

The result from UMAP contains a lot of noise and it was a challenge at first to find exact topics these words refer to. However, some patterns were observed when we separate the 6 brands into 2 groups. “Mercedes”, “BMW” and “Audi” as our 1st group and “Leaf”, “BoltEV” and “Tesla” as our 2nd group. The 1st group consists of three traditional car makers, although some of them are starting to build some electric vehicles, the focus is still on gasoline cars now. The 2nd group differentiates itself from the first group by focusing solely on electric vehicles.

When we carefully examine the most mentioned keywords or topics within the first group, we noticed words like “AMG”, “Stage”, “Tuning”, “bhp” (base horsepower). All these words refer to the performance line of these car brands. This makes perfect sense because people who browse these Reddit are usually not customers who bought standard models from these brands, a vast majority of them are either interested in performance tuning or “modding” their beloved vehicles. The point is that a Mercedes C300 owner will be less likely to browse Reddit frequently comparing to someone who bought a Mercedes AMG C63 (a sportier version of the vehicle). Another interesting aspect our analysis was able to capture was “Nardo Gray” and “Beam” from Audi. “Nardo gray” was a color that is unique to Audi and high-quality LED beam light is also one of the features of the brand. It shows that a vast majority of these posts discuss about these unique features of a specific brand.

If we look at the electric car makers. The keywords and topics that were mentioned shifted quite a lot comparing to traditional companies. One thing we immediately noticed was ‘http’ and ‘YouTube’ were mentioned more frequently in these electric vehicle makers, which suggest that electric vehicle posts tend to have more videos or images links. Another interesting observation is that the founder name or the parent company name were mentioned very frequently in these electric vehicle posts. We see ‘Elon Musk’ and ‘Chevrolet’ frequently appear in the posts, which suggest a how big of an impact the company owner or the parent company have on the child company in these new energy vehicles.

In Conclusion, BERT and UMAP was able to provide us with some insights into different topics being discussed under these 6 Subreddit. Although many topics and words are a bit challenging to understand or to summarize into a specific topic. Looking at the keywords allow us to see the difference between the traditional car makers and new energy vehicle companies.

Model Comparison for Audi Car Data:

LDA:

	C1	**C2**	**C3**	**C4**	**C5**
W1	look	engine	audi	like	thank
W2	delet	issu	http	drive	audi
W3	wheel	time	post	look	work
W4	like	tune	model	car	know
W5	black	chang	reddit	audi	tire

Word2Vec:

	C1	**C2**	**C3**	**C4**	**C5**
W1	entir	silver	unfortun	decent	fail
W2	basic	grey	sort	pretti	caus
W3	add	white	exactli	aren	fix
W4	assum	black	mention	near	problem
W5	curiou	color	assum	probabl	issu

BERT:

	C1	**C2**	**C3**	**C4**	**C5**
W1	spot	based	space	seats	earlier
W2	shokan	lostredditors	disassembled	leaving	closed
W3	cent	stage	tapes	visits	cassette
W4	nano	curb	wagons	watch	nardo
W5	thanks	3a	mpre	rover	thank

Conclusion

When we compare the three methods, we think word2vec generated topics that makes the most sense, the words being grouped in the topic are easily understandable and there are less noise comparing to the other results and thus is a winner of the three.

Future Work

There are more things we can explore in the future:

1. Setting up the proper GPU environment and running the complete dataset instead of sampling 10,000 data from each brand. This will help us get a better understanding of the dataset.
2. We can also explore many other functionalities associated with transformers library since it is capable of doing many more types of tasks with fine tuning a downstream layer of the network.
3. While working with tSNE visualization for word2Vec, it is highly recommended to use another dimensionality reduction method (e.g. PCA for dense data or TruncatedSVD for sparse data) to reduce the number of dimensions to a reasonable amount (e.g. 50) if the number of features is very high. This will suppress some noise and speed up the computation of pairwise distances between samples.