Indian Institute of Technology Madras
Chennai, Tamil Nadu, India

Natural Language Processing for Review Clustering

MS4610: Introduction to Data Analytics

Advisor: Prof. Nandan Sudarsanam
Team: Group 12

December 2021

Contents

Introduction

Natural Language Processing is application of data-driven techniques to interpret natural language and speech. Text Analysis in specific uses Machine Learning to extract information from text data. The following pages present various algorithms, unsupervised techniques and approaches to perform hierarchical clustering of text data.This study aims to serve as a guide to clustering textual data of any kind using the listed algorithms.

The text data consists of various product reviews collected from Amazon, Walmart, CVS and Influenster. The report starts with an introduction to the given dataset, followed by exploratory data analysis of the same.

Thereafter, text cleaning techniques like removal of punctuation, special characters, emojis, stop words, spaces, tabs, etc including converting all words to lowercase is carried out. Stemming and Lemmatization is performed on this cleaned data to extract the root word and retain semantic similarity in reviews.

These reviews are then encoded to vectors using various encoding techniques, each of which is explained and weighed against it's advantages and disadvantages.The vectors so obtained are then sent into different clustering models to obtain meaningful interpretation and topic modelling. Implementations of models like TextRank, LDA, LSA, NMF, HDBSCAN, etc are explored.

The study also includes reading on Silhoutte techniques and Ensemble methods for clustering. Results are then presented for all the implementations with brief explanation of observations, followed by conclusion and future directions.

About Dataset

2.1 Introduction

In this case study, we aim to perform 2-level hierarchical clustering on the salonpas product review data set. The data set consists 71,674 reviews, out of which are unique. The features in the dataset include:

- Date: The date on which the review was written - 5762 unique dates
- Before Translation: No non-null values
- Review: Entire review, as posted by the author; the data consists of 20,768 unique reviews
- Author: Author of the reviews, identified by username or websites, such as Amazon, Walmart, Kindle, where the review was posted.
- Location: All locations in the data set are the US
- Before Pronoun Resolution: No non-null values
- Brand: The brands on which reviews have been written. There are 9 brands in total.
- Number of Ratings: Number of customers who have rated the product
- Price: Price of the product
- Product Rating: Rating given by customers for the products
- Review Count: Number of customers who have reviewed the product
- Review Rating: Rating of the customer review
- Review Title: Title of the review
- Root: URL of the product
- SKU: Name and SKU of product for which the review is written
- Source: Website from which the review is obtained. The websites include - Amazon, Walmart, Influenster and CVS, with Amazon being the most common website.
- Aspect: Level - 1 cluster to which the review belongs – 23 such clusters exist.
- Context Aspect: Level - 2 cluster to which the reviews belong. There are 2548 such sub-clusters.
- Sentence: Portion of review conveying the most important points
- Combined Max Score
- Overall Sentiment
- Sentiment Score

- Sentiment Confidence
- Clustering Confidence
- Context Aspect Group
- Aspect Sentiment
- Aspect Sentiment Score
- Context Aspect Sentiment
- Context Aspect Sentiment Score
- Context Aspect Group Sentiment
- Context Aspect Group Sentiment Score

2.2 Exploratory Data Analysis

In figure 2.1, which plots the most frequently occurring tokens, we observe that the token pain occurs the most followed by product and work.

Treating 'Aspect' as a label by which reviews can be clustered, we observe a highly skewed distribution of points among the clusters. Effectiveness and Pain and Spasm Types have above 20,000 reviews attributed to them while a majority of them have only around 100 reviews in the cluster, as can be seen in figure 2.2. This tells us that we may need a more equitable distribution of points among clusters, requiring different target labels.

On plotting the distribution of product prices in figure 2.3, we can clearly see that most of the products lie below $30 while some are priced at around $50. To see whether higher pricing leads to more negative reviews among buyers, we plot the distribution of product prices vs sentiments of reviews in figure 2.4. From this figure we observe that there is no significant difference between the prices of products reviewed positively, negatively and neutrally. If we expect that reviewers would critique costly products more, it would be reasonable to assume that negative reviews would have a higher average product price. However this is not the case. Thus, we infer that pricing does not create hesitation among the buyers.

In figure 2.5 we compare the review ratings for various brands. We observe that the 75^{th} quartile of each brand lies at 5 stars. Further, the brand Aspercreme has the largest variance. Finally we see that Dr.Scholls has no review ratings below 4 stars.

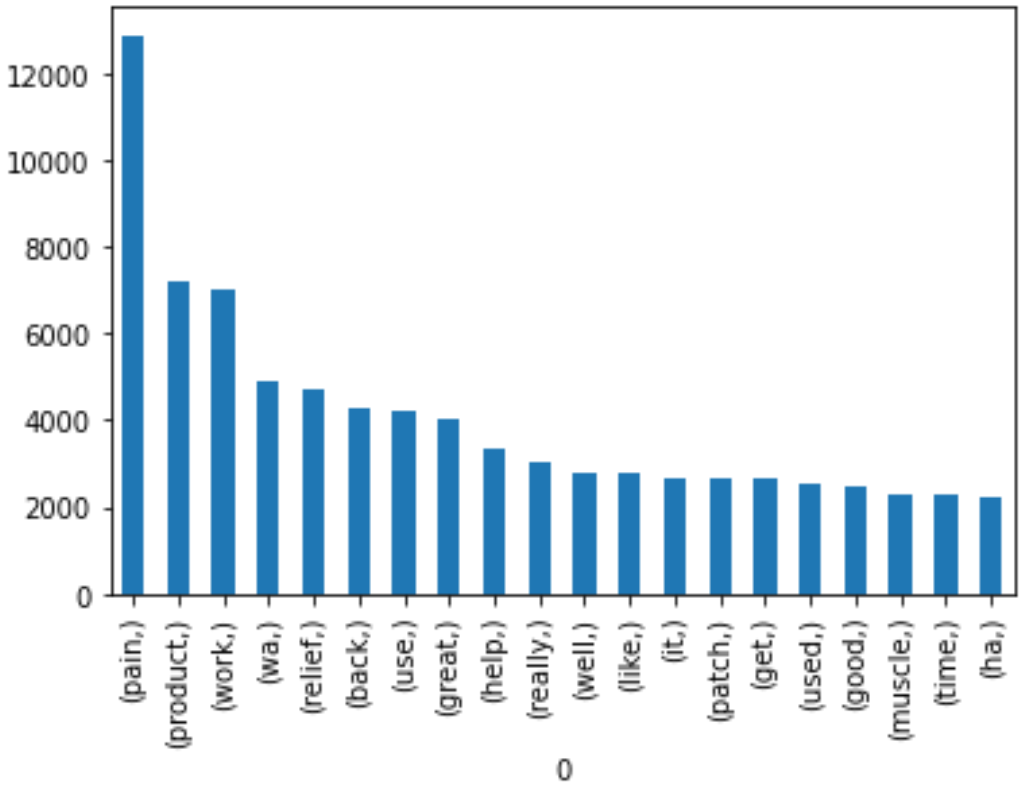

Figure 2.1: Top-20 Most Frequent Tokens and their Frequencies of Occurence in the Reviews

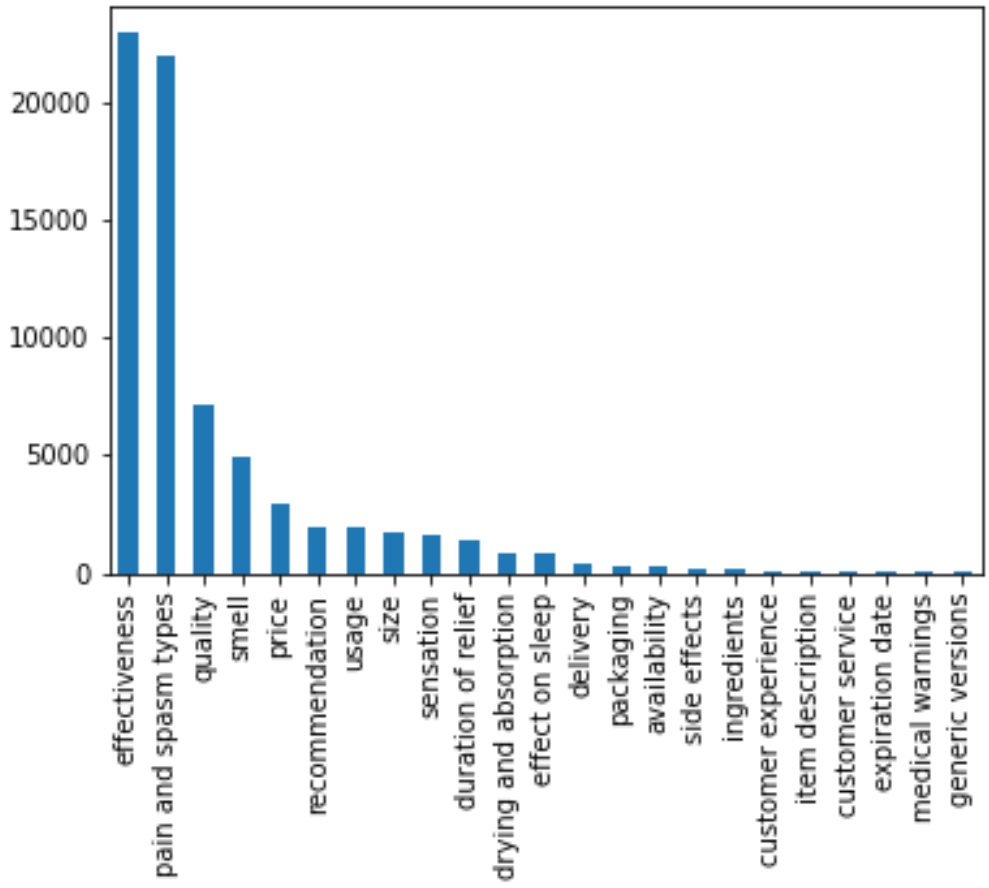

Figure 2.2: Number of Reviews in each Cluster, Clustering by Aspects

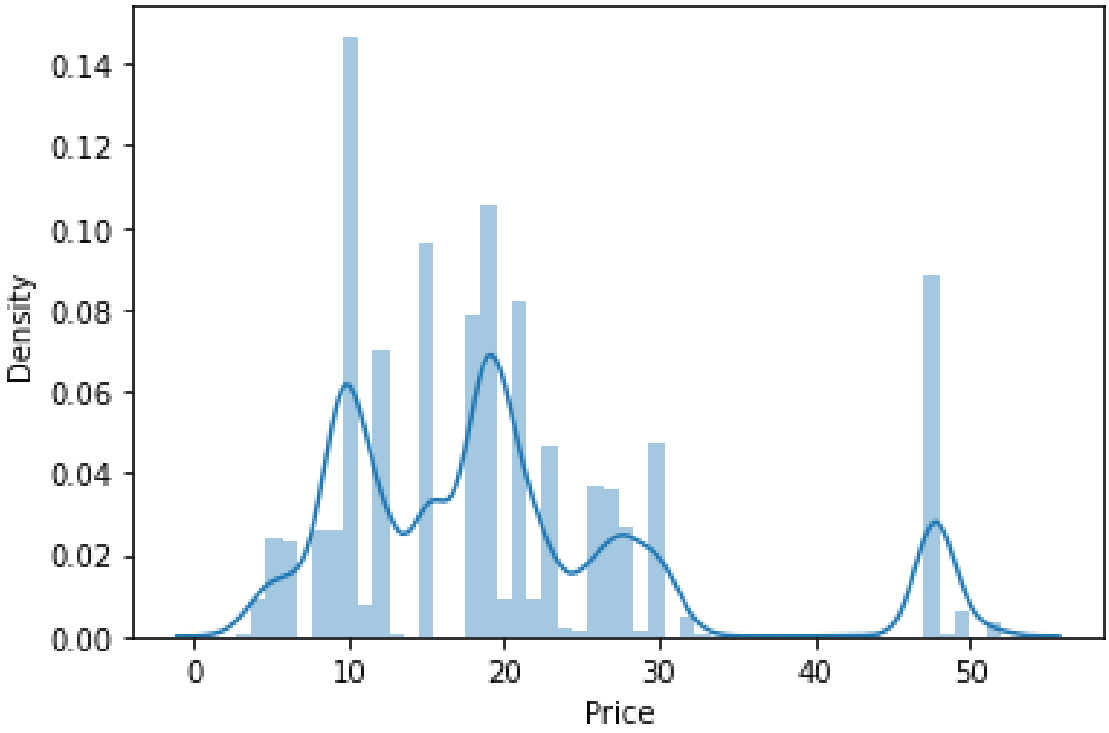

Figure 2.3: Distribution of Product Prices in the data set

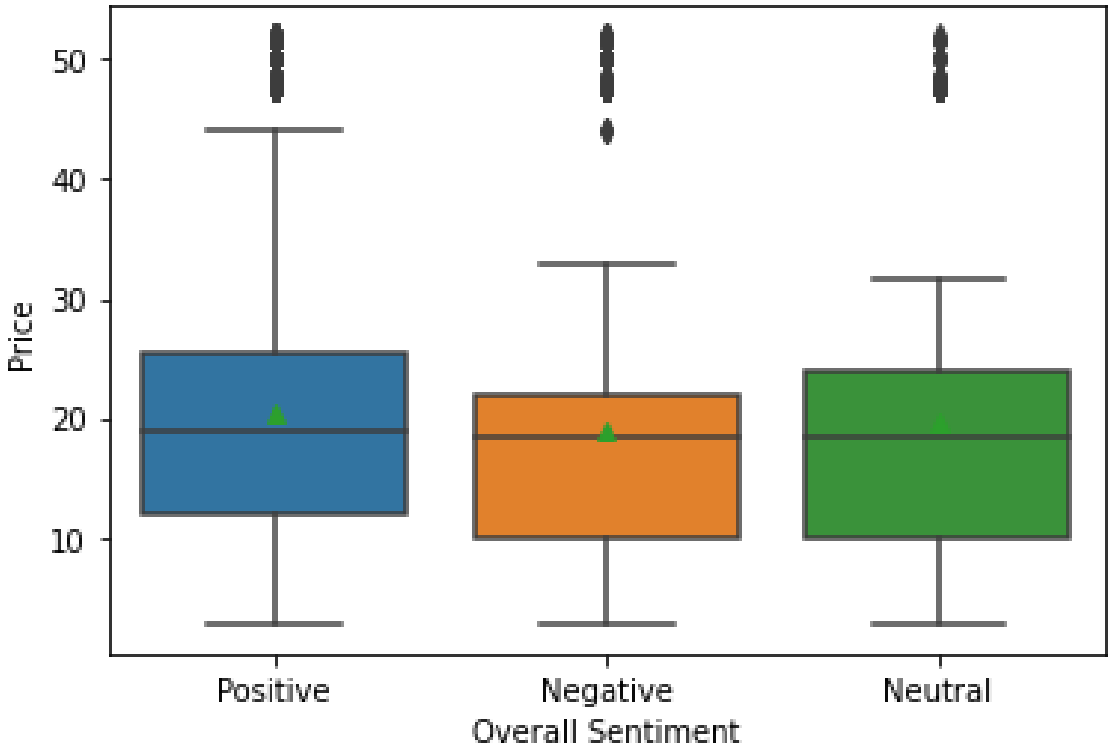

Figure 2.4: Distribution of Price of Product vs Sentiments of Reviews

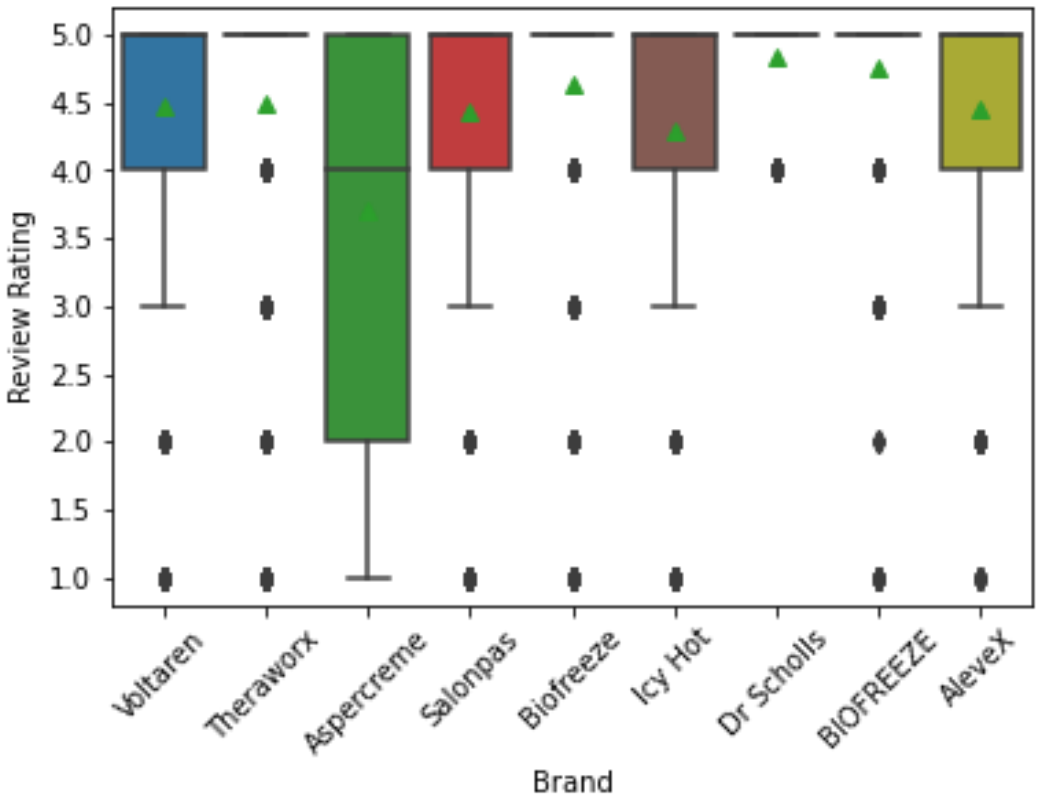

Figure 2.5: Distribution of Ratings given by Users for each Brand

Figure 2.6: Word Cloud representation of cleaned data

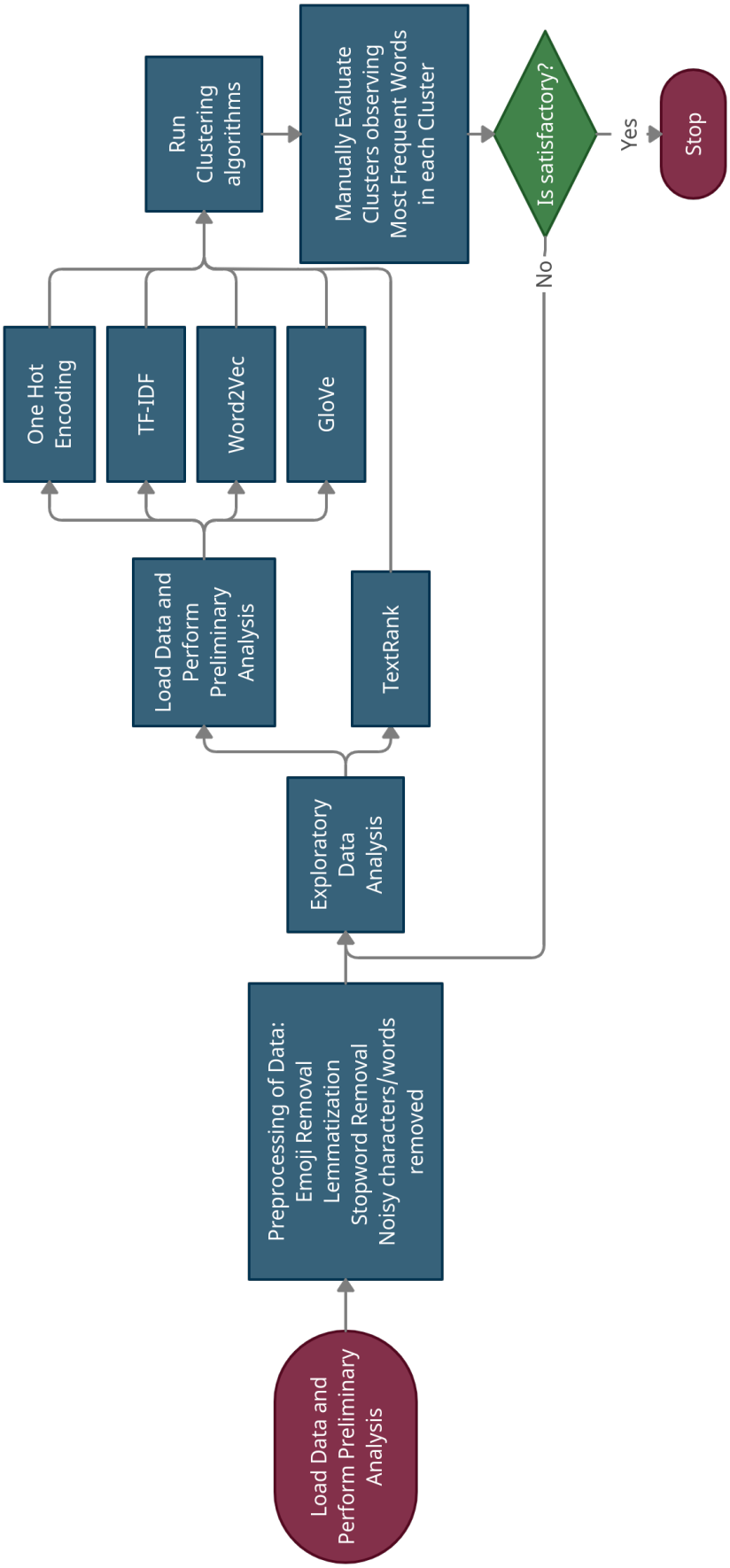

Figure 2.7: Overview of Procedure followed in this study

Pre-processing and Cleaning of Textual Data

3.1 Text Cleaning

Text Cleaning is one of the most crucial set of steps before the text is encoded to vectors. Real word data, especially text, can get very problematic if it is not cleaned and directly fed into the system. It is very similar to consuming vegetables without properly washing the dirt off. Here, we have followed a four step process to ensure the text is cleaned as much as possible. These include removal of punctuation, special characters, capitalisation and stop-words. We have also applied lemmatisation and stemming on the above cleaned text.

3.1.1 Punctuation and Special Characters

While scientific text is relatively well framed with lesser punctuation and special characters, texts from reviews/ tweets/ social media are exactly the opposite. They have a lot of emojis, '.', '...', '"' and other many such special characters that the model finds of no use and therefore have to be removed. These if converted to vectors will increase the computational expense and mislead the model by drawing unnecessary similarities.

We have used regrex and a combination of other basic libraries to remove emojis from the text. Additionally, special characters such as '.', ' ', '...', '<', '>', '++', '!', '-', spaces, new-lines and tabs are replaced with ' '. lstrip() and rstrip() is used to remove any extra spaces in the beginning or end of sentences.

3.1.2 Capitalisation

We have use str.lower() to remove any capitalisation and convert all word into lowercase. This helps the embedding techniques to assign similar vectors to words with same meaning. "THE", "The", "the" are otherwise treated as different vectors/ words due to their ASCII representations.

3.1.3 Stop words

Stop words are generally the most common words in a language. Example, the, what, where, how, has, had etc. These are filtered out before processing of natural language data. Since this is an unsupervised learning problem, we have directly filtered them out. Otherwise, number of stop words can be extracted to a feature column that can provide useful information about the structuring of sentences in cases of labelled data.

Generally, these don't add any value to vectors and hence are always removed before the words are fed into encoders. We have used a library called nltk, a basic NLP library to achieve the same.

3.1.4 Stemming and Lemmatization

Stemming and Lemmatization are closely related to each other. A stemmer returns the stem of a word, which is usually not identical to the morphological root of the word. "veri" for "very", "thi" for "this", etc. Stemmer also operates on a single word without the knowledge of context. It cannot differentiate between words which have different meaning depending on part of speech.

On the other hand, Lemmatization returns the dictionary form for the root word and hence is always valid. Here, part of speech is first determined and then the root word is found. We have used standard stemmer and lemmatizer modules from nltk library to achieve the same.

3.2 Text Embedding

Representation is an important task for any kind of data. Embeddings are representations of word (text) data. Types of embeddings decide a representation of words, where semantic meaning of words are captured and represented as real-valued vectors. The technique, of course, determines the way of representation and helps us understand the similarity between words in a sentence/ review/ paragraph. We started off exploring various kinds of embeddings for two reasons. We wanted to understand how they behave for the kind of data presented to us and check which one performs the best. We also wanted to translate and understand how the codes work, in case we are presented a similar situation in the future.

3.2.1 One-hot encoding

One-hot encoding is binary encoding process where every word is represented in the form of vectors of 1's and 0's. Additionally, every encoded word (vector) is a unique representation of a combination of 1 and 0. No two words have the same encoding. These vectors for words are then arranged in a matrix where every vector is an array representing one word, which is then fed into the model.

This type of encoding proved to be very disadvantageous for many reasons. It does not capture the similarity between words and every word is a unique vector representation in orthogonal space. This further increases the size, if we were to analyse millions of reviews amazon products receive on a daily basis. With thousands of words, the length of these arrays would be of thousands long, increase the corpus size and computational expense while not impacting the meaning by a great deal.

3.2.2 TF-IDF

TF-IDF (term frequency-inverse document frequency) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents.

TF-IDF for a word in a document is calculated by multiplying two different metrics:

- The term frequency of a word in a document. There are several ways of calculating this frequency, with the simplest being a raw count of instances a word appears in a document. Then, there are ways to adjust the frequency, by length of a document, or by the raw frequency of the most frequent word in a document.

- The inverse document frequency of the word across a set of documents. This means, how common or rare a word is in the entire document set. The closer it is to 0, the

more common a word is. This metric can be calculated by taking the total number of documents, dividing it by the number of documents that contain a word, and calculating the logarithm.

If the word is very common and appears in many documents, this number will approach 0. Otherwise, it will approach 1

$$\text{tf } idf(t, d, D) = tf(t, d).\,\text{idf}(t, D)$$

Where:

$$tf(t, d) = \log(1 + \text{ freq } (t, d))$$
$$\text{idf}(t, D) = \log\left(\frac{N}{\text{count}(d \in D : t \in d)}\right)$$

3.2.3 Word2Vec

Word2Vec is a semi-supervised 2-layer neural network that produces embeddings for text based on distributional semantics. To learn the representation of the word, it takes into account the context in which the word occurs. This results in similar words occurring with similar context words. For the context words at a position (say t), the technique looks at maximizing the likelihood of context words wrt to the center word. The likelihood function is represented as,

$$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} P\left(w_{t+j} \mid w_t; \theta\right)$$

Below is a simplified expression we found on the internet. It converts the equation to a minimisation problem by taking the log of the equation and multiplying it by -1 to calculate the -ve log-likelihood.

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

Probability of (context word | center word) is represented by, Uw and Vw, two sets of vectors.The equation for center word o and context word c was given as:

$$P(O = o \mid C = c) = \frac{\exp\left(\boldsymbol{u}_o^\top \boldsymbol{v}_c\right)}{\sum_{w \in \text{Vocab}} \exp\left(\boldsymbol{u}_w^\top \boldsymbol{v}_c\right)}$$

In this case study, we use word2Vec and other embeddings to generate embeddings for each review. To do so, we weight the vectors of each word with the tf-idf scores for each word in that review. Thus, each review is represented by a 100-dimensional vector which is further fed into the clustering algorithms

3.2.4 BERT-based embeddings

In the process of learning about different types of embedding, we came across BERT (Bidirectional Encoder Representations from Transformer) - based encoding. BERT and Transformers are increasingly being used to finding meaningful interpretations from large corpus of data, especially in the recent times.These have proved to have an edge over conventional techniques such as LSTM, RNN and GRU, due to their unsupervised nature and effective capture of long

term dependencies of words (vectors) in a temporal sequence. These models use an encoder-decoder architecture which uses attention mechanisms to extract information from data.

Even though BERT is trained on a large corpus of data and later utilised for Relation Extraction, Named Entity Recognition and other such transfer learning tasks, it can be surprisingly used to understand the semantic structure in the data. This was achieved by masked language modeling by Google, where they masked 15 percent of the words and used their positional information to infer them using the last 4 layers of the same architecture.

We haven't deployed this embedding for the current dataset, but found it very intriguing to learn about them. There are a tons of articles which we found helpful to gain an understanding of the same.

Algorithms and Approaches

In this unsupervised learning study, we try out multiple clustering approaches - Traditional ones such as Non-negative Matrix Factorization, Latent Semantic Analysis Analysis and Latent Dirichlet Allocation, as well as, combinations of current-day advanced approaches such as HDBSCAN, TextRank, etc. In this section we highlight the procedure and algorithm behind each of these approaches.

4.1 Non-negative Matrix Factorization

Non-negative Matrix Factorization (NMF) is a technique which factorises high-dimensional vectors into low-dimensional ones. NMF takes advantage of the fact that the vectors are non-negative and gives non-negative coefficients. Some terminology,

A is a document-word matrix containing words appearing in reviews.

W is the basis vectors for the topics (clusters) discovered from reviews.

H is coefficient matrix representing weights for each of the topics.

W and H is calculated by optimizing the objective function and updating parameters until convergence.

Given matrix A, W and H can be found by A= WH. NMF also has an inherent clustering property, such that

$$\frac{1}{2}\|\mathbf{A}-\mathbf{WH}\|_{\mathrm{F}}^{2}=\sum_{i=1}^{n}\sum_{j=1}^{m}\left(A_{ij}-(WH)_{ij}\right)^{2}$$

In this objective function, error is measured between A, W and H, based on euclidean distance. Using the objective function, and iterative update of W and H,

$$W_{ic}\leftarrow W_{ic}\frac{(\mathbf{A}H)_{ic}}{(\mathbf{WH}H)_{ic}}\quad H_{cj}\leftarrow H_{cj}\frac{(W\mathbf{A})_{cj}}{(W\mathbf{WH})_{cj}}$$

The updated values are calculated in parallel operations. Reconstruction error is re-calculated and the whole algorithm is repeated until convergence. This is a brief explanation of NMF algorithm. However, we have used the already existing libraries to implement the algorithm on the given dataset.

4.2 Latent Semantic Analysis

Latent Semantic Analysis (LSA), is one of the early and fundamental techniques in topic modeling. Here, the term document matrix is first generated and decomposed into seperate document-topic and topic-term matrices.

To generate document-term matrix. Given m documents and n words m × n matrix A, where each row represents a document and each column represents a word is generated. LSA models further replace raw counts in the document-term matrix with a tf-idf score. Tf-idf, or term frequency-inverse document frequency, assigns a weight for term j in document i as follows:

$$w_{i,j} = tf_{i,j} \times \log \frac{N}{df_j}$$

Intuitively, a term has a large weight when it occurs frequently across the document but infrequently across the corpus. This pattern can be clearly observed in words like "pain", "use", "product", "work", etc. This is a brief overview for LSA algorithm. However, we have used the already existing libraries to implement the algorithm on the given dataset.

4.3 Latent Dirichlet Allocation

LDA (Latent Dirichlet Allocation) is a probabilistic generative model that assumes each topic is a combination of an underlying set of words and each document is a mixture of topic probabilities.This is a popular and effective topic modelling algorithm for unsupervised learning in text analysis.

In this method, each review is encoded into a word embedding using term frequency method and is fed into the LDA model. This is a generative model which takes these vectors as inputs and models them into prior given number of topics.

The model is further visualised by pyLDAvis visualisation package present in python. It provides an interactive interface where every topic is represented as a cluster (Inter topic Distance Map) of words and contributions of top most frequent words to each cluster can be displayed.

The Intertopic Distance Plot shows how topics are related to each other. If the clusters are far away from each other, this shows that there is a clear boundary in the information pattern of words constituting the cluster. If clusters are overlapping or are subsets of each other, it shows that these are related to each other and a potential high-relation can be extracted from it.

We have also represented the most important words of these clusters in the form of "Word Clouds" for visual appeal and one glance summary of the model.The results for the same can be seen in 6.4 and 6.5.

Additionally, we have written a code to assign topics given new sentences. These sentences need to be inputed a list in "documents" variable. The code uses the probabilities of terms constituting the topics and sets a threshold value to use as a clustering baseline value. If the passed sentence (review) from documents list matches the threshold value for the corpus, it is assigned to that cluster.

For hierarchical clustering, this same code can be extended to get one cluster label for each review. Further, each cluster can be again modelled by LDA to get a further set of clusters. This task was computationally expensive on our PC, but is a possible approach to achieve assignment to two clusters.

We also tried increasing number of clusters and found that many clusters overlap with each other. This represents that reviews can belong to one or more clusters.

4.4 Hierarchical Density-Based Spatial Clustering of Applications with Noise

In this method, each review is converted to a word embedding using each of the methods listed above. To these vectors, the remaining numerical features in the data set, such as sentiment scores, are appended as additional dimensions. For example, if Word2Vec is used, we get 100-dimensional vectors for each review. Upon appending the remaining given features - price, product rating, review rating, source, combined max score, sentiment score, aspect sentiment score, context aspect sentiment score, context aspect group sentiment score, we get 109 dimensional vectors for each sample. Using this newly obtained data matrix D we perform Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN). In this algorithm,

1. Picturing each sample as a vector in a high dimensional vector space, densities around the points in the vector space are estimated
2. Regions with high density are picked to form simple clusters
3. Points in these high density regions are combined to form hierarchies of clusters

4.5 TextRank

TextRank is one of the very interesting algorithms that we came across. It is based on a popular algorithm called PageRank used for keyword extraction and text summarization. For this dataset, we have used "Extractive Summarization" wherein, we have extracted information from "Sentence" and "Review" columns and stacked (concatenated) them to generate a summarized column with information from both the columns.

This column is further broken down to sentences, then converted to vectors using term frequency tokenisation and a similarity matrix between sentences is created. The similarity matrix is converted to a graph for sentence rank calculation. In the rank matrix, sentences are vertices and similarity scores are edges. Finally, a certain number of top-ranked sentences are taken into summary. This mainly helps us with LDA/ LSA models extracting important keywords from top sentences and gives best of both "Sentences" and "Review" columns.

4.6 Word Embedding Hierarchical Clustering

Using the method described in section 4.4, we build data matrix D using each of the word embeddings described. Following this, we carry out agglomerative clustering:

1. Each data point is assigned as a single cluster
2. Determine the distance measurement and calculate the distance matrix
3. Determine the linkage criteria to merge the clusters
4. Update the distance matrix
5. Repeat the process until every data point become one cluster

In this study we use Ward's method as as the linkage criteria. To determine the right number of clusters we use the gap statistic. To get a two level hierarchical clustering, we cut the dendrogram at the level that gives us the number of clusters determined above, as well as, the level below it to get the sub-clusters.

4.7 Fuzzy Clustering using Cosine Similarity

This algorithm only applies to cases where the titles of desired clusters and sub-clusters are known. It also operates on the assumption that cluster titles are taken from among the tokens in reviews. In this method:

- Each of the n reviews and m cluster titles are converted to word-embeddings using the described methods such as Word2Vec
- The cosine similarity between each of the m titles and the j_{th} vector is calculated for all n review vectors
- Each review is assigned to the cluster with highest similarity or alternatively, it can be left fuzzy, with each review having a belongingness score for each cluster, for level-1

$$\text{cosine similarity } = S_C(R_j, T) := \cos(\theta) = \frac{\mathbf{R_j} \cdot \mathbf{T}}{\|\mathbf{R_j}\|\|\mathbf{T}\|} = \frac{\sum_{i=1}^{d} R_{ji} T_i}{\sqrt{\sum_{i=1}^{d} R_{ji}^2}\sqrt{\sum_{i=1}^{d} T_i^2}},$$

  where R_j = j^{th} Review vector of dimension d,
  T = Title vector of dimension d

- To obtain the level-2 clusters, the procedure is repeated with reviews of each cluster and titles of the sub-clusters.

4.8 Determining Optimal Number of Clusters and Sub-Clusters

Silhouette Coefficient or silhouette score is a metric used to calculate the goodness of a clustering technique. Its value ranges from -1 to 1.

- 1: Means clusters are well apart from each other and clearly distinguished
- 0: Means clusters are indifferent, or we can say that the distance between clusters is not significant
- -1: Means clusters are assigned in the wrong way

Silhouette Score = $\frac{b-a}{max(a,b)}$
where, a = average intra-cluster distance i.e the average distance between each point within a cluster
b = average inter-cluster distance i.e the average distance between all clusters.

This metric can first be used to determine the optimal number of clusters in level-1 and further, the number of sub-clusters under each cluster in level-2.

Ensemble of Clustering Algorithms

A clustering ensemble, also referred to as a consensus ensemble or clustering aggregation, can be defined as the process of combining multiple clustering models (partitions) into a single consolidated partition. In principle, an effective clustering ensemble should be able to produce more consistent, reliable and accurate clustering results compared with the individual clustering algorithms.

Problem: Given an unlabeled data set $D = \{x_1, x_2, \ldots, x_n\}$ An ensemble approach computes:
- A set of clustering solutions $\{C_1, C_2, \ldots, C_k\}$, each of which maps data to a cluster: $f_{(j)} = m$
- A unified clustering solutions f^* which combines base clustering solutions by their consensus

Challenges: The correspondence between clusters in different clustering solutions is unknown.

Future Direction: To extend this work we can combine all of the approaches highlighted in Section 4 to produce a stronger clustering algorithm in which we can place more confidence.

Results

Here, we present the various results obtained in the process of this study. Figure 6.1 shows the original dataframe which is a subset of the presented review dataset. Columns "Review and "Sentence" are of particular interest to us

	Review	Sentence	Product Rating	Sentiment Score	Aspect	Context Aspect
0	Very damaging to heart and kidneys. It helps w...	it helps with arthritis pain for a short perio...	4.5	0.8	pain and spasm types	arthritis pain
1	Very damaging to heart and kidneys. It helps w...	it helps with arthritis pain for a short perio...	4.5	0.8	size	short period
2	works quite well for neck and shoulder pain du...	works quite well for neck and shoulder pain du...	4.5	0.9	pain and spasm types	arthritis pain
3	works quite well for neck and shoulder pain du...	works quite well for neck and shoulder pain du...	4.5	0.9	pain and spasm types	shoulder pain
4	works quite well for neck and shoulder pain du...	the smell is not as strong as many of the drug...	4.5	-0.4	smell	smell strong

Figure 6.1: Initial dataframe

Figure 6.2 shows the cleaned dataframe. Columns "Review and "Sentence" are taken through a four step cleaning process. Punctuation, special characters, stop words, spaces, tabs, emojis and unnecessary characters are removed. Then the text is converted to lower case and lemmatized to extract root word of every word in the sentence.

review	sentence	rating	sentiment	aspect	context_aspect	concat
damage heart kidneys help arthritis pain short...	help arthritis pain short period time make hea...	4.5	0.8	pain and spasm types	arthritis pain	damage heart kidneys help arthritis pain short...
damage heart kidneys help arthritis pain short...	help arthritis pain short period time make hea...	4.5	0.8	size	short period	damage heart kidneys help arthritis pain short...
work quite well neck shoulder pain due arthrit...	work quite well neck shoulder pain due arthritis	4.5	0.9	pain and spasm types	arthritis pain	work quite well neck shoulder pain due arthrit...
work quite well neck shoulder pain due arthrit...	work quite well neck shoulder pain due arthritis	4.5	0.9	pain and spasm types	shoulder pain	work quite well neck shoulder pain due arthrit...
work quite well neck shoulder pain due arthrit...	smell strong many drugstore brand	4.5	-0.4	smell	smell strong	work quite well neck shoulder pain due arthrit...

Figure 6.2: Dataframe after complete text cleaning

Figures 6.3a, 6.3b, 6.3c show the top 5 most dominant words for every topic in each of NMF (Non-negative Matrix Factorization), LSA (Latent Semantic Analysis) and LDA (Latent Dirichlet Allocation) models respectively. We can notice how these clusters a clearly different from each other. For example, in 6.1, cluster 2 points to sentences which associate with "leg", "cramps", "pain" and cluster 3 is all those products associated with products that "work great" and cluster 4 is about products that give "relief" and are termed as "best" in their review.

	0	1	2	3	4
0	pain	work	cramp	great	relief
1	help	work great	leg	great product	pain relief
2	wa	really work	leg cramp	product	pain
3	product	product work	night	work great	relief pain
4	use	really	cramps	price	best

(a) Top-5 most dominant words NMF

	0	1	2	3	4
0	pain	great	cramp	product	relief
1	work	work	leg	great product	pain relief
2	product	work great	leg cramp	great	work
3	great	great product	night	relief	cramp
4	relief	product	cramps	pain relief	leg

(b) Top-5 most dominant words LSA

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
Term1	use	pain	pain	pain	work
Term2	pain	use	cramp	use	pain
Term3	work	back	use	relief	product
Term4	get	work	work	work	great
Term5	product	patch	relief	help	use

(c) Top-5 most dominant words LDA

Figure 6.4 is a collection of Word Clouds of 20 most frequent words in every topic as predicted by LDA.

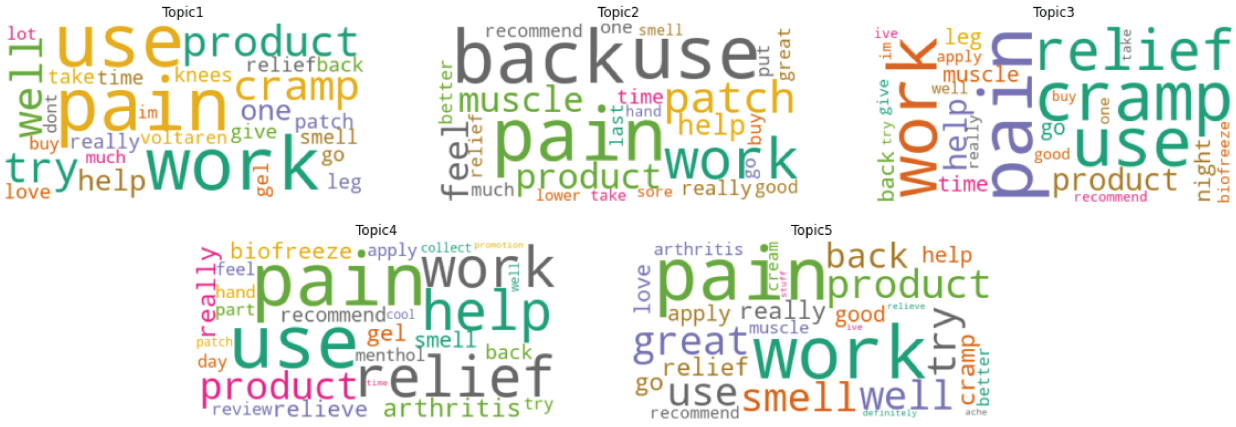

Figure 6.4: Mini Word Clouds for 5 topics as found by LDA

Figures 6.5 and 6.6 are interactive bubble visualisation plots where every topic is represented as a cluster (Inter topic Distance Map) of words and contributions of top most frequent words to each cluster can be displayed. The Intertopic Distance Plot shows how topics are related to each other. If the clusters are far away from each other, this shows that there is a clear boundary in the information pattern of words constituting the cluster. If clusters are overlapping or are subsets of each other, it shows that these are related to each other and a potential high-relation can be extracted from it.

The interactive plots are for both when LDA has 10 clusters and 5 clusters.

In figure 6.8, we see an approximate representation of the vectors learnt by Word2Vec by plotting the first two principal components. It can be inferred from this plot that the Word2Vec

model effectively places similar words together. For example, "mother" and "mom" are two similar words placed together. Similarly, the words "leg", "calf", "feet", "ankle" and "thigh" which are very related in meaning have their vectors close to eachother.

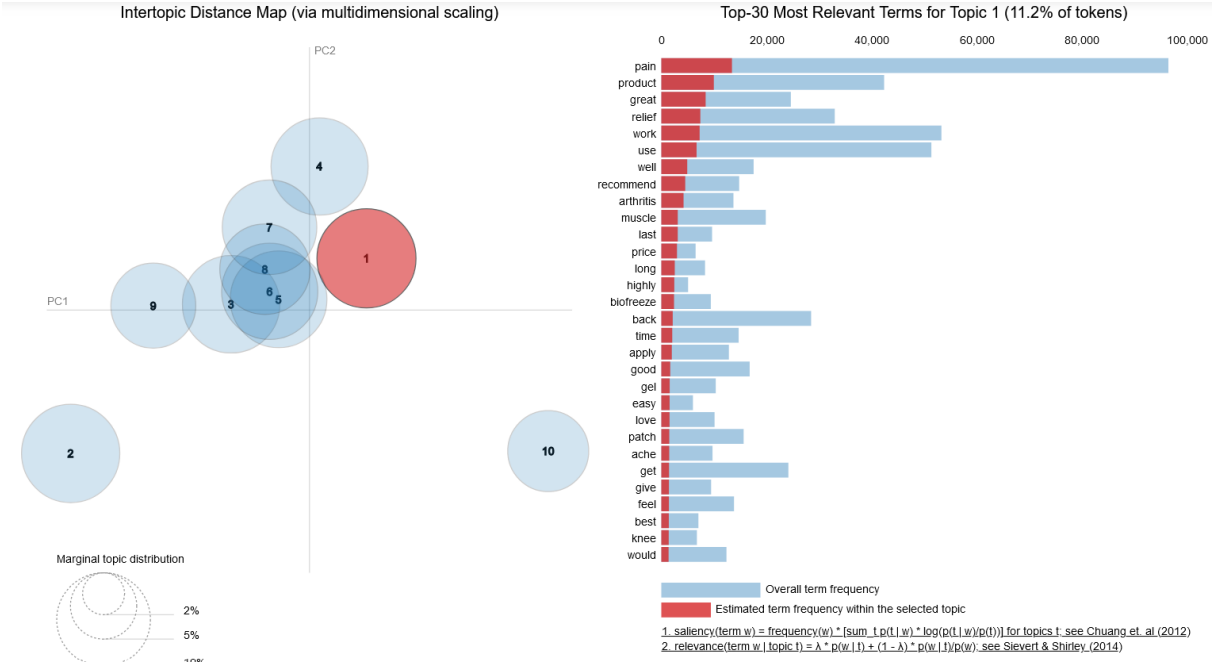

Figure 6.5: Bubble Visualisation for 10 topics

We used pyLDAvis visualisation library to plot the below graphs. The most salient terms can be changed by adjusting the relevance matrix and varying $lambda$ value

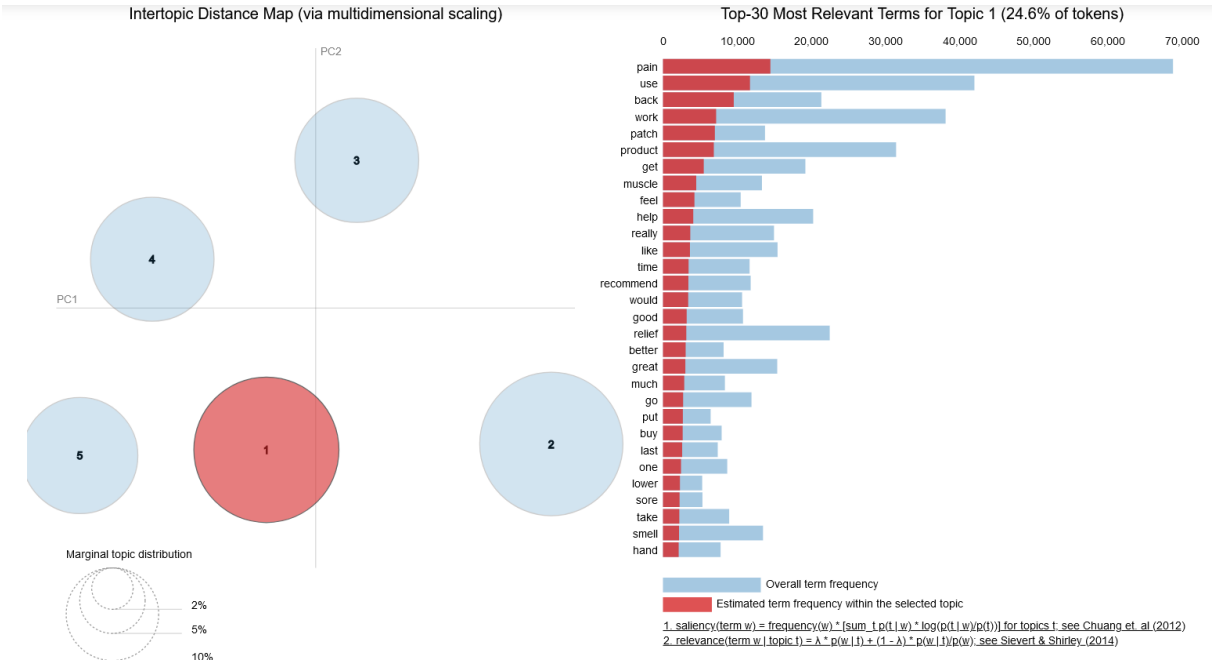

Figure 6.6: Bubble Visualisation for 5 topics

Figure 6.7 shows a table of 30 most frequent terms per topic as termed by LDA.

	Terms per Topic
Topic1	use, pain, work, get, product, try, cramp, well, help, one, really, patch, go, relief, time, give, knees, smell, would, leg
Topic2	pain, use, back, work, patch, product, get, muscle, feel, help, really, like, time, recommend, would, good, relief, better, great, much
Topic3	pain, cramp, use, work, relief, product, help, muscle, back, get, night, leg, time, would, like, go, apply, biofreeze, also, good
Topic4	pain, use, relief, work, help, product, like, arthritis, really, get, relieve, gel, biofreeze, smell, recommend, back, menthol, review, hand, part
Topic5	work, pain, product, great, use, smell, well, try, like, back, really, relief, go, cramp, apply, help, love, good, arthritis, get

Figure 6.7: Top 30 most frequent terms per topic for LDA

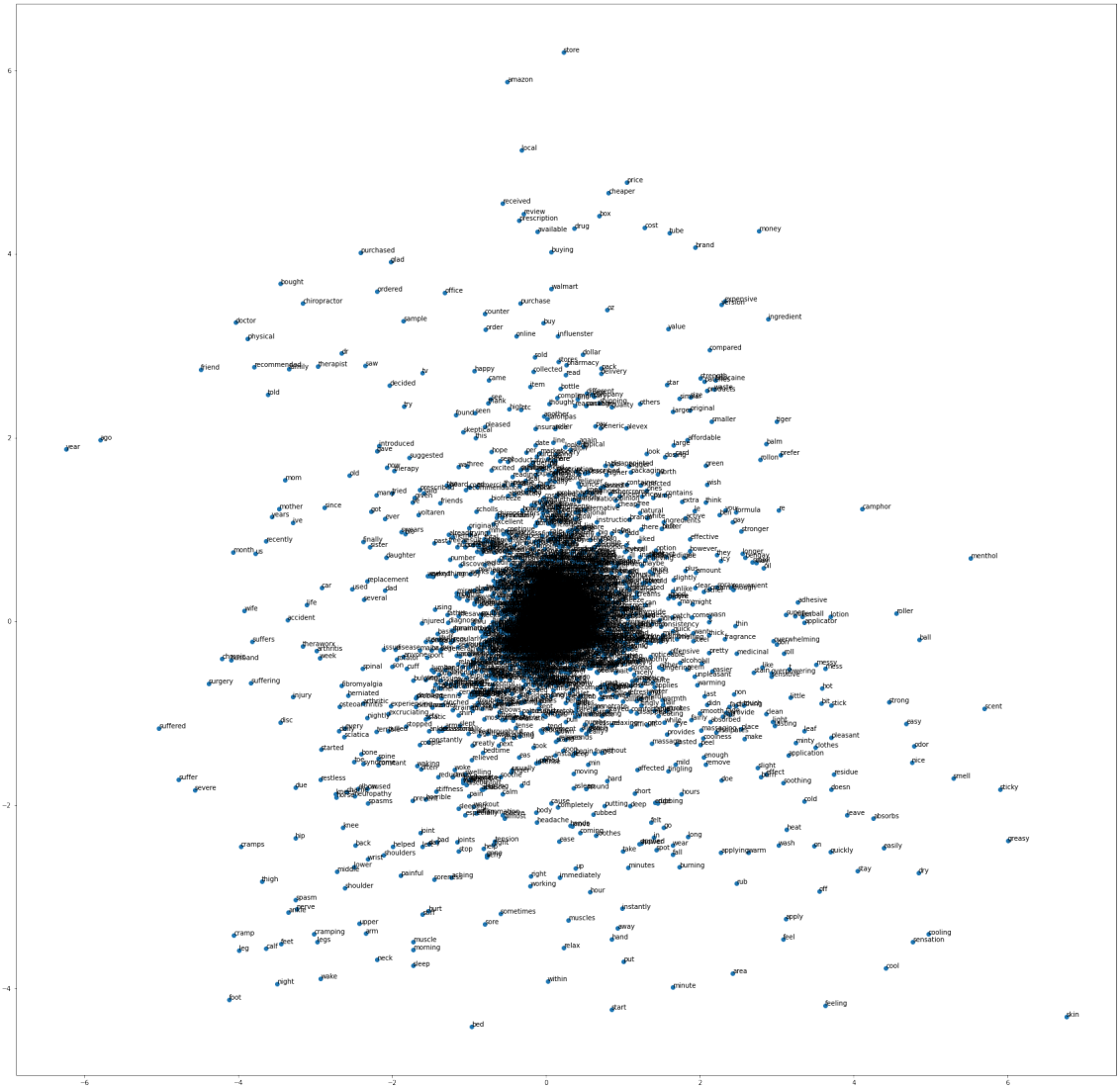

Figure 6.8: Plot of first two vectors after Principal Component Analysis on embedded Word2Vec reviews

Conclusions

In this study, we analysed the reviews textual data and attempted to make meaningful interpretations using various algorithms and approaches.

1. We started by cleaning the data, lemmatizing it and embedding it to vectors. We followed various vector embeddings and found that Word2Vec and Term Frequency methods performed excellently while Onehot encoding failed to capture the semantic similarity between words in a sentence (review).

2. While we explored textrank as one of the powerful summarization techniques, we tried to combine information from both review and sentence columns of the dataset.

3. We then tried multiple approaches to tackle the problem. We implemented NMF, LSA and LDA for topic modelling and found interesting results. We also tried algorithms like HDBSCAN, PCA, and cosine similarity.

4. The code at the end of LDA model in the notebooks submitted assigns a new review to already formed 5 clusters. We have verified it's credibility by testing on a cleaned "documents" list.

Bibliography

1. Bhardwaj, Ashutosh. "Silhouette Coefficient - Towards Data Science." Medium, 14 Dec.2021, towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10

2. Doshi, Sanket. "Latent Semantic Analysis — Deduce the Hidden Topic from the Document." Medium, 13 Dec. 2021, towardsdatascience.com/latent-semantic-analysis-deduce-the-hidden-topic-from-the-document-f360e8c0614b#:\%7E:text=Latent\%20Semantic\%20Analysis(LSA)\%20is,actual\%20topic\%20of\%20the\%20document.

3. "HDBSCAN - Michael Fuchs Python." HDBSCAN, 20 June 2020,michael-fuchs-python.netlify.app/2020/06/20/hdbscan.

4. Kumar, Satyam. "Silhouette Method — Better than Elbow Method to Find Optimal Clusters." Medium, 16 Dec. 2021, towardsdatascience.com/silhouette-method-better-than-elbow-method-to-find-optimal-clusters-378d62ff6891\#:\%7E:text=The\%20silhouette\-\%20Method\%20is\%20also,cluster\%20compared\%20to\%20other\%20clusters.

5. "Python LSI/LSA (Latent Semantic Indexing/Analysis)." DataCamp Community, 2020, www.datacamp.com/community/tutorials/discovering-hidden-topics-python.

6. Xu, Joyce. "Topic Modeling with LSA, PSLA, LDA & Lda2Vec|NanoNets." Medium, 1 July 2021,medium.com/nanonets/topic-modeling-with-lsa-psla-lda-and-lda2vec-555ff65b0b05.

7. "Latent Semantic Analysis — Deduce the Hidden Topic from the Document." Medium, 13 Dec. 2021, towardsdatascience.com/latent-semantic-analysis-deduce-the-hidden-topic-from-the-document-f360e8c0614b#:\%7E:text=Latent\%20Semantic\%20Analysis(LSA)\%20is,actual\%20topic\%20of\%20the\%20document.

8. Khangarot, Abhijeet. "News Documents Clustering Using Python (Latent Semantic Analysis)." Medium, 9 Dec. 2021, medium.com/kuzok/news-documents-clustering-using-python-latent-semantic-analysis-b95c7b68861c.

9. Chawla, Ravish. "Topic Modeling with LDA and NMF on the ABC News Headlines Dataset." Medium, 20 June 2018, medium.com/ml2vec/topic-modeling-is-an-unsupervised-learning-approach-to-clustering-documents-to-discover-topics-fdfbf30e27df.

10. Clustering Ensemble: https://cse.buffalo.edu/~jing/cse601/fa12/materials/clustering_ensemble.pdf

11. "Parameter Selection for HDBSCAN* — Hdbscan 0.8.1 Documentation." HDBSCAN, 2020, hdbscan.readthedocs.io/en/latest/parameter_selection.html.

12. Berba, Pepe. "A Gentle Introduction to HDBSCAN and Density-Based Clustering." Medium, 15 Dec. 2021, towardsdatascience.com/a-gentle-introduction-to-hdbscan-and-density-based-clustering-5fd79329c1e8.

13. Luvsandorj, Zolzaya. "Introduction to NLP - Part 3: TF-IDF Explained - Towards Data Science." Medium, 19 Dec. 2021, towardsdatascience.com/introduction-to-nlp-part-3-tf-idf-explained-cedb1fc1f7dc.

14. "PyLDAvis — PyLDAvis 2.1.2 Documentation." PyLDAvis, 2020, pyldavis.readthedocs.io/en/latest/readme.html#usage.

15. "Gensim: Topic Modelling for Humans." Gensim, 2018, radimrehurek.com/gensim/models/ldamulticore.html.

16. "Topic Modelling with LDA – Medium." Medium, 2020, towardsdatascience.com/end-to-end-topic-modeling-in-python-latent-dirichlet-allocation-lda-35ce4ed6b3e0.

17. Liang, Xu. "Understand TextRank for Keyword Extraction by Python." Medium, 7 Dec. 2021, towardsdatascience.com/textrank-for-keyword-extraction-by-python-c0bae21bcec0.

18. Joshi, Prateek. "Automatic Text Summarization Using TextRank Algorithm." Analytics Vidhya, 23 Dec. 2020, www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python.

19. Karani, Dhruvil. "Introduction to Word Embedding and Word2Vec - Towards Data Science." Medium, 2 Sept. 2020, towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa.

20. Pogiatzis, Andreas. "NLP: Contextualized Word Embeddings from BERT - Towards Data Science." Medium, 9 Dec. 2021, towardsdatascience.com/nlp-extract-contextualized-word-embeddings-from-bert-keras-tf-67ef29f60a7b#:\%7E:text=Token\%20embeddings\%20are\%20the\%20vocabulary,between\%20sentence\%20A\%20and\%20B.&text=As\%20discussed\%2C\%20BERT\%20base\%20model,used\%20as\%20a\%20word\%20embedding.

21. Jeet. "One Hot Encoding of Text Data in Natural Language Processing." Medium, 15 Dec. 2021, medium.com/analytics-vidhya/one-hot-encoding-of-text-data-in-natural-language-processing-2242fefb2148.

22. HDBSCAN Paper: https://arxiv.org/abs/1911.02282