Praxitelis-Nikolaos Kouroupetroglou

January 2019

Capstone Project

Review of Feature Extraction Techniques for Textual Sentiment Analysis

Definition

Project Overview

The project's domain background revolves around the area of Sentiment analysis. Sentiment analysis or Opinion Mining is a substantial task in Natural Language Processing, in Machine Learning and Data Science. It is used to understand the sentiment in social media, in survey responses, and healthcare for applications ranging from marketing to customer service to clinical medicine. In general Sentiment analysis main goal is to determine the attitude of a speaker or writer [1].

Historically, the Sentiment analysis originates back from WW2, during that era the primary motivation is highly political in nature. The rise of modern sentiment analysis happened only in the mid-2000s, and it focused on the product reviews available on the Web. Before 2000, the use of sentiment analysis has reached numerous other areas such as the prediction of financial markets and reactions to terrorist attacks. Moreover, the use of Sentiment analysis was useful for many problems such as irony detection and multi-lingual identification. Furthermore, over the years more research efforts are advancing from simple polarity detection to more complex identification of emotions and differentiating negative emotions such as anger and grief. Nowadays The area of sentiment analysis has become so large that anyone can face many challenges and issues when you try to keep track of all the activities in the area and the information overload [1].

In general, to process textual data, there is a need to convert the text and words to tangible data suitable for use for Exploratory data analysis, unsupervised and supervised learning. Nowadays, there are numerous feature extraction techniques that are used for this task. Some of them are the following:

- Bag-of-words or one-hot encoding or Vector Space Feature Extraction Techniques which some of them are the following:
    - TF-IDF which stands for Term Frequency – Inverse Term Frequency, is used to examine the relevance of key-words to documents in corpus [2].
    - Counter vectorization convert a collection of text documents to a matrix of token counts. This implementation produces a sparse representation of the counts of the words in a sentence [3].

Although the simplicity from these two feature extraction from text techniques there is a drawback, they lead to high dimensional spaces which from its part leads to the curse of dimensionality. However, recently more robust feature reduction methods have been developed which they contain the most related information from the textual data and reduce the textual information in a lower dimensionality space [4].

- Word Embedding Techniques, Word Embedding solve the problem of high dimensional space. Word embedding is a technique for language modelling and feature learning, which transforms words in a vocabulary to vectors of continuous real numbers. The technique normally involves a mathematic embedding from a high-dimensional sparse vector space to a lower-dimensional dense vector space. Each dimension of the embedding vector represents a latent feature of a word [5]. Two-word embedding techniques will be used for the project combined with Deep Learning models:

- Training word Embeddings
- Use of pre-trained Embeddings

Problem Statement

The project that the proposal infers to is called "Movie Review Sentiment Analysis" a past Kaggle Competition. The competition's main goal is to classify the sentiment of reviews from users from the Rotten Tomatoes dataset" and is located in Kaggle. The Rotten Tomatoes movie review dataset is a corpus of movie reviews used for sentiment analysis. This competition provides the chance to Kaggle users to implement sentiment-analysis on the Rotten Tomatoes dataset. The main task is to label phrases on a scale of five values: negative, somewhat negative, neutral, somewhat positive, positive. There are many obstacles such as sentence negation, sarcasm, terseness, language ambiguity, and many others make this task very challenging. In general, this particular Sentiment Analysis is a multiclass classification task to be faced [6].

Metrics

The performance of each classifier is evaluated using four metrics; classification accuracy, precision, recall and F1 score. It is using true positive (TP), true negative (TN), false positive (FP) and false negative (FN). True Positive (TP) stands for the number of correct predictions that a case is true which means that it is occurring when the positive prediction of the classifier agrees with a positive prediction of target variable. True Negative (TN) is the a number of correct predictions that a case is false, for example it occurs when both the classifier, and the target variable suggests the absence of a positive prediction. The False Positive (FP) is the number of incorrect predictions that a case is true. Finally, False Negative (FN) is the number of incorrect predictions that a case is false. The table below shows the confusion matrix for a two-class classifier.

	Predicted No	**Predicted Yes**
Actual No	TN	FN
Actual Yes	FP	TP

Rotten Tomatoes – Movie Review Sentiment Analysis requires all the submissions to be evaluated in their predictions' accuracy over the Test Set [10]. Classification accuracy is defined as the ratio of the number of correctly classified cases and its formula to the sum of TP and TN divided by the total number of cases.

$$Accuracy = \frac{TP + TN}{TN + FN + TP + FP}$$

Since the train set is unbalanced, F1 score as a secondary metric will be used which combines the other two metrics; precision and recall. Their formulas are the following:

Precision is defined as the number of true positives (TP) over the number of true positives plus the number of false positives (FP).

$$Precision = \frac{TP}{TP + FP}$$

The recall is defined as the number of true positives (TP) over the number of true positives plus the number of false negatives (FN).

$$Recall = \frac{TP}{TP + FN}$$

F1 score it considers both the precision and the recall.

$$\text{F1} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Analysis

Data Exploration

The dataset contains tab-separated files with phrases from the Rotten Tomatoes dataset. The train/test split has been preserved to benchmark, but the sentences have been shuffled from their original order. Each Sentence has been parsed into many phrases by the Stanford parser. Each phrase has a PhraseId. Each sentence has a SentenceId. Phrases that are repeated (such as short/common words) are only included once in the data [6].

The Train Set (source) has 4 columns and 156060 cases/rows. Its features are the following:

1. ***PhraseId***, is a unique Phrase identifier per phrase. Multiple phrases originate from the same Sentence/movie review and its type is “numeric”. We have 156060 unique PhraseIds in the train set.
2. ***SentenceId***, is a unique Sentence / review identifier. In the trainset we have 8543 unique Sentences/reviews in the train set.
3. ***Phrase***, it is type of “string” and it stems from the Sentence that is referenced by SentenceId. In total they are 156060 unique Phrases and each phrase is the result from a unique split to the Sentence /review that belongs to.
4. ***Sentiment***: Is the Sentiment Labels and the target feature that must be predicted in the Test Set. Its labels are the following: 0 – negative, 1 - somewhat negative, 2 – neutral, 3 - somewhat positive, 4 – positive.

The Test Set (source) has 3 columns and they are the following:

1. ***PhraseId***, is a unique Phrase identifier per phrase. Multiple phrases originate from the same Sentence/movie review and its type is “numeric”. We have 66292 unique PhraseIds in the test set.
2. ***SentenceId***, is a unique Sentence / review identifier. In the trainset we have 3310 unique Sentences/reviews in the test set.
3. ***Phrase***, it is type of “string” and it stems from the Sentence that is referenced by **SentenceId**. In total they are 156060 unique Phrases in the test set and each phrase is the result from a unique split to the Sentence /review that belongs to.

The following figure demonstrates how the cases look like from the train set:

PhraseId	SentenceId	Phrase	Sentiment
1	1	A series of escapades demonstrating the adage that what is good for the goose is also good for the gander , some of which occasionally amuses but none of which amounts to much of a story .	1
2	1	A series of escapades demonstrating the adage that what is good for the goose	2
3	1	A series	2
4	1	A	2
5	1	series	2
6	1	of escapades demonstrating the adage that what is good for the goose	2
7	1	of	2
8	1	escapades demonstrating the adage that what is good for the goose	2
9	1	escapades	2
10	1	demonstrating the adage that what is good for the goose	2
11	1	demonstrating the adage	2
12	1	demonstrating	2
13	1	the adage	2
14	1	the	2
15	1	adage	2
16	1	that what is good for the goose	2

Figure 1 - Example Cases from Train Set

During the training of the Machine and Deep Learning models the **PhraseId** and **SetenceId** will not be used since they do not provide any predictive advantage, they are just Id incremental numbers and they do not have any predictive ability during Machine Learning and Deep Learning training. However, the Phrases will definitely be used during the project.

Furthermore, the dataset is unbalanced, which means that the train set does not provide almost equal number of cases for all the different types of sentiment that must be predicted. This is obvious at the following figure which depicts the distribution of the sentiment at the train set:

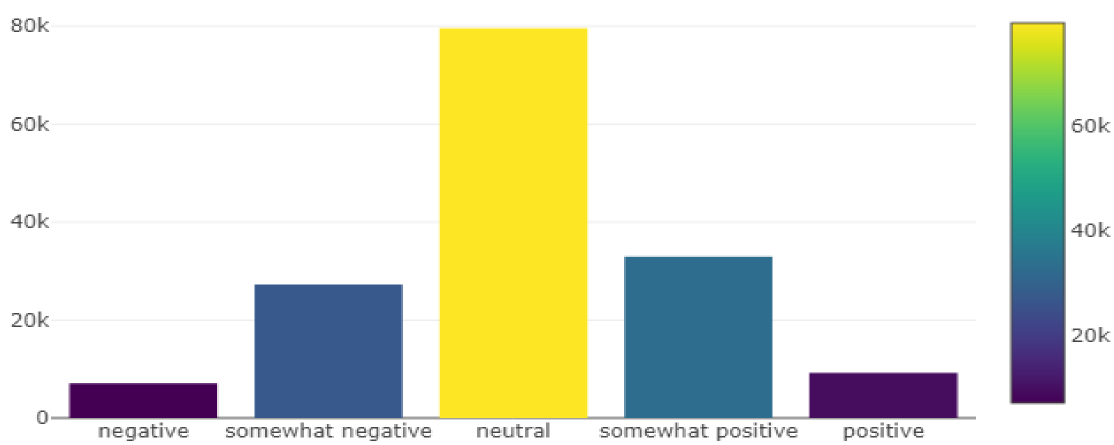


Figure 2 - Sentiment Distribution from Train Set

Sentiment Distribution	
Sentiment	**Count**
0 - negative	7072
1 - somewhat negative	27273
2 - neutral	79582
3 - somewhat positive	32927
4 - positive	9206

It is obvious that the Sentiment “2 - Neutral” is the dominant one. Having an unbalanced dataset may lead us to classifiers/models that can not identify and classify cases that belong to positive or negative Sentiments and they may misclassify them.

Moreover, the dataset does not contain any missing values, thus this help later to the analysis. At first by observing the dataset some “anomalies” were found hinting that there are some inconsistencies in the dataset. To be more specific when a word or a punctuation symbol is missing from a phrase then the Sentiment changes. Some examples pointing to this phenomenon are the following:

- The absence of full stop punctuation:

PhraseId	SentenceId	Phrase	Sentiment
1	1	A series of escapades demonstrating the adage that what is good for the goose is also good for the gander , some of which occasionally amuses but none of which amounts to much of a story .	1
2	1	A series of escapades demonstrating the adage that what is good for the goose	2

And here:

517	19	As inept as big-screen remakes of The Avengers and The Wild Wild West .	1
518	19	As inept as big-screen remakes of The Avengers and The Wild Wild West	2

- The absence of “comma (,)” in phrases changes the sentiment:

67	2	quiet , introspective and entertaining independent	4
68	2	quiet , introspective and entertaining	3

- The absence of the Exclamation mark in phrases changes the sentiment:

10737	456	Just bring on the Battle Bots , please !	3
10738	456	bring on the Battle Bots , please !	2
10739	456	bring on the Battle Bots , please	3

- Even the absence of a single word changes the sentiment:

22	1	good for the goose	3
23	1	good	3
24	1	for the goose	2

- The absence of several words changes the sentiment:

46	1	amuses but none of which amounts to much of a story	2
47	1	amuses	3

- Furthermore, strange words / symbols such as *(-RRB- -LRB-)* appear in phrases:

1225	46	Less the sensational true-crime hell-jaunt purists might like and more experimental in its storytelling -LRB- though no less horrifying for it -RRB- .	3
1226	46	the sensational true-crime hell-jaunt purists might like and more experimental in its storytelling -LRB- though no less horrifying for it -RRB- .	3
1227	46	the sensational true-crime hell-jaunt purists might like and more experimental in its storytelling -LRB- though no less horrifying for it -RRB-	4

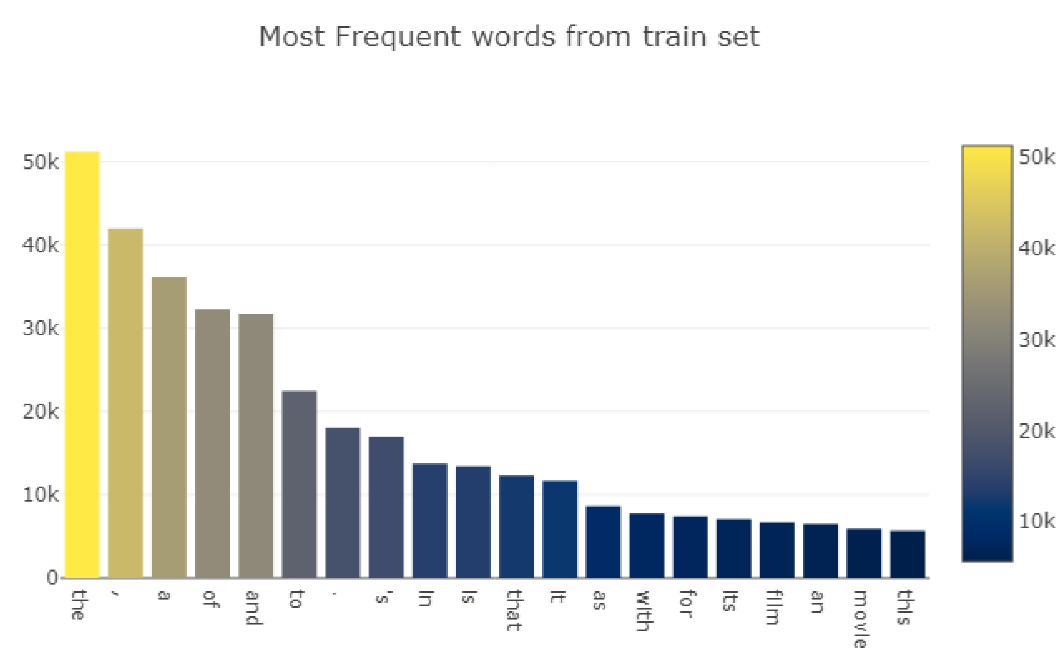
Based on these inconsistences to sentiments from phrase to phrase with just a little change they will be considered later for the Machine Learning and Deep Learning analysis.

Exploratory Visualization

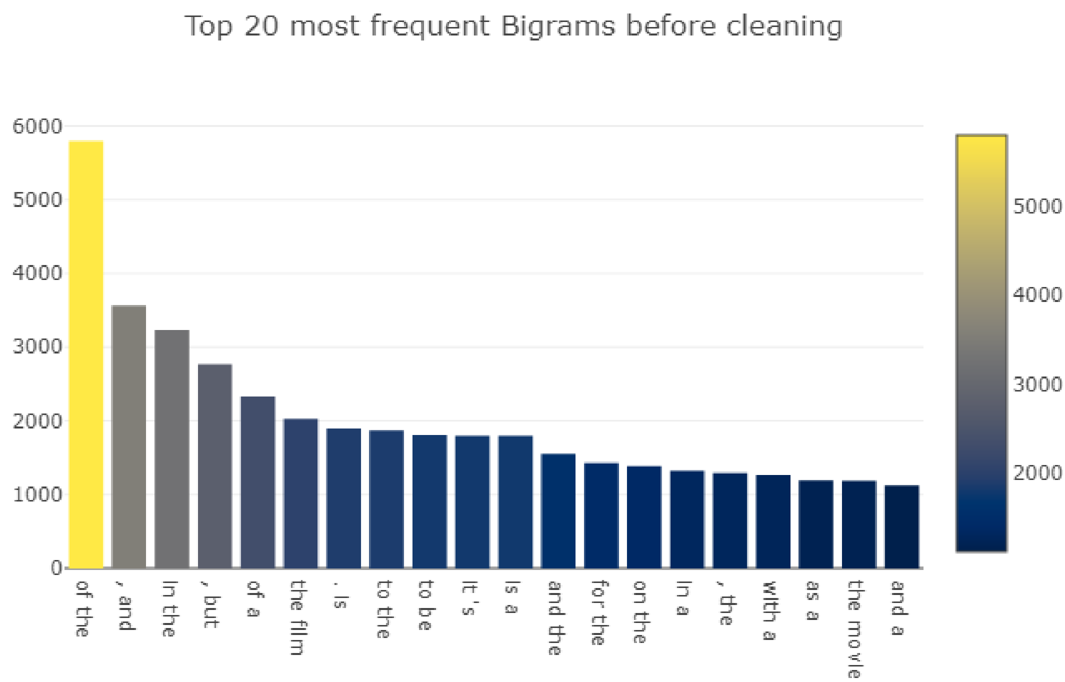
EDA Question: who are the most Frequent uncleaned words

One of the questions from Exploratory Data Analysis is what are the most frequent Unigrams, Bigrams and Trigrams in the uncleaned raw phrases from the Train Set.

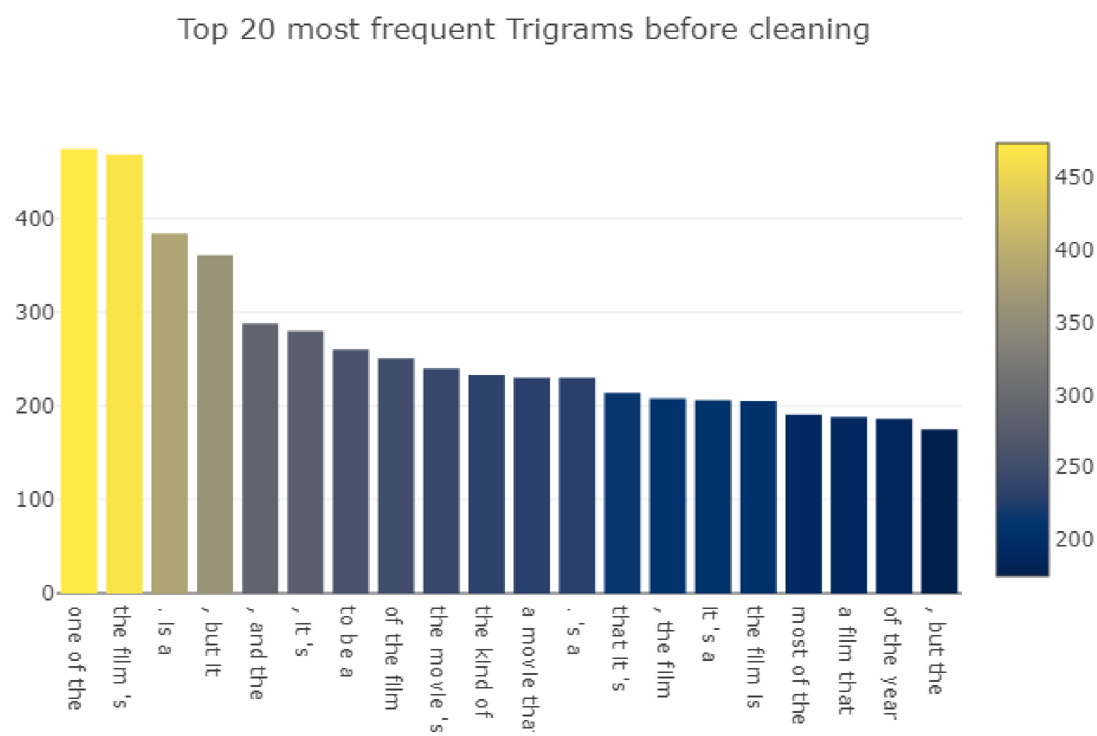
- Most Frequent uncleaned Unigrams:



Most Frequent uncleaned Bigrams:



- Most frequent uncleaned Trigrams:



It is clear that many “dirty” words like “the”, “but” etc. are very frequent in phrases. As we are moving from unigrams to trigrams these words continue to appear as frequent words and more important tangible words such as “movie” or “film” are making their appearance.

It is clear that in order to investigate the dataset and to answer questions that stem from the Exploratory Data Analysis, the text cleaning in mandatory. Text cleaning will help to remove redundant and uninformative words and will provide phrases with qualitative information.

Text Cleaning

Text cleaning is required to undercover all the hidden information from the phrases, the text cleaning steps are the following: The process that will be followed during cleaning is the following:

1. Remove redundant space, custom word simplification and removing punctuation
2. Remove Stop words
3. Lemmatize the Phrases

After the text cleaning process, the vocabulary size from the Train Set was reduced from 16540 words to 12622 words. This means that 3918 words were noisy information and hinder all the tangible information.

EDA Question: what the longest Words after Text Cleaning are

The Biggest number of characters with the longest words in the Train Set is: 18

- *'oversimplification', 'characteristically', 'transmogrification'*

The Second biggest number of characters with the longest words in the Train Set is: 17

- *'counterproductive', 'uncharismatically', 'characterizations', 'eckstraordinarily', 'characterisations', 'parapsychological', 'sanctimoniousness'*

The Third biggest number of characters with the longest words in the Train Set is: 16

- *'unapologetically', 'characterization', 'schneidermeister', 'unsalvageability', 'underappreciated', 'quintessentially', 'institutionalize', 'autobiographical', 'bruckheimeresque', 'overmanipulative', 'responsibilities', 'journalistically', 'characterisation', 'enthusiastically', 'incomprehensible', 'manipulativeness', 'unsatisfactorily', 'preposterousness'*

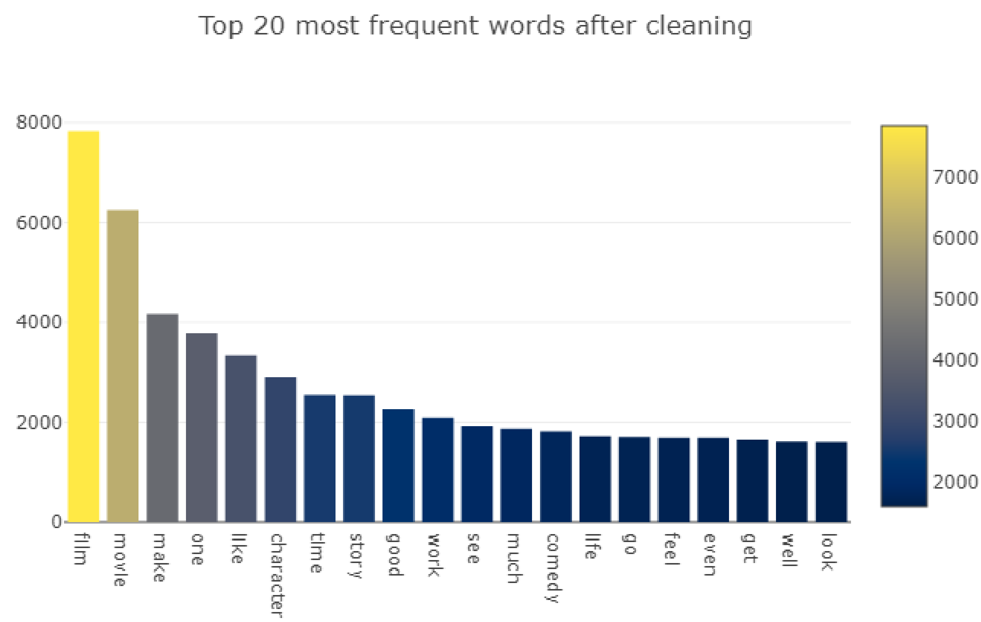
EDA Question: Visualize a Wordcloud of most frequent words after Text cleaning

The figure above, depicts all the tangible information that can be derived from the Train Set. It is obvious that words such as film and movie occur more often than others.

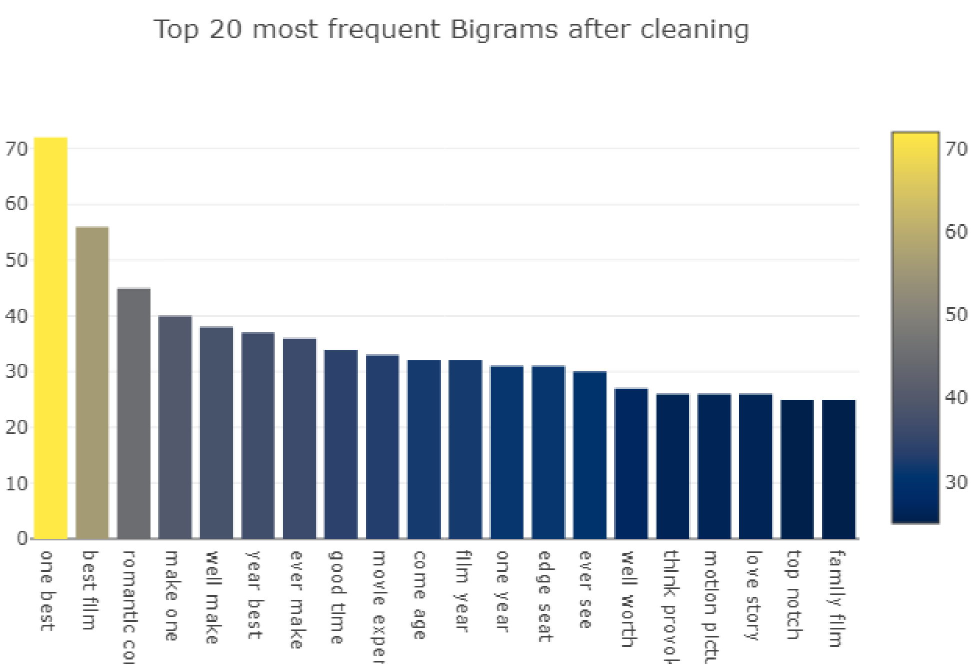
EDA Question: who are the most Frequent words after text cleaning

Now that Text cleaning took place, we return to the same question; what are the most frequent Unigrams, Bigrams and Trigrams in the cleaned phrases from the Train Set.

- Most Frequent cleaned Unigrams:



- Most Frequent cleaned Bigrams:



- Most Frequent cleaned Trigrams: Text cleaning has a drawback, information was lost due to the fact that many repetitive words have disappeared, so no trigram word frequencies could be created.

EDA Question: Can some Named Entities be extracted from the cleaned Text

Named-entity recognition is a task of information extraction that seeks to locate and classify named entity mentions in unstructured text into pre-defined categories such as the person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc source.

The extracted Named Entities from the Text are the following:

- ORGANIZATION: 'u.n.'
- PERSON: 'mr.'
- GPE/LOCATION: 'a.s.', 'u.s', 'u.s.'

The conclusion that derives from the extracted named entities is that the phrases from the reviews do not refer in specifically to a Location or an Organization and even a Person. They are just very general and express only the crowd's Sentiment.

EDA Question: Identifying most significant / important words in Phrases from reviews from Train Set using TF-IDF

tf-idf is the acronym for Term Frequency–inverse Document Frequency. It quantifies the importance of a word in relative to the vocabulary of a collection of documents or corpus. The metric depends on two factors:

Term Frequency: measures the occurrences of a word in a given document

Inverse Document Frequency: the reciprocal number of times a word occurs in a corpus of documents Think about of it this way: If the word is used extensively in all documents, its existence within a specific document will not be able to provide us much specific information about the document itself. So, the second term could be seen as a penalty term that penalizes common words such as "a", "the", "and", etc. tf-idf can therefore, be seen as a weighting scheme for words relevancy in a specific document [11].

words	**TF – IDF coefficient**
good	5.262244
time	5.131160
story	5.125591
character	4.990722
like	4.867906
one	4.744892
make	4.644150
movie	4.238848
film	4.003974

EDA Question: How is the visualization from

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 [12]. To apply the phrases from plain textual form to a vector space, TF – IDF vectorizer was used [13]. Then the TF - IDF vectors were fed to an SVD dimensionality reduction model to reduce the sparse TF – IDF matrix to a dense one to vectors size of 30 and then the latter is fed to the t-SNE algorithm to reduce the dimensions from 30 to 2 axes. The result is the following visualization:

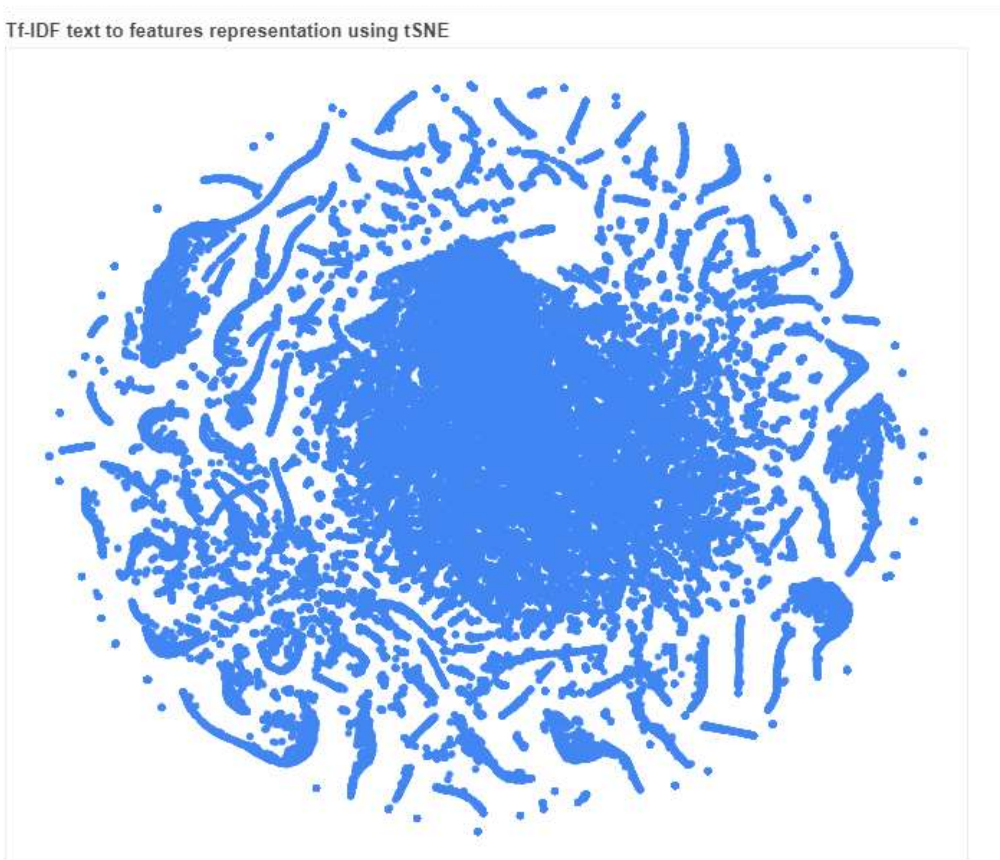


Figure 3 - TF - IDF vectorized Phrases visualized with t-SNE

EDA Question: Can the phrases be clustered using Kmeans and what are the centers

In order to cluster the phrases from the trainset, they must be applied to the TF-IDF vectorizer from sklearn [13]. To find the optimal number of clusters the use of Silhouette score was applied which measures whether or not a case is assigned to the current cluster. The Silhouette score presented that 12 clusters are the optimal number. Finally, to visualize again in 2 axes, t-SNE algorithm was applied to the distances of the cases from their cluster centers they were appointed to. The result is the following visualization:

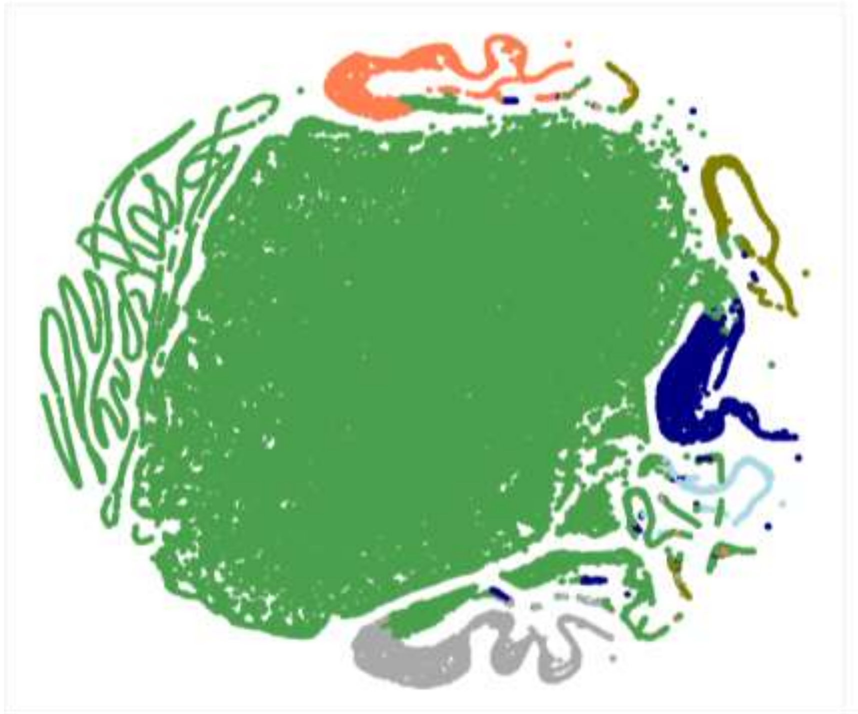


Figure 4 - Kmeans Clusters visualized with t-SNE

The cluster centers that are most representative terms for each cluster are the following:

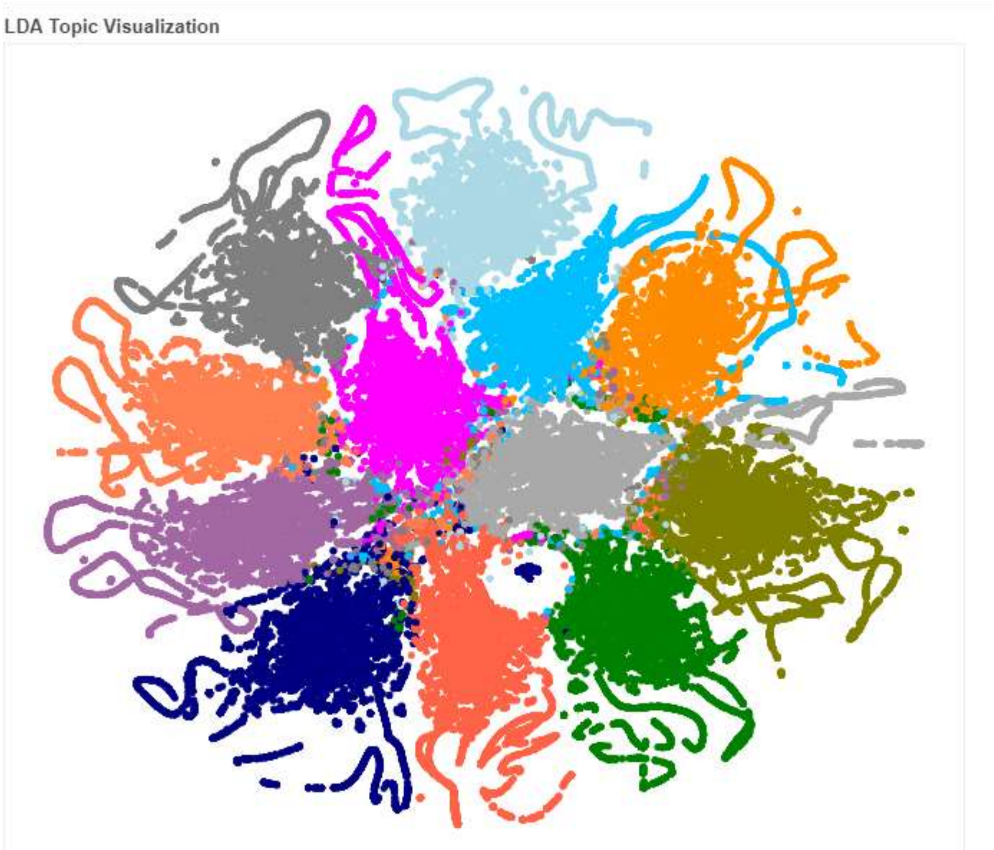
Representative terms per cluster center:

- Cluster 0: time |story |interest |bad |go| run |tell |good |love |run time
- Cluster 1: director bruce |bruce mcculloch |outstanding director |mcculloch |bruce |outstanding |director |funny |expect much |talent outstanding
- Cluster 2: masterpiece elegant|elegant wit |wit artifice |artifice |elegant |masterpiece |wit |wilde play |wilde |play
- Cluster 3: make |movie |film |make movie |well make |well |movies |make film |like |make movies
- Cluster 4: memories one |fantastic visual |visual trope |one fantastic | trope| memories| fantastic| visual| one | daydream memories
- Cluster 5: movie |bad |one |bad movie |like |action movie |see |good |action |good movie
- Cluster 6: one |character |like |work |good |see |much |comedy |life |get
- Cluster 7: macy thanksgiving| day parade| parade balloon| thanksgiving day |balloon |macy |thanksgiving |parade |day| comedy
- Cluster 8: film |one |good film |good |first |like |action film |best |see |best film
- Cluster 9: infamy |charm |charm little |say picture |respective |little |best thing |thing say |bullock hugh |cute moments
- Cluster 10: way |new |York |new York |york city |get way |movie |city |find |long way
- Cluster 11: anti feminist |feminist equation |familiar anti |equation |feminist |anti |familiar |career kid |kid misery |misery

EDA Question: Can LDA (Latent Dirichlet Allocation algorithm) model topics in phrases

Latent Dirichlet Allocation (LDA) is an algorithms used to discover the topics that are present in a corpus.

LDA starts from a fixed number of topics. Each topic is represented as a distribution over words, and each document is then represented as a distribution over topics. Although the tokens themselves are meaningless, the probability distributions over words provided by the topics provide a sense of the different ideas contained in the documents [14]. Both K-means and Latent Dirichlet Allocation (LDA) are unsupervised learning algorithms, where the user needs to decide a priori the parameter K, respectively the number of clusters and the number of topics. If both are applied to assign K topics to a set of N documents, the most evident difference is that K-means is going to partition the N documents in K disjoint clusters (i.e. topics in this case). On the other hand, LDA assigns a document to a mixture of topics. Therefore, each document is characterized by one or more topics (e.g. Document D belongs for 60% to Topic A, 30% to topic B and 10% to topic E). Hence, LDA can give more realistic results than k-means for topic assignment [15]. Its input is a bag of words, i.e. each document represented as a row, with each column containing the count of words in the corpus. In order to find the correct number of LDA topics a grid search based on the LDA model's perplexity was applied and 12 topics was the optimal number [16]. The following figure illustrates the LDA topics depicted in 2 axes with the aid of t-SNE algorithm:



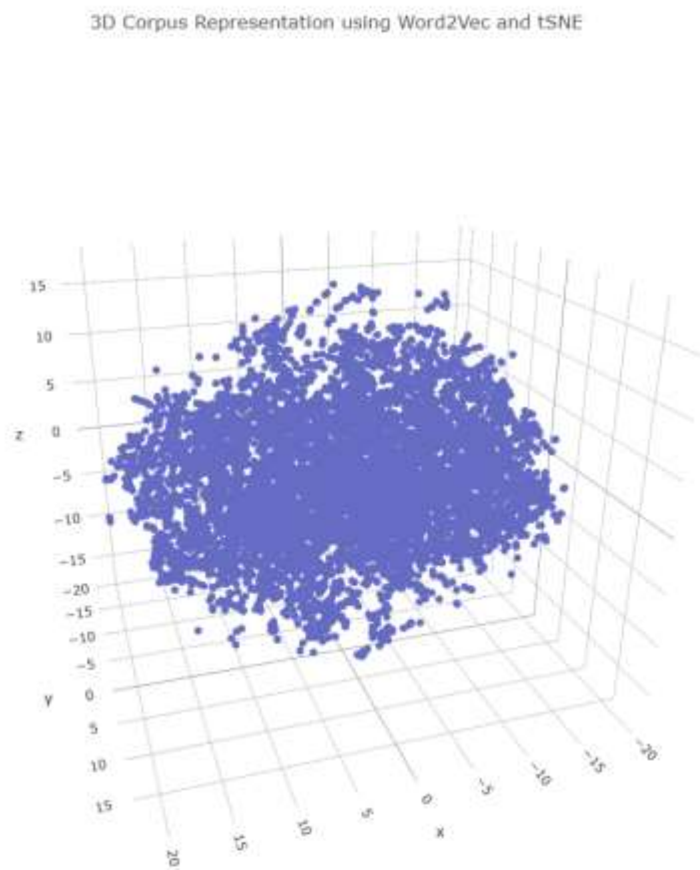
Top representative keywords per topic:

- Topic 0: movie | story | love | interest | minutes | hollywood | entertain | less | need | set
- Topic 1: film | go | little | give | never | may | could | human | young | emotional
- Topic 2: one | time | plot | watch | old | another | hard | bite | right | material
- Topic 3: good | director | us | something | many | cast | sense | humor | want | laugh
- Topic 4: make | see | movie | would | one | without | ever | nothing | long | kind
- Topic 5: get | feel | well | movies | best | first | try | year | show | know
- Topic 6: character | work | comedy | funny | bad | world | drama | screen | big | charm
- Topic 7: people | think | play | leave | kid | often | might | things | moments | face
- Topic 8: like | look | enough | end | seem | self | live | still | run | move
- Topic 9: life | come | act | action | two | really | every | man | great | real
- Topic 10: much | new | audience | better | family | script | performance | heart | cinema | full

- Topic 11: even | way | take | find | turn | back | keep | also | almost | thriller

EDA Question: Can the words from the phrases visualized in 3D axes

Examining the phrases from the reviews back to words and try to visualize the words in 3D axes now using again the t-SNE algorithm. Here in order to convert the words into a tangible form the Word Embeddings technique will be used. Word embedding is one of the most popular feature representation of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words Word embeddings are vector representations of a particular word. Word2Vec is one of the most popular technique to learn word embeddings using shallow neural network. It was developed by Tomas Mikolov in 2013 at Google [17]. To illustration below depicts word embeddings from words by the phrases from the trainset. The words were applied to a trained word2vec model and reduced their dimensions to 3 with the help of t-SNE algorithm.



EDA and Unsupervised Learning Summary

During EDA it was clear and inevitable that text cleaning must be applied in order to discover how the people express from sentiment to sentiment.

Furthermore, during EDA various insights have been discovered from word frequencies, bigrams, trigrams, named entities, wordclouds, most relevant words per sentiment.

During Unsupervised Learning 2 major dimensionality reduction techniques were applied; SVD and PCA. SVD as a formula to reduce the dimensions from the TF - IDF matrix to 30 dimensions and PCA as a parameter inside t-SNE algorithm. Phrases / reviews and words visualizations in 2D and 3D were created with the aid of t-SNE depicting the phrases in the cartesian system, Kmeans clustering, LDA Topic Modeling and Word Embeddings.

Algorithms and Techniques

The next process from EDA and Unsupervised Learning is the Supervised Learning. The goal of this capstone project is to evaluate 3 different feature extraction / representation techniques and apply them in Machine Learning and Deep Learning predictive models. To sum up the following experiments will be performed:

1. Create Machine Learning models with
   - Feature Extraction using TF – IDF
   - Download and use of pretrained Word Embeddings for Feature Extraction.
2. Create Deep Learning models with
   - Live on-premise training for Word Embeddings
   - Download and use of pretrained Word Embeddings for Feature Extraction

In every experiment I will evaluate my models with train – validation split with ratio of 80 / 20 to evaluate the models' performance.

The Machine Learning models that will be used are the following

- Logistic Regression (LR), Logistic Regression, the most prevalent algorithm for solving industry scale problems, although its losing ground to other techniques with progress in efficiency and implementation ease of other complex algorithms.
- K-Nearest Neighbors (KNN), is a simple machine learning algorithm that categorizes an input by using its k nearest neighbors. K-NN is non-parametric, which means that it does not make any assumptions about the probability distribution of the input. This is useful for applications with input properties that are unknown and therefore makes k-NN more robust than algorithms that are parametric.
- Classification Trees (CART), Decision trees cut feature space in rectangles which can adjust themselves to any monotonic transformation. Since decision trees are designed to work with discrete intervals or classes of predictors
- Naive Bayes models (NB), naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features
- Support Vector Machines (SVM), A Support Vector Machine is a supervised machine learning algorithm that can be employed for both classification and regression purposes. SVMs are more commonly used in classification problems and as such, SVMs are based on the idea of finding a hyperplane that best divides a dataset into two classes.
- Random Forests (RF), Random forest is just an improvement over the top of the decision tree algorithm. The core idea behind Random Forest is to generate multiple small decision trees from random subsets of the data (hence the name "Random Forest")
- XGBoost (XGB), XGBoost is one of the state-of-the-art algorithms. XGBoost is a part of an ensemble of classifiers which are used to win data science competitions. XGBoost is similar to gradient boosting algorithm but it has a few tricks up its sleeve which makes it stand out from the rest.
- Ensemble after training and evaluation, select the top performed Machine Learning Models using the statistical mode over the predicted classes from the validation set and later on the test set.

The list above summarizes all the Machine Learning families. They will be used and evaluated each one of them and those with the best accuracy will be kept.

The Deep Learning architectures that will be used is the following:

- Long Short-term Memory Recurrent Networks (LSTM), Long Short Term Memory networks LSTMs are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997) and were refined and popularized by many people in following work. They work tremendously well on a large variety of problems and are now widely used. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn! All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer. LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way [18].
- Bidirectional Long Short-term Memory Recurrent Networks (BiLSTM). A major issue with all the Recurrent networks is that they learn representations from previous time steps. Sometimes, you might have to learn representations from future time steps to better understand the context and eliminate ambiguity. Take the following examples, "He said, Teddy bears are on sale" and "He said, Teddy Roosevelt was a great President". In the above two sentences, when we are looking at the word "Teddy" and the previous two words "He said", we might not be able to understand if the sentence refers to the President or Teddy bears. Therefore, to resolve this ambiguity, we need to look ahead. This is what Bidirectional RNNs accomplish. The repeating module in a Bidirectional RNN could be a conventional RNN, LSTM or GRU [19].
- Convolutional Neural Networks (CNN). Convolutional Neural Networks are very famous for applications in image classification. The whole idea about ConvNets stems from the notion that by adding more and more layers to the network the DL model can understand more and more features from an image and categorize it easier and more efficiently [20]. Moreover, the same architecture presents great results with Text classification problems [21].
- Long Short-term Memory Recurrent Networks - Convolutional Neural Networks (LSTM - CNN). There are papers in the scientific literature that combine both LSTM and CNN to improve the DL model's performance by deepening the network [22].
- Bidirectional Long Short-term Memory Recurrent Networks - Convolutional Neural Networks (BiLSTM - CNN), Following the precious paradigm lets took the liberty to combine and a bidirectional LSTM and a CNN together.

Ensemble after training and evaluation the top performed Deep Learning Models using the statistical mode over the predicted classes from the validation set and later on the test set.

The Deep Learning models were chosen based on the scientific literature and because the deeper the Deep Learning architecture the better fit to the data [5].

Benchmark

The given dataset is a typical supervised learning problem. In Machine Learning and in general in many Kaggle Competitions XGB - Extreme Gradient Boosting models perform better than others [7]. So Extreme Gradient Boosting (XGB) as a benchmark will be picked and it will be tried to try to be the benchmark with other machine learning models. The more Machine Learning models the better, they may even try to outperform XGBoost.

For the Deep Learning models, as a benchmark will be used the LSTM - Long Short-term Memory Recurrent Networks. The notion behind this pick is a philosophical principle which is called "Occam's Razor" which says that between two explanations choose the one that is has the least speculations/assumptions [9]. In other words, sometimes follow the simplest ideas. Since LSTM is simpler to be implemented in code than the other 4 Deep Learning models as described above, then this model as a benchmark will be picked and I will try to be the benchmark with the rest Deep Learning models. Besides, LSTM models are widely used for Sentiment Analysis [8], so based on that I will try to find more effective Deep Learning models to increase their accuracy.

Methodology

Data Preprocessing

At first, there must be mention that after EDA an odd conclusion was made. The dataset of this competition turned to have some unique features. we have only phrases as data. And a phrase can contain a single word. And one punctuation mark can cause phrase to receive a different sentiment. Also assigned sentiments can be strange. This means several things:

- using stopwords can be a bad idea, especially when phrases contain one single stopword.
- puntuation could be important, so it should be used
- ngrams are necessary to get the most info from data

As you can see sentence id denotes a single review with the phrase column having the entire review text as an input instance followed by random suffixes of the same sentence to form multiple phrases with subsequent phrase ids. This repeats for every single new sentence id (or new review per se). The sentiment is coded with 5 values 0= Very negative to 4=Very positive and everything else in between.

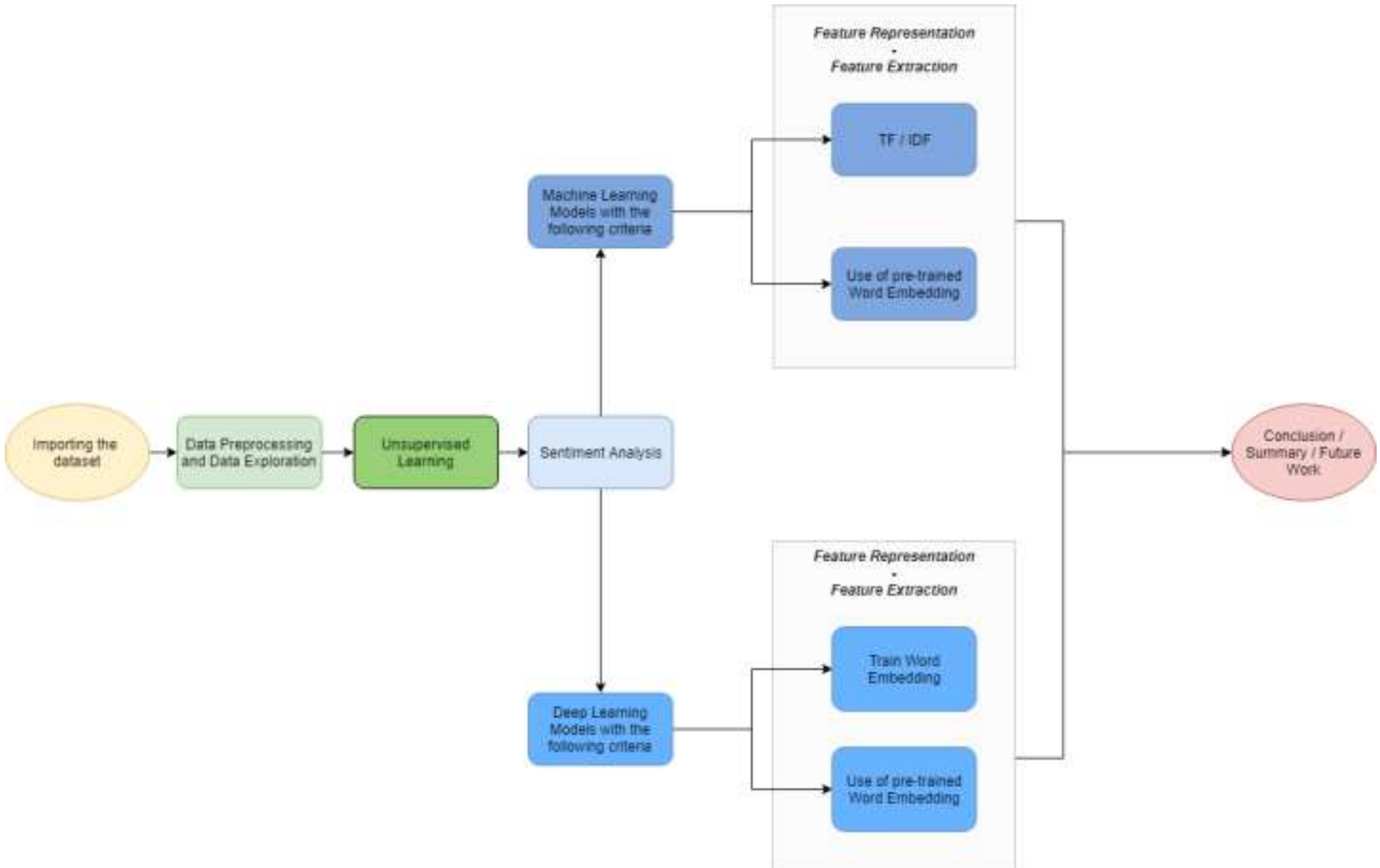
A quick glance will show you that the data is a little weird for a sentiment corpus:

- Phrases of sentences are chopped up completely randomly. So, logic like sentence tokenization based on periods or punctuations or something of that sort doesn't apply
- Certain phrases are with one single word!
- For some phrases inclusion of a punctuation like a comma or a full stop changes the sentiment from say 2 to 3 i.e neutral to positive.
- Some phrases start with a punctuation like a backquote.
- Some phrases end with a punctuation
- There are some weird words such as -RRB-, -LRB-
- All these weird aspects of this dataset can be helpful and may be predictive. Afterall, we are looking for patterns in data. Therefore, it would be easier for us to engineer features, I mean apart from the text features that can be extracted from the corpus.

Implementation

The Kaggle Competition is Kernel based, this means that all the code must be executed on Kaggle premises and

The project follows typical predictive analytics hierarchy as shown in the following figure:



Following the direction of the arrow as shown, with the dataset we chose the workflow of solving this problem will be in the following order:

1. Loading the data
2. Data Preprocessing and Data Exploration.
    a. Cleaning the text data from noisy information.
    b. Observing anomalies in the Train Set
    c. Measure word frequencies (Unigrams, Bigrams and Trigrams).
    d. Recognize named entities.
    e. Create wordclouds.
    f. Discover most significant words.
3. Unsupervised Learning
    a. Train set reviews' visualization over the 2-axis using t-SNE.
    b. K-means clustering over the reviews from train set and visualize the clusters using t-SNE.
    c. Topic Detection over the reviews from train set using LDA (Latent Dirichlet Allocation algorithm) and visualize the topics using t-SNE.
    d. Word Embeddings over the train set and visualize their similarity using t-SNE.
    e. Dimensionality reduction techniques such as PCA – Principal Component Analysis and SVD – singular value Decomposition may be used during Unsupervised Learning.

For Steps 1 to 3 a Kaggle Python Jupyter notebook has been created can be found here.

4. Machine Learning
    a. Apply Machine Learning models and measure their accuracy using TF – IDF as feature extraction.
    b. Apply Machine Learning models and measure their accuracy using word embeddings as feature extraction.

For the Step 4a, a Kaggle Python Jupyter notebook has been created can be found here.

For the Step 4b, a Kaggle Python Jupyter notebook has been created can be found here.

5. Deep Learning
    a. Apply Deep Learning models and measure their accuracy using the training of word embeddings as feature extraction.
    b. Apply Deep Learning models and measure their accuracy using pretrained word embeddings as feature extraction.

For the Step 5a, a Kaggle Python Jupyter notebook has been created can be found here.

For the Step 5b, a Kaggle Python Jupyter notebook has been created can be found here.

6. Summarize, Conclusions, Future Work

Refinement

It must be noted that during the Machine Learning phase, having TF – IDF as feature extraction / representation technique Decision Trees, Random Forest, Extra Tree and Extra Trees with default parameters outperformed the XGBoost, XGBoost's accuracy with default parameters was close to 0.54 and the other 4 ML models were close to 0.64 to 0.65. Thus I concluded that Boosting Trees do not work with this feature representation and only the other 4 do work. So then there was a need to tune these top 4 Machine Learning models to improve accuracy and I left XGBoost model.

From the other hand tuning the Deep Learning models was very time consuming due to time limitations from Kaggle Kernel run time.

Results

Model Evaluation and Validation

The trainset was split in ratio 80:20 train and validation set respectively. In every execution the textual data was transformed in either TF – IDF matrix, trainable word embedding matrix or pre-trained word embeddings matrix.

Machine Learning Models evaluated over the Test Set from Kaggle:

The Machine Learning models that were developed along with 2 feature extraction techniques:

- TF – IDF as feature extraction / representation
- pre-trained word embeddings as feature extraction / representations

- *Machine Learning models and TF – IDF as feature extraction / representation:*

The trainset and the test set were converted via the TF – IDF vectorizer from sklearn. We applied and compared XGB model out of the box vs the rest of Machine Learning models. The following table show the accuracy results over the Test Set and submitted to the Kaggle:

XGBoost model performed poorly with default parameters than most of the rest Machine Learning models. From this Execution only ExtraTrees, RandomForest, Logistic Regression and SVM were performed better than others. Their selection as based on their accuracy and their F1-score. Their performance results over the Validation and Test Set are the following:

ML Models	Accuracy over the Validation Set	F1-score over the Validation Set	Accuracy over the Test Set
Logistic Regression	0.633	0.639	0.5772
Linear SVM	0.655	0.646	0.6089
Extra Trees	0.628	0.616	0.5916
Random Forest	0.622	0.601	0.585
Ensemble ML Models with the statistical mode	0.647	0.628	0.5966
XGBoost	0.544	0.449	Did not apply for Test Set predictions due to low performance

And after tuning and ensemble the above top performed ML models the performance results over the Validation and Test Set are the following:

Tuned ML Models	Accuracy over the Validation Set	F1-score over the Validation Set	Accuracy over the Test Set
Tuned logistic Regression	0.658	0.543	0.610
Tuned Linear SVM	0.656	0.545	0.607
Tuned Extra Trees	0.628	0.510	0.591
Tuned Random Forest	0.625	0.493	0.583
Ensemble tuned models with the statistical mode over the predictions	0.650	0.637	0.601

In general, the ML models here have a good accuracy but low precision and high recall, this means that many cases from the validation set are misclassified in different sentiment class to the correct one, hence and the low F1-score.

- *Machine Learning models and pre-trained word embeddings as feature extraction / representation*

In this experiment we combine Machine Learning models with pre-trained word embeddings for each word from the train set. The pre-trained word embeddings have been downloaded from Stanford NLP GloVe. Their performance results over the Validation and Test Set are the following:

ML Models	Accuracy over the Validation Set	F1-score over the Validation Set	Accuracy over the Test Set
Decision Tree	0.5041	0.171	0.521
Extra Tree	0.5053	0.175	0.523
Extra Trees	0.5055	0.176	0.521
Random Forest	0.5044	0.167	0.525

Here the ML models cannot cooperate well with pre-trained word embeddings, the accuracy and the F1-score are worse than before. The highlighted row is the model / outcome we can get from this experiment.

Deep Learning Learning Models evaluated over the Test Set from Kaggle:

The Deep Learning models that were developed along with 2 feature extraction techniques:

- Trainable Word Embeddings as feature extraction / representation
- Pre-trained word Embeddings as feature extraction / representations

- *Machine Learning models and Trainable Word Embeddings as feature extraction / representation:*

In this experiment we combine Deep Learning models with trainable word embeddings for each word from the train set. The Word Embeddings training took place via the Embedding Layers from Keras for each Deep Learning model. Their performance results over the Validation and Test Set are the following:

DL Models	Accuracy over the Validation Set	F1-score over the Validation Set	Accuracy over the Test Set
LSTM	0.676	0.669	0.644
Bidirectional_LSTM	0.653	0.636	0.645
CNN	0.667	0.657	0.632
LSTM_CNN	0.677	0.668	0.643
Bidirectional_LSTM_CNN	0.673	0.671	0.645
Ensemble tuned models with the statistical mode over the predictions	0.689	0.682	0.658

The Deep Learning Models here perform better than the Machine Leaning models, also LSTM model seem to be outperformed by the other DL models. Still the models suffer from low precision and high recall and thus the low accuracy and F1-score. The highlighted row is the model / outcome we can get from this experiment.

- *pre-trained word embeddings as feature extraction / representation:*

In this experiment we combine Deep Learning models with pre-trained word embeddings for each word from the train set. The Word Embeddings have been downloaded from Stanford NLP GloVe. Their performance results over the Validation and Test Set are the following:

DL Models	Accuracy over the Validation Set	F1-score over the Validation Set	Accuracy over the Test Set
LSTM	0.674	0.669	0.656
BiLSTM	0.678	0.674	0.658
CNN	0.682	0.675	0.657
LSTM_CNN	0.681	0.676	0.659
BiLSTM_CNN	0.685	0.678	0.660
Ensemble tuned models with the statistical mode over the predictions	0.696	0.691	0.674

The Deep Learning Models here with pre-trained word embeddings perform better than the previous experiment. Furthermore, the LSTM model seems to be outperformed by the other DL models. Still the models suffer from low precision and high recall and thus the low accuracy and F1-score. The highlighted row is the model / outcome we can get from this experiment.

Justification

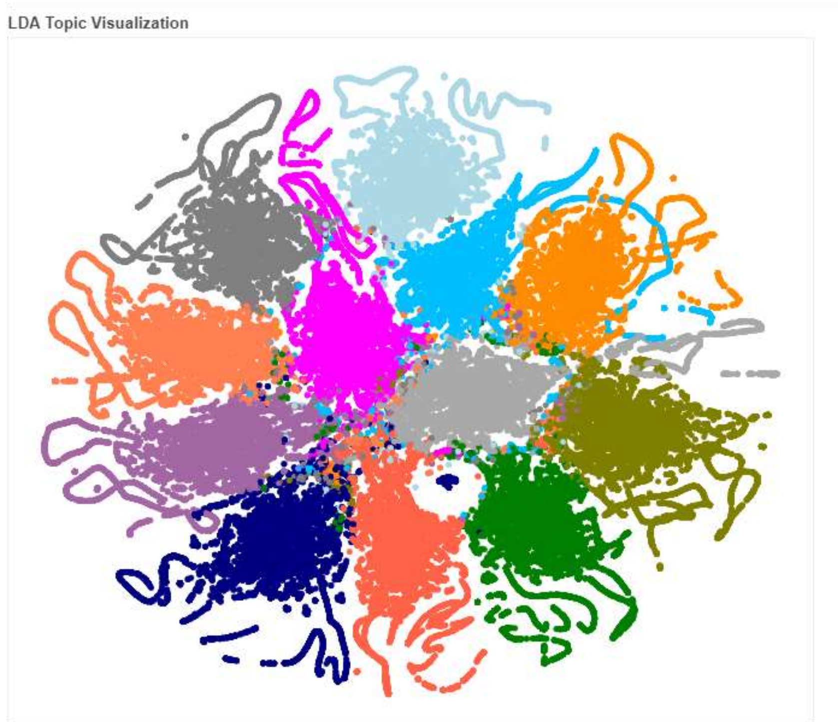
There is room for improvement on the final results, the tuned ML models with TF – IDF as feature extraction made no significant improvement over the untuned ML models. There could be more ways we could improve the accuracy and F1-score. Via Deep Learning models, as long as more deep learning complex and in-depth models are introduced, they will fit the dataset, although the exhaustive experiments that must be done with the current installment, we should not forget that the dataset is very strange, and the sentiments alter even with the absence of a single word or punctuation.

Conclusion

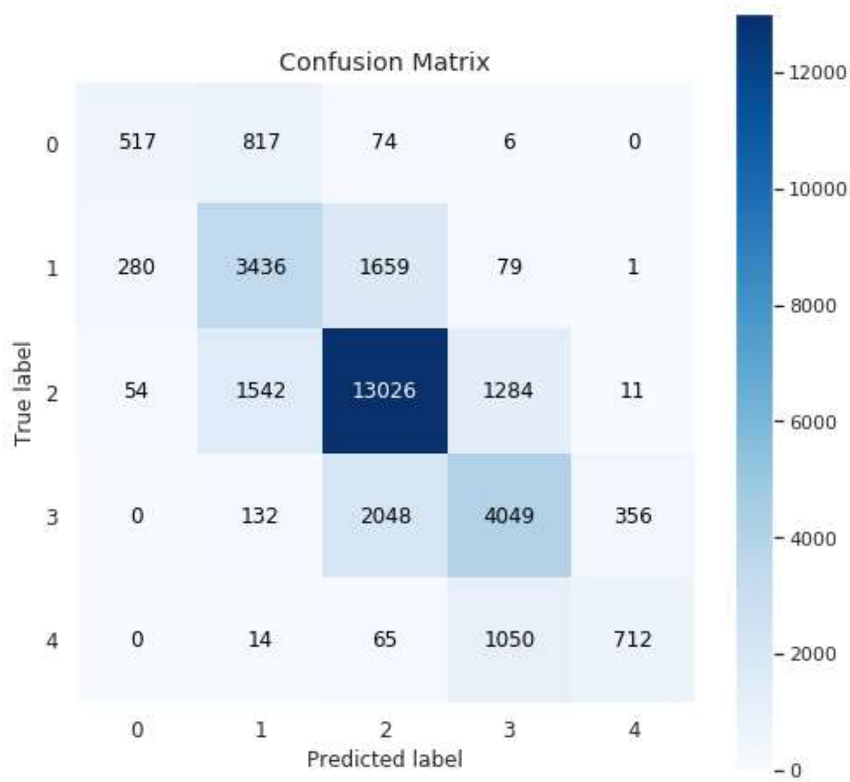
Free-Form Visualization

- One of the most satisfying visualization during EDA is the following it depicts all the most frequent words after text cleaning in the train set:

- Another beautiful visualization is the LDA topics via t-SNE, this illustration depicts the assignment of phrases in topics and t-SNE helps to reduce the dimensions to 2 axes:



- Finally, the last visualization is the fitting history and confusion matric for out best model which is the ensemble of Deep Learning models with pre-trained word embeddings as feature extraction / representation.



Reflection

- The most important and time-consuming part of the problem was data cleaning since there are many noisy data that must be cleaned. Once the data was prepared and ready, the next challenge is EDA and to focus on the word frequencies for unigrams, bigrams and trigrams. Data cleaning was a necessity for named entities extraction and identifying the most significant words in the trainset.

- During Unsupervised Learning it was time consuming to find the optimal number of clusters and the optimal number of topics in LDA. But it was satisfying to visualize them using t-SNE.
- During Machine Learning and Deep Learning models, it was unknown which model would fit the data with great accuracy and F1-score. Four experiments were made; 1 unsatisfactory and 3 successful experiments were achieved to improve accuracy and F1-score over the validation set.

Improvement

Deep Learning models show great potential and fit with great success the dataset. So, more in-depth deep learning models have to be developed. Tuning Deep Learning models is another way, however, it is very time consuming. Moreover, exhaustive Machine Learning tuning may be used. Furthermore, better and more innovative model ensemble techniques should be used. In addition, experimentation with text to feature extraction / representation. Finally another idea is to use and download other Word embeddings from other sources the Web.

References

[1] Mika V. Mäntylä Daniel Graziotin, Miikka Kuutila, The Evolution of Sentiment Analysis - A Review of Research Topics, Venues, and Top Cited Papers
[2] Qaiser, Shahzad & Ali, Ramsha. Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. International Journal of Computer Applications.
[3] Sklearn feature extraction - CountVectorizer
[4] Dixa Saxena, S. K. Saritha, K. N. S. S. V., Survey Paper on Feature Extraction Methods in Text Categorization
[5] Lei Zhang, Shuai Wang, Bing Liu, Deep Learning for Sentiment Analysis: A Survey
[6] Movie review sentiment analysis
[7] Xgboost top machine learning method kaggle explained
[8] Sentiment analysis using rnns lstm
[9] Occam's razor
[10] Movie review sentiment analysis - Evaluation
[11] TF - IDF
[12]TSNE
[13]TF - IDF sklearn vectorizer
[14]The two paths from natural language processing to artificial intelligence
[15]Kmeans vs LDA
[16]Perplexity to evaluate topic models
[17]Introduction to word embedding and word2vec
[18]Understanding LSTMs
[19]Introduction to sequence models rnn bidirectional rnn lstm gru
[20]A-Beginners-Guide-To-Understanding-Convolutional-Neural-Networks
[21]Understanding how convolutional neural network cnn perform text classification with word embeddings
[22]cLSTM, a combination of LSTM and CNN networks