# Keyphrase Extraction

Given a list of short text responses (1000-2000 responses) you are supposed to generate a set of key phrases which can be used to summarize/analyse the set of responses. Good key phrases would be those which can be assigned to multiple responses and which make sense. The key phrases can be generated by encoding and decoding the information in the responses or they can be taken directly from the responses.

Example:

| |
|---|
| I really like the show because it is thought provoking and i like shows that make me think |
| The cast. |
| Nothing |
| It is the most complex and original idea I have ever seen or heard of. Also, because it delves into the topic of human emotions, but in an artificial way, if I were to be punny and serious all at once. From what I have seen of the show, I believe that we, as human beings, can compare ourselves to the androids, because we can definitely relate to them. |
| it has many twists and I like that |
| It's unpredictable so you are left wanting more |

EXPECTED OUTPUTS / POSSIBLE KEYPHRASES:

1. Good/like the show/everything
2. Thought provoking
3. Complex storyline
4. Original
5. Suspense/Mystery/thrillers
6. Good/like the show/everything
7. Attention grabbing/holds interest/keeps you wanting to watch more

# Analysis

At first, I applied some basic data cleaning techniques on the data, such as:

- Make text all lower case
- Remove punctuation
- Remove numerical values
- Tokenize text
- Remove stop words
- Stemming / lemmatization

I also created a **word cloud** of the cleaned data for a better understanding of the word density in it. Then, I applied **part of speech tagging** to the data which showed the kind of words I was dealing with. Finally, evaluated the **most common words** (i.e. their frequencies).

There are many other techniques for data cleaning which can be used here.

Checkout the code file here : https://github.com/DD102K/Key_Phrase_Extraction/blob/main/Keyphrase%20Extraction_data_cleaning.ipynb

The task given lies under the category of text summarization. The process of writing a big piece of text in a short and concise manner stating the context and all the important information in the original text is called text summarization.

On the basis of the output, majorly, there are two methods of summarization:

1. Extraction Text Summarization

   This is what different researchers started with to find ways of automatic text summarization. Basically in this, the summarized data/ key phrases comes from the original text, as in exact sentences which seem important in context to the whole corpus are printed as the result.

2. Abstraction Text Summarization

This is a more advanced way as we encode the original text using which the model generates and outputs new phrases which summarizes the whole context of the feeded data.

## Approaches :

### Text summarization using TextRank Algorithm

It is based on the concept that words which occur more frequently are significant. We generate a cosine similarity matrix where we have the similarity of each sentence to each other. A graph is then generated from this cosine similarity matrix. We then apply the PageRank ranking algorithm to the graph to calculate scores for each sentence. The top-ranked sentences make it to the summary. It is an unsupervised graph based ranking algorithm. Going through the Github repositories, I realised that this is one of the widely used methods of text summarization.

Cosine Similarity - It measures the cosine of the angle between two vectors (sentence 1, sentence 2) projected in an N-dimensional vector space. Each sentence is converted into a vector. The smaller the angle, the similar the sentences.

PageRank algorithm - PageRank (PR) is an algorithm used by Google Search to rank web pages in their search engine results. It works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

Procedure :

We input data in a dataframe (created using pandas library) which is ordered by all the consecutive words present in the sentences being compared, the columns can include the sentence number (whether the word is in 1st or 2nd or nth sentence), lemmatized word, its part of speech tag. We can edit this dataframe as per our need, in a manner which feels more informative and useful.

(For keyword extraction) Then we can remove the stop/irrelevant words and calculate the frequency of each keyword.

(For sentence extraction) The algorithm basically computes weights between sentences by looking at which words are overlapping. It's on us what type of words we want the algorithm to check for overlapping, for ex. Nouns, adjectives etc or we can create our own list of words which are supposed to be compared.

Thus textrank then compares the common words in the sentences and ranks them as per it. Therefore, the top n sentences are then printed constituting the summary of the original text.

Advantages :

1. TextRank is completely unsupervised, and unlike other supervised systems, it relies exclusively on information drawn from the text itself,

which makes it easily portable to other text collections, domains, and languages. It does not require training corpora.

2. It gives a ranking over all sentences in a text – which means that it can be easily adapted to extracting very short summaries, or longer more explicative summaries, consisting of more than 100 words.
3. TextRank works well because it does not only rely on the local context of a text unit, but rather it takes into account information recursively drawn from the entire text.

Disadvantages :

1. The output summary has less semantics

References :

https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf
https://cran.r-project.org/web/packages/textrank/vignettes/textrank.html
https://towardsdatascience.com/understanding-nlp-word-embeddings-text-vectorization-1a23744f72

https://github.com/mzhao98/text-summarization

https://github.com/ceshine/textrank_demo

## Latent Semantic Analysis

Latent semantic analysis (LSA) is an unsupervised method for extracting a representation of text semantics based on observed words. The LSA method first builds a term-sentence matrix (n by m matrix), where each row corresponds to a word from the input (n words) and each column corresponds to a sentence (m sentences). . Each entry of the matrix is the weight of the word i in sentence j. The weights of the words are computed by TF IDF technique and if a sentence does not have a word the weight of that word in the sentence is zero. Then Singular Value Decomposition (SVD) is used on the matrix and transforms the matrix A into three matrices: $A = U \Sigma V^T$ . Matrix U ($n \times m$) represents a term-topic matrix having weights of words. Matrix Σ is a diagonal matrix ($m \times m$) where each row i corresponds to the weight of a topic i. Matrix $V^T$ is the topic sentence matrix. The matrix $D = \Sigma V^T$ describes how much a sentence represents a topic. The mathematical technique called singular value decomposition is used to reduce the number of columns while preserving the similarity structure among rows. Words are then compared by taking the cosine of the angle between the two vectors formed by any two rows. Values close to 1 represent similar words and values close to 0 represent dissimilar words. LSA gets this name because Singular Value Decomposition applied to document word matrices, group documents that are semantically related to each other, even when they do not share common words.

It extracts the meaning of words and meaning of sentences. It assumes that words that are close in meaning will occur in similar pieces of text. The word co-occurs in sentences that mean that sentences are semantically related and high chances to include in the summary.

Procedure :

We will use the gensim library for the LSA model. So, input the data and clean it. For example tokenize, lemmatize etc the data. Then we create the term matrix containing the word frequencies. Apply the LSA model from gensim on the term matrix. Sort the output documents according to the weightage. Select the top documents and then select the top sentences from the sorted list of sentences in from them. The top sentences constitute the summary.

Advantages :

1. The main advantage of using LSA vectors for summarization rather than the word vectors is that conceptual (or semantic) relations as represented in the human brain are automatically captured in the LSA.
2. Reduce the dimensionality of the original text-based dataset.
3. It helps us understand what each topic is encoding.
4. Analyze word association in text corpus

Disadvantages :

1. This strategy has a drawback due to the fact that a topic may need more than one sentence to convey its information. For improvement, One enhancement is to leverage the weight of each topic to decide the relative size of the summary that should cover the topic, which gives the flexibility of having a variable number of sentences.
2. If the text is inhomogeneous, the calculation is complex.

3. The modified tf-idf approach lacks performance because it removes some of the sentences/words from the input matrix, assuming that they cause noise.

References :

http://lsa.colorado.edu/papers/JASIS.lsi.90.pdf

http://www.iaeme.com/MasterAdmin/UploadFolder/IJCET_09_01_004/IJCET_09_01_004.pdf

https://towardsdatascience.com/document-summarization-using-latent-semantic-indexing-b747ef2d2af6#:~:text=Latent%20Semantic%20indexing%20is%20an,sorted%20according%20to%20sematic%20similarity.

https://github.com/luisfredgs/LSA-Text-Summarization

My choice of approach :

I will choose TextRank Algorithm for text summarization as it gives better results. LSA outputs less number of words/sentences. Zero semantics are being shown in the summary. TextRank also has this problem, but the magnitude is less.

There are alot of new models involving deep learning models, seq2seq models, transformer library with pretrained models such as BART etc which are new.

For a better comparison, we can see on the website that Textrank algorithm is better than LSA : https://www.machinelearningplus.com/nlp/text-summarization-approaches-nlp-example/

**Thank You!**