**Extractive method** :

* For extractive summarization I’ve used Text Rank Algorithm which primarily tries to find the importance of vertex in a given graph.
* Here each sentence is considered as a vertex.
* If two sentences are joined by an edge it means they are similar, they only have an edge if the two sentences represent similarity more than a certain threshold.
* **Keyword extraction** is done using text rank algorithm, where the given text is first tokenized and parts of speech are tagged, then single words are added to the word graph as nodes. Then similarity is measured using the co-occurrence of words.
* Here we do **sentence extraction task** which is much similar to keyword extraction where entire sentences are represented as nodes and then similarity is found.
* We first get all the sentences from the article then we build the similarity matrix.
* After that we find the page rank from the similarity matrix using the method mentioned in this [paper](https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf).
* At last we concatenate the ranked sentences to get our extractive summary.

**Abstractive method:**

* My approach to solve summarization for abstractive method would be using encoders and decoders networks.
* First we’ll clean and create our dataset by parsing and separating text and summary from story dataset.
* Our dataset will have long text and there summary. The information can be used as labels and targets for training our encoder-decoder networks. We will vectorize our labels and targets, form a vocabulary.
* Next, we will pick the embeddings up from word2vec or Glove for the words in our vocabulary and then fit the labels and targets to our model for training.
* Words predicted by the decoder are generated using a softmax.
* The model is trained in mini batches because we can’t use all the words in our vocabulary as our target word set as the softmax will have a huge number of output nodes, and the prediction will be computationally inefficient.