Text Summarization

Extractive Based Approach

Workflow

Preprocessing

Feature Selection

Similarity measures

Content selection

PREPROCESSING

Sentence Segmentation: Splitting a text into sentences using boundary identification algorithm

Word Segmentation: Splitting a sentence into words based on spaces and punctuation

StopWord Removal: Remove word like ‘The’, ‘a’

Stemming: A rule based stemming algorithm, the Porter stemmer

Tokenization

Tokenization is the process by which big quantity of text is divided into smaller parts called **tokens**.

- word tokenization
- sentence tokenization

```
text = "God is Great! I won a lottery."

Word_tokenize: ['God', 'is', 'Great', '!', 'I', 'won', 'a', 'lottery', '.']

Sentence_tokenize: ['God is Great!', 'I won a lottery ']
```

StopWords

Stop Words: A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) . NLTK(Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages.

We basically append the word that are not in nltk list so that we can obtain the result without any stopwords.

Stemming

The porter stemming is a process for removing suffixes from words in English.

Example: Connect, connected, connecting,connection

This is done by removal of -ed, -int, -ion, -ions to leave single term connect.

Feature Selection

Features are obtained by selecting unique words from the preprocessed sentences

Each sentence is represented as a context vector

The vectors are accumulated into a representation matrix

Bags of Words

In BoW model a sentence or a document is considered as a 'Bag' containing words.

Example: D1 - “I am feeling very happy today”
D2 - “I am not well today”
D3 - “I wish I could go to play”

	I	am	feeling	very	happy	today	not	well	wish	could	go	to	play
D1	1	1	1	1	1	1	0	0	0	0	0	0	0
D2	1	1	0	0	0	1	1	1	0	0	0	0	0
D3	2	0	0	0	0	0	0	0	1	1	1	1	1

TF-IDF

Term Count (TC) Represents sentences as vectors of which elements are the absolute frequency of words in the sentence.

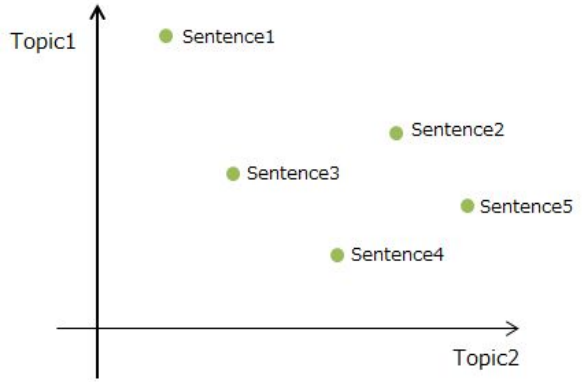
TF = number of times the term appears in the doc/total number of words in the doc
IDF = ln(number of docs/number docs the term appears in)
Higher the TF-IDF score, the rarer the term is and vice-versa.

	W1	w2	w3	w4	w5	w6
S1	0.23	0	0	0	0.46	0.23
S2	0.23	0.23	0.46	0	0	0
S3	0	0.35	0	0.35	0	0
S4	0	0	0	0.35	0	0.35

TF-IDF representation Matrix

LSA (Latent Semantic Analysis)

LSA (Latent Semantic Analysis) also known as LSI (Latent Semantic Index) LSA uses bag of word(BoW) model, which results in a term-document matrix(occurrence of terms in a document). Rows represent terms and columns represent documents. LSA learns latent topics by performing a matrix decomposition on the document-term matrix using Singular value decomposition. LSA is typically used as a dimension reduction or noise reducing technique.



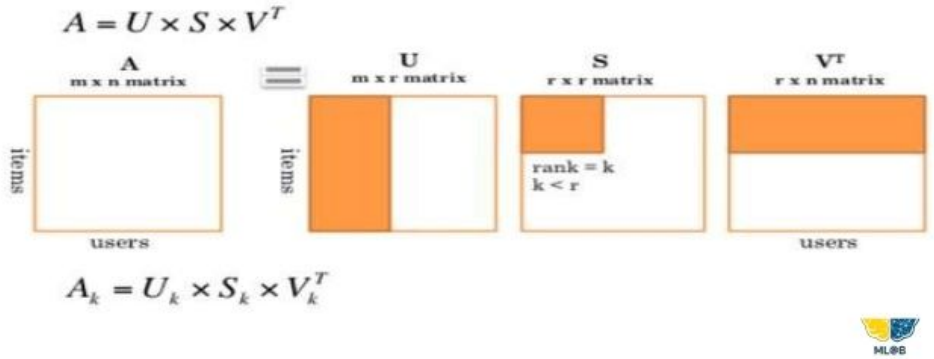
	Topic1	Topic2
Sent1	0.9	0.25
Sent2	0.7	0.7
Sent3	0.45	0.4
Sent4	0.3	0.6
Sent5	0.4	0.9

LSA

LSA Algorithm

1. Creating the Count Matrix
2. Modify the Counts with TFIDF
3. Using the Singular Value Decomposition
   A=U*S*(VT)

4. Sentence Selection for summary.



Pros and cons

LSA algorithm is faster compared to other available algorithms because it involves document term matrix decomposition only.

Latent topic dimension depends upon the rank of the matrix so we can't extend that limit. LSA decomposed matrix is a highly dense matrix, so it is difficult to index individual dimension. LSA unable to capture the multiple meanings of words.

TextRank algorithm

The summarising is based on ranks of text sentences using a variation of the PageRank algorithm.
TextRank is a general purpose, **graph based** ranking algorithm for NLP.

Graph-based ranking algorithms are a way for deciding the importance of a vertex within a graph, based on global information recursively drawn from the entire graph.

$$PageRank\ of\ site = \sum \frac{PageRank\ of\ inbound\ link}{Number\ of\ links\ on\ that\ page}$$

OR

$$PR(u) = (1 - d) + d \times \sum \frac{PR(v)}{N(v)}$$

- The first step would be to concatenate all the text contained in the articles
- Then split the text into individual sentences
- In the next step, we will find vector representation (word embeddings) for each and every sentence
- Similarities between sentence vectors are then calculated and stored in a matrix
- The similarity matrix is then converted into a graph, with sentences as vertices and similarity scores as edges, for sentence rank calculation
- Finally, a certain number of top-ranked sentences form the final summary

Page	Link to
A	B, C, D, E
B	C, E
C	D
D	B, D, E
E	A, C

	A	B	C	D	E
A		1	1	1	1
B			1		1
C				1	
D		1		1	1
E	1		1		

Transition Matrix

	A	B	C	D	E
A		1/4	1/4	1/4	1/4
B			1/2		1/2
C				1	
D		1/3		1/3	1/3
E	1/2		1/2		

Transition Probability Matrix (=M)

Page Surfer	
A	0.2
B	0.1
C	0.3
D	0.1
E	0.3

1

Page Surfer State (=P)

MP = P when t -> ∞

Solution

1. Solve Eigenvalue Problem of **MP = P**.
2. Repeat the transition until convergence(**MP - P < threshold**).

$$P_i' = (1-d) + d * M_i^T P_i$$

$$\sum (P_i' - P_i) < threshold$$

*The page surfers randomly click the page with a probability of **1-d**. (d = usually 0.85)*

LEXRANK Algorithm

This is same as Textrank But the algorithm for calculating pagerank is different

$$P(U) = \frac{d}{N} + (1-d) \sum_{V \in adj[U]} \frac{idf\_modified\_cosine(U,V)}{idf\_modified\_cosine(Z,V)} \times P(V)$$

NOTE: The "ifd modified cosine similarity" can be calculated by using the following equation:

$$\frac{\sum_{W \in X,Y} tf_{W,X}\, tf_{W,Y}\, (idf)^2}{\sqrt{\sum_{X_i \in X} (tf_{X_i,X}\, idf_{X_i})^2} \times \sqrt{\sum_{Y_i \in Y} (tf_{Y_i,Y}\, idf_{Y_i})^2}}$$

Pros and cons

This algorithm generates the summary that are short and more compressed in nature.This algorithm primarily uses the similar types of sentence

It discard lower-ranked sentences that are too similar to higher-ranked ones

Sentence Scoring based on Word Frequency

Here we assign weights to each word based on the frequency of the word in the passage. For example, if “Soccer” occurs 4 times within the passage, it will have a weight of 4.

Using the weights assigned to each word, we will create a score for each sentence. In the end, we will be taking the score of the top `N` sentences for the summary.

Using Skip-Thought Encoder

Here the sentence are encoded into Skip-Thought Vector.
Then encoded sentences are clustered.
The sentence corresponding to sentence embedding closest to the cluster centers are chosen as the summary