NLP Report

# Summarization Technologies:

There are two common approaches to solve the summarization task. The first approach tries to analyze the text, and to rewrite or rephrase it in a short way. To the best of my literature survey, until today this approach didn’t achieve any substantial results. The second approach tries to extract the key sentences from the text, and then tries to put them together properly. Famous famous algorithms that implements this approach are TextRank and Lexical Chains.

There cannot be a single best algorithm for summarization of all kinds of genre. Since summarization is about scoring each sentence from the text and then creating summary from those scored sentences, to get a good summary you need to have different scoring algorithm for different genres.

# Naive Summarization Algorithm:

Our summarization algorithm constitutes of following steps.

## Intersection Function:

This function receives two sentences, and returns a score for the intersection between them.

We just split each sentence into words/tokens, count how many common tokens we have, and then we normalize the result with the average length of the two sentences.

Therefore,

IntersectionFunction (s1, s2) = | {w | w in s1 and w in s2}| / ((|s1| + |s2|) / 2),

where s1 and s2 are sentences, w represents word, |s1| is length of sentence1, |s2| is length of sentence2.

## Sentence Dictionary:

It receives our text as input, and calculates a score for each sentence. The calculations is composed of two steps:

In the first step we split the text into sentences, and store the intersection value between each two sentences in a matrix (two-dimensional array). So matrix[i][j] will hold the intersection score between sentence #i+1 and sentence #j+1.

So, we just converted our text into a fully-connected weighted graph! Each sentence is a node in the graph and the two-dimensional array holds the weight of each edge!

In the second step we calculate an individual score for each sentence and store it in a key-value dictionary, where the sentence itself is the key and the value is the total score. We do that just by summing up all its intersections with the other sentences in the text (not including itself). (Technically, we calculates the score for each node in our graph. We simply do that by summing all the edges that are connected to the node.)

## Building the summary:

The final step of our algorithm is generating the final summary. We do that by splitting our text into paragraphs, and then we choose the best sentence from each paragraph according to our sentences dictionary.

The Idea here is that every paragraph in the text represents some logical subset of our graph, so we just pick the most valuable node from each subset. So this is a way is similar to identifying the most weighted tree in a connected graph.

## Why this algorithm works?

If one sentence has a good intersection with many other sentences, it probably holds some information from each one of them, or in other words, this is probably a key sentence in our text!

## How to make this algorithm work even more efficient?

We haven’t used any NLP techniques such as tokenization and lemmatization. They help in identifying the relation among words and hence if two words are similar, the score calculation logic can be improvised.

In intersection function, we can use list of *stopwords* to prune certain words and according to the domain we are working, we can assign weights to words.

Instead of picking only one best sentence from each paragraph, we can pickup ‘n’ number of sentences depending upon the sentence length.

# References:

<http://web.stanford.edu/class/cs276b/handouts/lecture14.pdf>

<https://www.cs.cmu.edu/~afm/Home_files/Das_Martins_survey_summarization.pdf>

<https://thetokenizer.com/2013/04/28/build-your-own-summary-tool/>

<https://www.uleth.ca/dspace/bitstream/handle/10133/226/MR03038.pdf?sequence=3>