
# Presentation on
# Text summarization using NLP

**Presented by:-**
**MD.Liaket Hossain (161412313)**
**Saiful Alam (161-41-2327)**
**Nazrul islam(161412329)**
**Mustafizur Rahman(161412318)**

# Introduction

The goal of summarization is to produce a shorter version of a source text by preserving the meaning and the key contents of the original document.

A well written summary can significantly reduce the amount of work needed to digest large amounts of text.

# Text Summarization

Basically, we can regard the “summarization” as the “function” its input is text and out put is summary.

And its input & output type helps us to categorize the multiple summarization tasks

**Single Text summarization:-**

[ *summary = summarize(text]*

**Multiple Text summarization:-**

[ *summary = summarize(text_1,text_2…)]*

## Update summarization :-

*summary = summarize(text , previous_text_or_summary)*
And the *“summary”* itself has some variety.
**Informative summary:-**

In contrast to the indicative summary, the informative summary includes full information of the text
**Keyword summary:**

Not the text, but the words or phrases from the input document
.
**Headline summary:-**

Only one line summary.

## Types of Text summarization

Extractive summarizes

Abstractive summarizes

# Extractive summaries

- Extractive summaries are created by reusing portions (words, sentences, etc.) of the input text document
- The system extracts text from the entire collection, without modifying the text document.
- Most of the summarization research today is on extractive summarization.

# Abstractive summaries

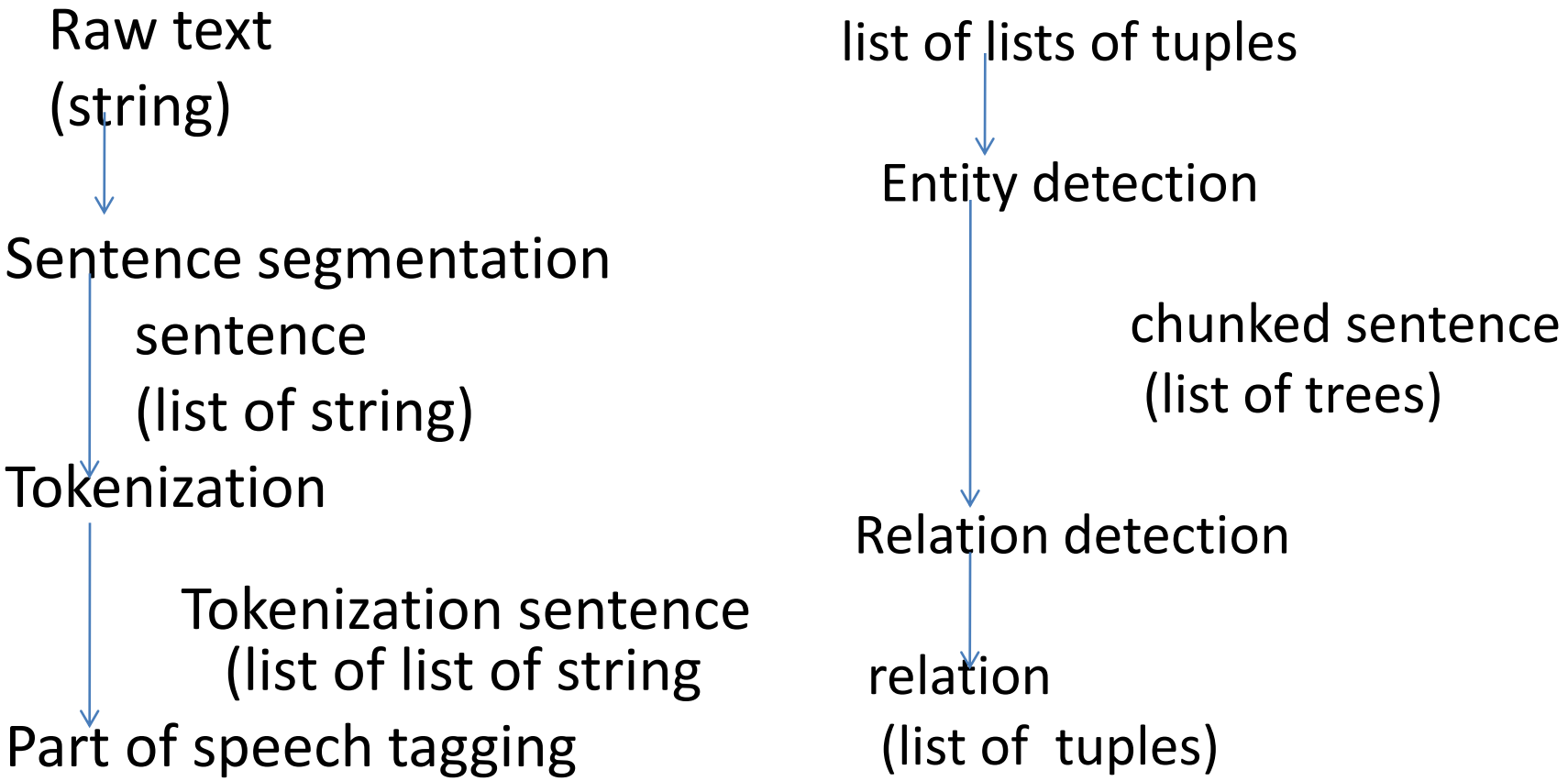
- Requires deep understanding and reasoning over the text
- It Provides own summary over input text without using same word or sentence in the input text
- Determines the actual and short meaning of each element, such as words,sentences and paragraphs

# Continued…

Following are the NLTK Libraries used in text summarization

- Word tokenizer
- Sentence tokenizer
- stopwords
- BeautifulSoup
- numpy library
- Tagging
- Parsing

# Linguistic preprocessing for Automatic summarization



# Sentence Segmentation

- Converts raw text into sentences
- List of strings
- Sentence tokenizer

Input Text: John owns a car. It is a Toyota.
Output: Segm1: John owns a car.
Segm2: It is a Toyota.

# Tokenization

- Identifies the word tokens from given sentence
- Provides a list of tokens as output
- Word tokenizer

Input: John owns a car.
Output: [[John], [owns], [a], [car], [.]]

# Part of speech tagging (POS Tagging)

- Assigns appropriate part of speech tag to each word
- POS is useful in extraction of nouns, adverbs, adjective, which provide some meaningful information about text
- Generates a list of tuples with POS annotation
-

Input: [[John], [owns], [a], [car], [.]]
Output: (NP (NNP John)) (VP (VBZ owns) (NP (DT a) (NN car))) (. .)

# Entity detection

- Identification of predefined categories such as person, location, quantities, organizations etc
- NER provides the entity detection for linguistic processing
- NER system uses linguistic grammar-based techniques and also statistical model to identify the entity

Input: (NP (NNP John)) (VP (VBZ owns) (NP (DT a) (NN car))) (. .)
Output: John->Person

# Relation detection

- Identifies the possible relation between two or more chunked sentences
- Co-reference chain provides a relation between two or more sentences
- Provides the link between pronouns and its corresponding nouns
- Replacement of the pronouns with proper nouns

Input Text: John owns a car. It is a Toyota. (In form of parse tree)
Output: "a car" -> "a Toyota"; "It" -> "a Toyota"

# Conclusion

- Automatic Text Summarization has been shown to be useful for Natural Language Processing tasks such as Question Answering or Text Classification and other related fields of computer science such as Information Retrieval. And the access time for information searching will be improved.