Amrutvahini College of Engineering, Sangamner

Department of Computer Engineering

Report of inauguration of Natural Language Processing club

Date: 03.01.2020 **Time: 4.00pm**

Department of Computer Engineering formally inaugurated NLP club on 3.1.20 at 4.00pm. Idea behind this club was to give a platform to students to work & explore in the field of Natural language processing.

Total 35 students from SE & TE participated in this NLP club. Our alumni of 2017 batch, Mr. Prasad Kharde (Oracle Pvt Ltd, Banglore) is going to help students for this activity.

Principal sir inaugurated the club by cutting the ribbon. Session was started by briefing about NLP by Ms. S. B. Bhonde & then Principal sir addressed students on importance of this club. Further, motivational speech was given by Prof. R. L. Paikrao on what are current applications of NLP.

- Few snaps taken during inaugural -

The goal of natural language processing (NLP) is to design and build computer systems that are able to analyze natural languages like English, Marathi, German or English. Full NLP system must perform following task:

- Paraphrase an input text.
- Translate the text into another language.
- Answer questions about the contents of the text.
- Draw inferences from the text.

At end, Prasad Kharde shared his view about this club & motivated more & more students to participate in this club. He also created one slack channel for smooth communication & execution of various activities under this club.

Coordinators

1. Ms. K. U. Rahane -
2. Ms. S. B. Bhonde -

Head of Department

Prof. R. L. Paikrao

Amrutvahini College of Engineering, Sangamner

Department of Computer Engineering

Report of activities under Natural Language Processing club

Date: 28.01.2020 **Time: 4.00pm**

Topics Covered : Pre-processing, tokenization, stemming, lemmatization

Ms. S. B. Bhonde Ms. K. U. Rahane Prof. R. L. Paikrao

Amrutvahini College of Engineering, Sangamner

Department of Computer Engineering

Report of activities under Natural Language Processing club

Date: 31.01.2020 **Time: 4.00pm**

Topics Covered :

- Possible project topics in NLP

Version 1 :
Your input would be list of words, their length character wise, what you want to search for - like synonyms or antonyms or any relationship of word to find with this listed word Your output should be the word which fulfills all the criteria. Note this is without any interdependency so expect to have lot of ambiguous output data and your code should handle this.

Version 2 :
Your input would be list of words, their length character wise, what you want to search for - like synonyms or antonyms or any relationship of word to find with this

listed word dependency on other words like word 2 of length 5 characters which is opposite of hate, start from 3rd character of word 1. (like a real crossword example in newspaper)
Model your input datasets or whatever the model structures you want to use as per your convenience. Your output should be the word which fulfills all the criteria.
Note this is with dependency on the other words.

Version 1 and 2 you can implement without any document restriction so we are considering all the words from the dictionary as a possible candidate for output word.

Version 3 :
Implement the version 2 model on a document only. You can take any document from Wikipedia page or any chapter from the novel ebook.

You can start with 2 approach :
Approach 1: start with english dictionary to find the output candidate.

Approach 2:
Construct a question from the input parameters.

- Write a crawler logic.
- Search the question on google using google API.
- Go through the crawled pages using basic NLP.
- Find correct answers and replace the candidate.

Approach 2 is good only in case of version 1 and version 2.
Or you can go ahead with any other approach you find comfortable.

*You cannot use spacy, NLTK, openNLP, word2vec, GLoVE libraries for any NLP **operations.***

- You can search for third party libraries but it'll be better if you write your own logic for these tasks.
- You are not supposed to use different approaches in same version.
- Your code should solve the all cases and should not contain hardcoded code modules for some type of input. You can create submodules for synonyms, antonyms etc. but not for cases including for specific word find specific result.
- You don't need to create any user interface for this.
- You can use any language you want - java, c, cpp, R, python, scala, go anything