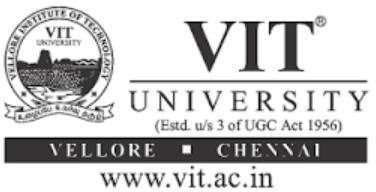
**“Key Phrase Extraction”**

**PROJECT REPORT**

Submitted for the course: Natural Language Processing (CSE4022)



By

**VARUN REDDY (16BCE0040)**

**APOORV TYAGI (16BCE0097)**

**ADITYA RANJAN(16BCE2207)**

Slot: G1

**Name of Faculty: SHARMILA BANU K**

**(SCHOOL OF COMPUTER SCIENCE AND ENGINEERING)**

**Abstract**

Phrase extractor is of significant importance as once we have [identified](https://www.searchtechnologies.com/blog/web-data-mining-tools-techniques), [extracted](https://www.searchtechnologies.com/blog/web-content-extraction-data-mining), and [cleansed](https://www.searchtechnologies.com/blog/data-cleansing-techniques-data-mining) the content needed for our use case, the next step is to have an understanding of that content. In many use cases, the content with the most important information is written down in a natural language (such as English, German, Spanish, Chinese, etc.) and not conveniently tagged. To extract information from this content we need to rely on some levels of text mining, text extraction, or possibly full-up natural language processing (NLP) techniques. Extracts sequences of tokens (phrases) that have a strong meaning which is independent of the words when treated separately. These sequences should be treated as a single unit when doing NLP. For example, “Big Data” has a strong meaning which is independent of the words “big” and “data” when used separately. All companies have these sorts of phrases which are in common usage throughout the organization and are better treated as a unit rather than separately.

Contents

[1. INTRODUCTION 4](#_Toc4707099)

[2. LITERATURE SURVEY 4](#_Toc4707100)

[2.1. SURVEY-1 4](#_Toc4707101)

[2.2. SURVEY-2 5](#_Toc4707102)

[2.3. SURVEY-3 5](#_Toc4707103)

[2.4. SURVEY-4 5](#_Toc4707104)

[3. OVERVIEW OF THE WORK 6](#_Toc4707106)

[3.1. REQUIRED MODULE 6](#_Toc4707107)

[3.2. SOFTWARE REQUIREMENT 6](#_Toc4707108)

[3.3. HARDWARE REQUIREMENT 6](#_Toc4707109)

[4. SYSTEM DESIGN 7](#_Toc4707110)

[5. IMPLEMENTATION 11](#_Toc4707118)

[6. CONCLUSION 13](#_Toc4707125)

[7. REFRENCES 14](#_Toc4707126)

# INTRODUCTION

This is a project on Natural Language processing course where a given training text file with phrases are provided with which model is trained and evaluated. Later the trained model is used to predict the phrases for the test file. Most of the text available on internet/online websites is simply a string of characters. Such texts are useless to apply the tools of Natural Language on. Hence, the primary step involves cleaning the input text so that processing can be done in later phases.

To achieve this, we have used the [Natural Language Toolkit (NLTK)](https://en.wikipedia.org/wiki/Natural_Language_Toolkit) which is a popular platform for building python programs to work with human language data. The NLTK provides most of the tools that is required for text cleaning and processing.

# LITERATURE SURVEY

## SURVEY-1

In article “NLP for Term Variant Extraction: Synergy Between Morphology, Lexicon, and Syntax” by Christian Jacquemin and Evelyn Tzoukermann we come across a natural language processing (NLP) approach to automatic indexing over controlled vocabulary which accounts for term variation. The approach combines a part of speech tagger, a generator of morphologically related forms, and a shallow transformational parser. The system is applied to the French language; it is trained on newspaper articles and tested on scientific literature.

Precision rate of indexing on term and variants is 97.2%. It is only slightly lower than indexing without accounting for term variation (99.7%). Recall rate of indexing on term and variants (93.4%) is much higher than recall of indexing on term occurrences only (72.4%). Conflation of term variants increases indexing coverage up to 30%.The system is a convincing example of the potential synergy between full-fledged morphological analysis and local syntactic analysis. Many details are provided on the implementation of the system. Illustrative examples of syntactic transformations for the French language are given together with the theoretical and empirical methods for their formulation.

## SURVEY-2

“**Empirical method of information extraction**” by Claire Cardie surveys the use of empirical, machine-learning methods for a particular natural language-understanding task-information extraction. The author presents a generic architecture for information-extraction systems and then surveys the learning algorithms that have been developed to address the problems of accuracy, portability, and knowledge acquisition for each component of the architecture.

## SURVEY-3

# “Semantic NLP-Based Information Extraction from Construction Regulatory Documents for Automated Compliance Checking” by [Jiansong Zhang](https://ascelibrary.org/author/Zhang,+Jiansong) and [Nora M. El-Gohary](https://ascelibrary.org/author/El-Gohary,+Nora+M) gibes us an Automated regulatory compliance checking requires automated extraction of requirements from regulatory textual documents and their formalization in a computer-processable rule representation. Such information extraction (IE) is a challenging task that requires complex analysis and processing of text. Natural language processing (NLP) aims to enable computers to process natural language text in a human-like manner. This paper proposes a semantic, rule-based NLP approach for automated IE from construction regulatory documents. The proposed approach uses a set of pattern-matching-based IE rules and conflict resolution (CR) rules in IE.

## SURVEY-4

“**Extracting Phenotypic Information from the Literature via Natural Language Processing**” by Lifeng Chen, Carol Friedman talks about the amount of biomedical knowledge has been increasing exponentially. Several Natural Language Processing (NLP) systems have been developed to help researchers extract, encode and organize new information automatically from textual literature or narrative reports. Some of these systems focus on extracting biological entities or molecular interactions while others retrieve and encode clinical information. To exploit gene functions in the post-genome era, it is necessary to extract phenotypic information automatically from the literature as well. However, few NLP projects have focused on this.

# OVERVIEW OF THE WORK

## REQUIRED MODULES

* Numpy
* Sklearn
* Pandas
* Pickle
* Nltk
* Collections
* Beautiful soup
* Urllib
* lxml

## SOFTWARE REQUIREMENT

* Operating system : Windows XP OR ABOVE.
* Coding Language : PYTHON
* Tool : JUPYTER NOTEBOOK

## HARDWARE REQUIREMENT

* System : i5-6200U CPU @ 2.30GHZ OR ABOVE
* Hard Disk : 4 GB OR ABOVE.
* Ram : 512 Mb OR ABOVE.

# SYSTEM DESIGN

## BASIC METHODOLOGY

The given eval\_data.txt is fed to regex\_generator.py, it generates a regex pattern file with one word previous and after of the label then the other pattern is with two words before the label. Then the input data is fed to SVM architecture and MLP with 100 layers that trains the model which gives us two classes

* Found
* Not Found

Then test data is fed to the trained models to predict the classes weather the phrases are found or not, if the phrases are found then the regex pattern is used to detect the phrase.

This mechanism helps us to do a meaningful “Phrase Extraction” as the model is trained with the mentioned 100 layers to give us a meaningful result in contrast to simple Phrase extraction which may result in unwanted phrases or words that may lead to anomalous results.

## CODE

**from** **sklearn.feature\_extraction.text** **import** TfidfTransformer **import** **os** **import** **pickle** **import** **pandas** **as** **pd** **import** **numpy** **as** **np** path\_training\_data = os.getcwd()+'/training\_data.tsv' path\_test\_data = os.getcwd()+'/eval\_data.txt'

**import** **regex\_generator**

regex\_generator.main\_regex\_generator(path\_training\_data)*####### regex extraction ######*

**import** **regex\_matcher**

print('inside\_extration')extractions\_from\_regex = regex\_matcher.main\_regex\_matcher(path\_test\_data)print('extracted')

**import** **training\_classifier**training\_classifier.main\_ML\_model()

f = open('svm\_classifier.pickle', 'rb')clf = pickle.load(f)f.close()

f = open('mlp\_classifier.pickle', 'rb')mlp = pickle.load(f)f.close()

f = open('my\_vectorizer.pickle', 'rb')vectorizer = pickle.load(f)f.close()

*###### loading data ######*data=[]master\_data = open(path\_test\_data, 'r')**for** sent **in** master\_data:

data.append(' '.join(sent.split()))*#using ' '.join(sent.split()) to remove the \n from the txt file*

df = pd.DataFrame(data) *# loading data to the dataframe*df=df[0]*# getting the dimentions right for prediction#*clean\_test\_data = []**for** i **in** df:

clean\_test\_data.append(i)*# print (clean\_train\_data)*

*# Get a bag of words for the test set, and convert to a numpy array*test\_data\_features = vectorizer.transform(clean\_test\_data)np.asarray(test\_data\_features)

tfidf\_transformer = TfidfTransformer()X\_test\_tfidf = tfidf\_transformer.fit\_transform(test\_data\_features)

*###### predicting from the classifier ######*

result1 = clf.predict(X\_test\_tfidf)result2 = mlp.predict(X\_test\_tfidf)

*###### saving the result in a dataframe ######*

output1 = pd.DataFrame(data={"sent": df, "label": result1})output2 = pd.DataFrame(data={"sent": df, "label": result2})predicted\_result1 = list(result1) *# converting the n dimentional array into a list*predicted\_result2 = list(result2)*###### output of the classisfier using BOW model ######*output1.to\_csv(('Bag\_of\_Words\_model\_new\_svm.csv'), index=**False**, quoting=3, escapechar='**\\**')output2.to\_csv(('Bag\_of\_Words\_model\_new\_mlp.csv'), index=**False**, quoting=3, escapechar='**\\**')

*###### final submission ######*

final\_result1=[]**for** idx, prediction **in** enumerate(predicted\_result1):

**if** (prediction=='Not Found'):

final\_result1.append('Not Found') *# if classifier identifies the sentence to be labled as Not Found it has power to over write the extracter*

**else**:

final\_result1.append(extractions\_from\_regex[idx])*# if the classifier identifies it to have a phrase then we use the extracted phrase*

final\_result2=[]**for** idx, prediction **in** enumerate(predicted\_result2):

**if** (prediction=='Not Found'):

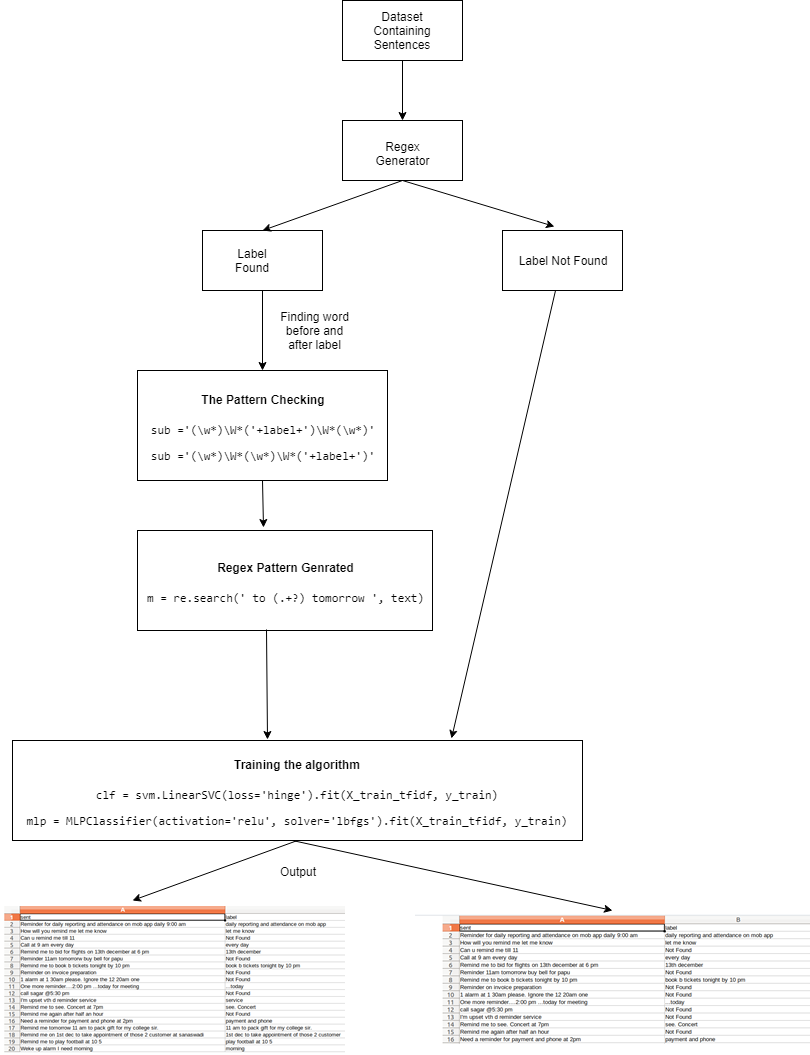
final\_result2.append('Not Found') *# if classifier identifies the sentence to be labled as Not Found it has power to over write the extracter*

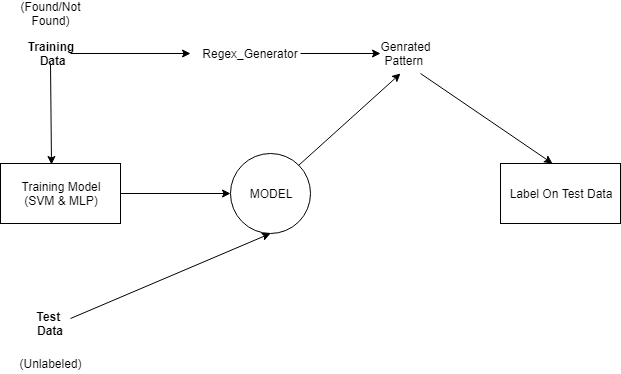
**else**:

final\_result2.append(extractions\_from\_regex[idx])

*###### loading the final output into df ######*final\_output1 = pd.DataFrame(data={"sent": df, "label": final\_result1})final\_output2 = pd.DataFrame(data={"sent": df, "label": final\_result2})*###### Final submission ######*final\_output1.to\_csv(('submission\_svm.csv'), index=**False**, quoting=3, escapechar='**\\**')final\_output2.to\_csv(('submission\_mlp.csv'), index=**False**, quoting=3, escapechar='**\\**')print("Output stored")

## 4.3 FLOWCHAT

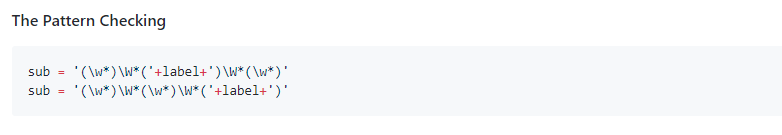


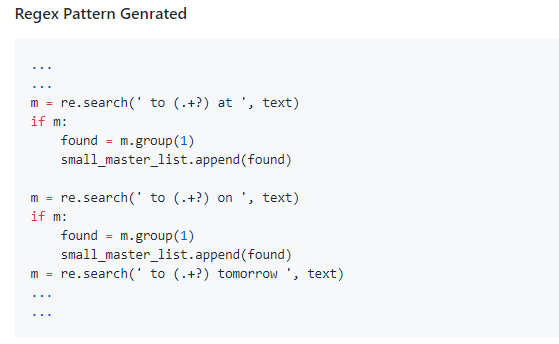


# IMPLEMENTATION ([Github Link](https://github.com/ApoorvTyagi/Phrase_Extracter))

## CODE SNIPPETS

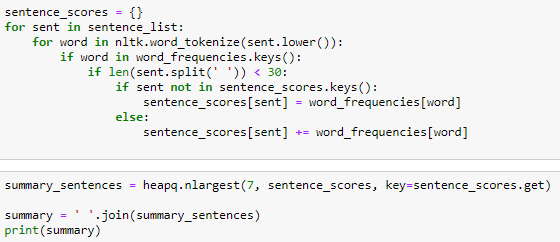
* Finding word before and after label





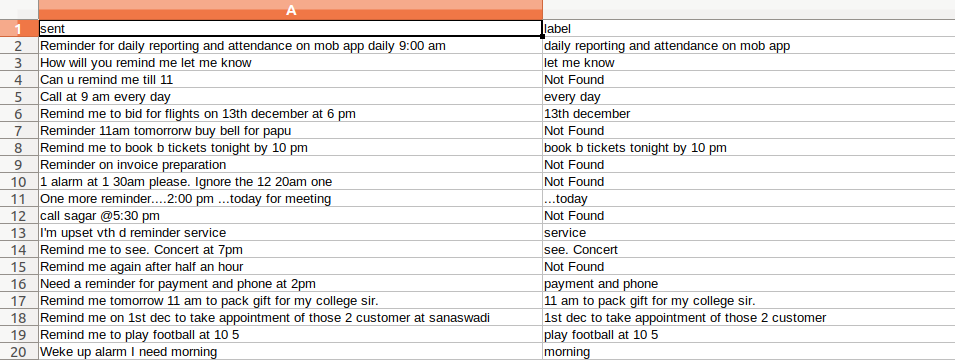


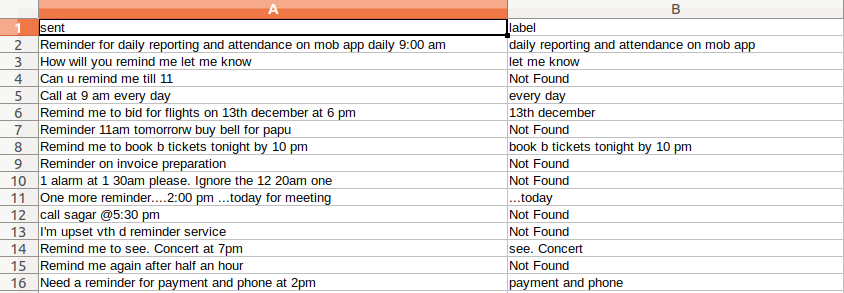
Predicting Summary (<https://github.com/ApoorvTyagi/Text-Summariser>)



## 5.3. OUTPUT

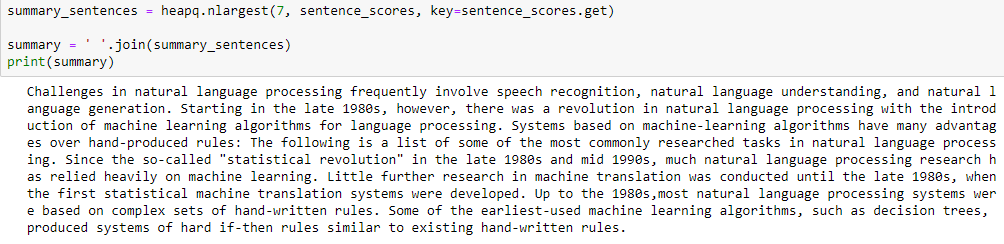
(a):





(b) Summary of a Wikipedia article

<https://en.wikipedia.org/wiki/Natural_language_processing>



# CONCLUSION

As can be seen, this approach seems to be working fairly well.  
The extracted chunks do convey some of the key themes present in the text.  
We can, of course, try out more complex techniques & alternate approaches to get more better results.  
However, the Extraction of important topical words and phrases from documents can easily be done using this approach and the usefulness of this techniques for certain NLP Tasks are insane.

# REFRENCES

[1] Ana Mestrovic, Beliga, Sanda Martincic-Ipsic, Slobodan, "An overview of graph-based keyword extraction methods and approaches", Journal of Information and Organizational Sciences 39, (2015).

[2] Aditi Sharan, Siddiqi, Sifatullah, "Keyword and keyphrase extraction techniques: A literature review", International Journal of Computer Applications” 109, (2015).

[3] Alexander Gelbukh, Erendira Rendon, GarciaHernandez, Rafael Cruz, Rene Arnulfo, Romyna Montiel, Yulia Ledeneva, "Text Summarization by Sentence Extraction Using Unsupervised Learning" , Mexican International Conference on Artificial Intelligence, Springer.

[4] Carl Gutwin, Craig G. Nevill-Manning, Eibe Frank, Gordon W. Paynter , Ian H., Witten, "KEA: Practical automatic key-phrase extraction" , Proceedings of the fourth ACM conference on Digital libraries.

[5] Peter D, Turney, "Learning algorithms for key-phrase Extraction", Information retrieval 2, no. 4 (2000).