“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, extracted, and cleansed 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

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) 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.

2. LITERATURE SURVEY

2.1.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.

2.2.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.

2.3.SURVEY-3

"Semantic NLP-Based Information Extraction from Construction Regulatory Documents for Automated Compliance Checking" by Jiansong Zhang and Nora M. El-Gohary 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.

2.4.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.

3. OVERVIEW OF THE WORK

3.1.REQUIRED MODULES

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

3.2.SOFTWARE REQUIREMENT

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

3.3.HARDWARE REQUIREMENT

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

4. SYSTEM DESIGN

4.1. 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.

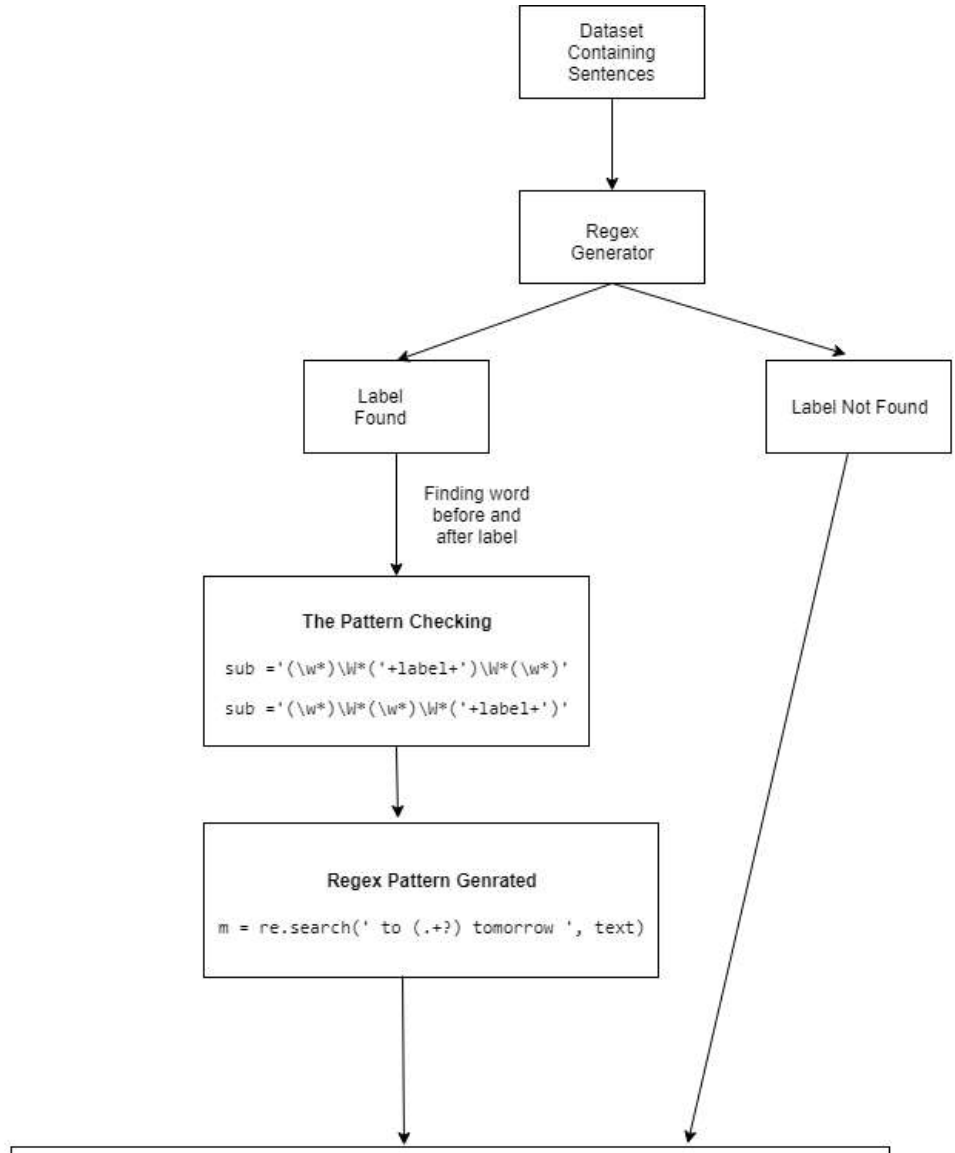
This mechanism helps us to do a meaningful “Phrase Extraction” as the model is trained with the mentioned 100 layers to give us a meaningful result in contrast to simple Phrase extraction which may result in unwanted phrases or words that may lead to anomalous results.

4.2. 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_classifiertraining_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 dataframedf=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 arraytest_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 listpredicted_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
```

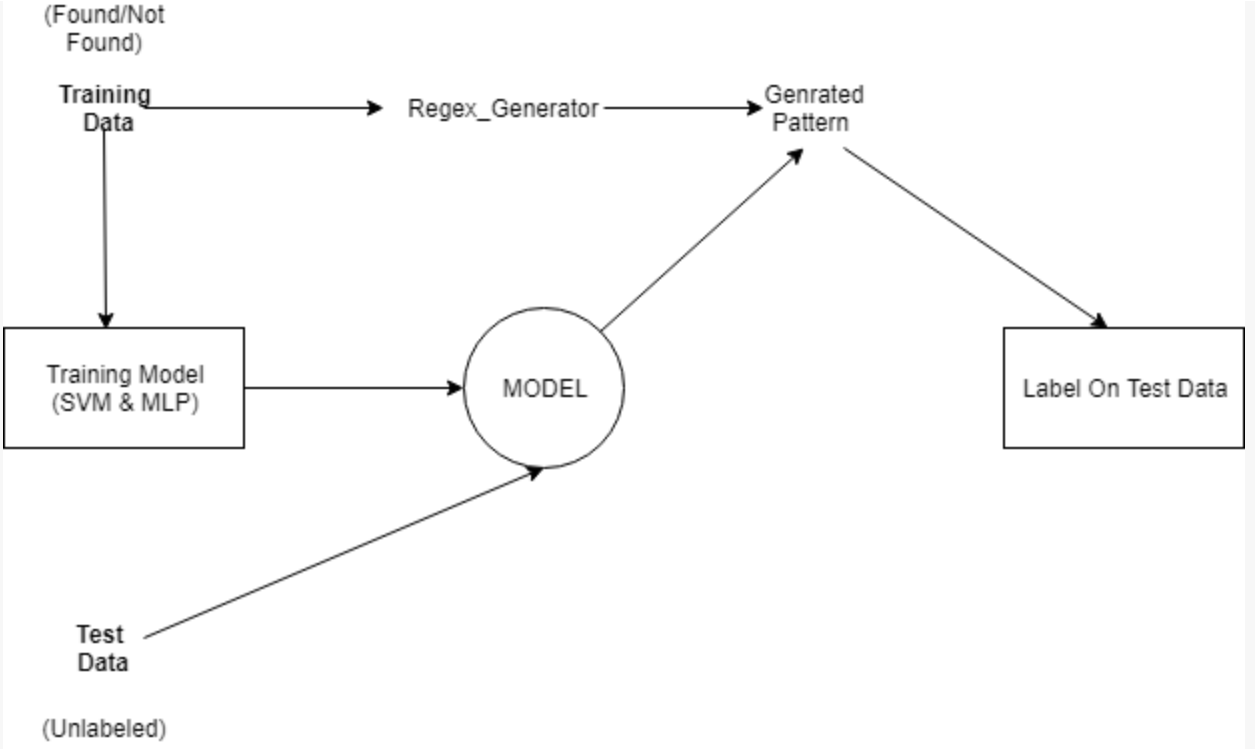
```
final_result2=[]for idx, prediction in enumerate(predicted_result2):
    if (prediction=='Not Found'):
        final_result2.append('Not Found') # if classifier identifies the sentence to be labled as Not Found it has power to over write the extracter
    else:
        final_result2.append(extractions_from_regex[idx])

###### 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



	A	B
1	text	label
2	Reminder for daily reporting and attendance on mob app daily 9:00 am	daily reporting and attendance on mob app
3	How will you remind me let me know	let me know
4	Can u remind me till 11	Not Found
5	Call at 9 am every day	every day
6	Remind me to bid for flights on 13th december at 6 pm	13th december
7	Reminder 11am tomorrow buy bell for papu	Not Found
8	Remind me to book b tickets tonight by 10 pm	book b tickets tonight by 10 pm
9	Reminder on invoice preparation	Not Found
10	1 alarm at 1 30am please. Ignore me 12 20am one	Not Found
11	One more reminder....2:00 pm ...today for meeting	...today
12	call sagar @5:30 pm	Not Found
13	I'm upset vth d reminder service	service
14	Remind me to see. Concert at 7pm	see. Concert
15	Remind me again after half an hour	Not Found
16	Need a reminder for payment and phone at 2pm	payment and phone
17	Remind me tomorrow 11 am to pack gift for my college sir	11 am to pack gift for my college sir
18	Remind me on 1st dec to take appointment of those 2 customer at saraswadi	1st dec to take appointment of those 2 customer
19	Remind me to play football at 10 5	play football at 10 5
20	Woke up alarm i need morning	morning

	A	B
1	text	label
2	Reminder for daily reporting and attendance on mob app daily 9:00 am	daily reporting and attendance on mob app
3	How will you remind me let me know	let me know
4	Can u remind me till 11	Not Found
5	Call at 9 am every day	every day
6	Remind me to bid for flights on 13th december at 6 pm	13th december
7	Reminder 11am tomorrow buy bell for papu	Not Found
8	Remind me to book b tickets tonight by 10 pm	book b tickets tonight by 10 pm
9	Reminder on invoice preparation	Not Found
10	1 alarm at 1 30am please. Ignore me 12 20am one	Not Found
11	One more reminder....2:00 pm ...today for meeting	...today
12	call sagar @5:30 pm	Not Found
13	I'm upset vth d reminder service	Not Found
14	Remind me to see. Concert at 7pm	see. Concert
15	Remind me again after half an hour	Not Found
16	Need a reminder for payment and phone at 2pm	payment and phone



4. IMPLEMENTATION [Github Link]

4.1.CODE SNIPPETS

- Finding word before and after label

The Pattern Checking

```
sub = '(\w*)\W*('+label+')\W*(\w*)'
sub = '(\w*)\W*(\w*)\W*('+label+')'
```

4.2.

Regex Pattern Genrated

```
...
...
m = re.search(' to (.+?) at ', text)
if m:
    found = m.group(1)
    small_master_list.append(found)

m = re.search(' to (.+?) on ', text)
if m:
    found = m.group(1)
    small_master_list.append(found)
m = re.search(' to (.+?) tomorrow ', text)
...
...
```

Training the algorithm

```
clf = svm.LinearSVC(loss='hinge').fit(X_train_tfidf, y_train)
mlp = MLPClassifier(activation='relu', solver='lbfgs').fit(X_train_tfidf, y_train)
```

Accuracy obtained using the model

```
Accuracy with SVM is :0.868020304568528
Accuracy score with Multi Layer Perceptron: 0.8477157360406091
```

Predicting Summary

```
sentence_scores = {}
for sent in sentence_list:
    for word in nltk.word_tokenize(sent.lower()):
        if word in word_frequencies.keys():
            if len(sent.split(' ')) < 30:
                if sent not in sentence_scores.keys():
                    sentence_scores[sent] = word_frequencies[word]
                else:
                    sentence_scores[sent] += word_frequencies[word]
```

```
summary_sentences = heapq.nlargest(7, sentence_scores, key=sentence_scores.get)

summary = ' '.join(summary_sentences)
print(summary)
```

5.3. OUTPUT

(a):

	A	
1	sent	label
2	Reminder for daily reporting and attendance on mob app daily 9:00 am	daily reporting and attendance on mob app
3	How will you remind me let me know	let me know
4	Can u remind me till 11	Not Found
5	Call at 9 am every day	every day
6	Remind me to bid for flights on 13th december at 6 pm	13th december
7	Reminder 11am tomorrorw buy bell for papu	Not Found
8	Remind me to book b tickets tonight by 10 pm	book b tickets tonight by 10 pm
9	Reminder on invoice preparation	Not Found
10	1 alarm at 1 30am please. Ignore the 12 20am one	Not Found
11	One more reminder....2:00 pm ...today for meeting	...today
12	call sagar @5:30 pm	Not Found
13	I'm upset vth d reminder service	service
14	Remind me to see. Concert at 7pm	see. Concert
15	Remind me again after half an hour	Not Found
16	Need a reminder for payment and phone at 2pm	payment and phone
17	Remind me tomorrow 11 am to pack gift for my college sir.	11 am to pack gift for my college sir.
18	Remind me on 1st dec to take appointment of those 2 customer at sanaswadi	1st dec to take appointment of those 2 customer
19	Remind me to play football at 10 5	play football at 10 5
20	Weke up alarm I need morning	morning

	A	B
1	sent	label
2	Reminder for daily reporting and attendance on mob app daily 9:00 am	daily reporting and attendance on mob app
3	How will you remind me let me know	let me know
4	Can u remind me till 11	Not Found
5	Call at 9 am every day	every day
6	Remind me to bid for flights on 13th december at 6 pm	13th december
7	Reminder 11am tomorrorw buy bell for papu	Not Found
8	Remind me to book b tickets tonight by 10 pm	book b tickets tonight by 10 pm
9	Reminder on invoice preparation	Not Found
10	1 alarm at 1 30am please. Ignore the 12 20am one	Not Found
11	One more reminder....2:00 pm ...today for meeting	...today
12	call sagar @5:30 pm	Not Found
13	I'm upset vth d reminder service	Not Found
14	Remind me to see. Concert at 7pm	see. Concert
15	Remind me again after half an hour	Not Found
16	Need a reminder for payment and phone at 2pm	payment and phone

(b) Summary of a Wikipedia article
https://en.wikipedia.org/wiki/Natural_language_processing

```
summary_sentences = heapq.nlargest(7, sentence_scores, key=sentence_scores.get)

summary = ' '.join(summary_sentences)
print(summary)
```

```
Challenges in natural language processing frequently involve speech recognition, natural language understanding, and natural l
anguage generation. Starting in the late 1980s, however, there was a revolution in natural language processing with the introd
uction of machine learning algorithms for language processing. Systems based on machine-learning algorithms have many advantag
es over hand-produced rules: The following is a list of some of the most commonly researched tasks in natural language process
ing. Since the so-called "statistical revolution" in the late 1980s and mid 1990s, much natural language processing research h
as relied heavily on machine learning. Little further research in machine translation was conducted until the late 1980s, when
the first statistical machine translation systems were developed. Up to the 1980s,most natural language processing systems wer
e based on complex sets of hand-written rules. Some of the earliest-used machine learning algorithms, such as decision trees,
produced systems of hard if-then rules similar to existing hand-written rules.
```

5. 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.

Class Code Assignments

- Activity 1 (https://github.com/varunreddy24/NLP_Course/tree/master/Activity%201)
- Activity 2 (https://github.com/varunreddy24/NLP_Course/tree/master/Activity%202)
- Activity 3 (https://github.com/varunreddy24/NLP_Course/tree/master/Activity%203)

6. 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).