Delhi Technological University

Project Report

Object Oriented Programming (CO – 203)

Keyword Extraction

Submitted By

Ankit Kumar Shrivastav (2K19/CO/060)
Anshuman Raj Chauhan (2K19/CO/067)

Ankit Kumar Shrivastav
(25/11/20)

Anshuman Raj Chauhan
(25/11/20)

DELHI TECHNOLOGICAL UNIVERSITY

Vision:

"To be a world class University through education, innovation and research for the service of humanity"

Mission:

1. To establish centres of excellence in emerging areas of science, engineering, technology, management and allied areas.
2. To foster an ecosystem for incubation, product development, transfer of technology and entrepreneurship.
3. To create an environment of collaboration, experimentation, imagination and creativity.
4. To develop human potential with analytical abilities, ethics and integrity.
5. To provide environment friendly, reasonable and sustainable solutions for local & global needs.

DEPARTMENT OF COMPUTER SCIENCE

AND ENGINEERING

Vision:

Department of Computer science and engineering to be a leading world class technology department playing its role as a key node in national and global knowledge network, thus empowering the computer science industry with the wings of knowledge and the power of innovation.

Mission:

The mission of the department is as follows:

1. To nurture talent of students for research, innovation and excellence in the field of computer engineering starting from undergraduate level.
2. To develop highly analytical and qualified computer engineers by imparting training on cutting edge technology.
3. To produce socially sensitive computer engineers with professional ethics.
4. To focus on R&D environment in close partnership with industry and foreign universities.
5. To produce well-rounded, up to date, scientifically tempered, design oriented engineers and scientists capable of lifelong learning.

Index

Objective

To build a keyword extraction model for extracting meaningful information from textual data.

Motivation

The growth of internet has transformed the world drastically. It has opened up doorways to infinite applications and possibilities like social media, knowledge sharing (blog based sites, online research papers, etc), communication (video conferencing, online lectures, etc) , gaming , entertainment and many more.

Today there are around 4.5 billion active internet users across the globe and estimates suggest a total of 1200 petabytes of data on the internet.

Extracting meaningful insights from these large amounts of data can help the world in ways unimagined.

In fact, this has resulted in emergence of altogether a different field in Computer Science called Data Science.

Keyword Extraction from textual data on the internet can help in various tasks like information retrieval, automatic indexing, automatic classification, automatic clustering, automatic filtering, faster search engines, etc

Introduction

Keywords are described as a series of one or more words which provide a compact representation of a documents' content. However, assigning keywords to documents manually is very costly, time consuming, and tedious task, therefore automatic keyword extraction has attracted the interest of researchers over the last few years.

Automatic keyword extraction is done through mainly 4 approaches-

1. Supervised approach
2. Unsupervised approach
3. Statistical approach
4. Hybrid approach.

We have used the graph based statistical approach in this project.

Many existing graph-based keyword extraction approaches determine keywords purely based on centrality measure. However, various features such as frequency, centrality and position of the keyword also affect the importance of a keyword.

Therefore, this projects implements a novel unsupervised graph-based keyword extraction method called keywords from collective weights (KCW) which determines the

importance of a keyword by collectively considering various influencing features. The KCW is based on node-edge rank centrality with node weight depending on various features.

Our model has been built based on the paper

Keyword extraction from micro-blogs using collective weight by (Monali Bordoloi[1] · Saroj Kr. Biswas[1])

and implemented on a dataset consisting of research papers on machine learning.

They built it specifically for set of tweets; we've tried to keep our algorithm more general.

OOP Involvement

Object-Oriented Programming or OOPs refers to languages that uses objects in programming. OOP aims to implement real-world entities like inheritance, hiding, polymorphism etc in programming.

In this project 'classes' for graph and text preprocessing have been used.

These classes contain all the functions and data required for their purpose encapsulated into one.

Preprocess Class:

Private Members:

1. text(corpus to preprocess)
2. sw(list of stop words)

Public Members:

1. stopwordRemoval (function)
2. computeAOF (for computing average occurrence frequency)
3. vectorise(for vectorisation of corpus)

GraphFormulation Class:

Data Members:

1. V(size of vocabulary)
2. nodeWts(array for storing node weights)
2. wt(adjacency matrix for graph)
3. F (first word node weight component)
4. L (last world node weight component)
5. SC (selective centrality node weight component)
6. TF (term frequency node weight component)
7. D (distance node weight component)

Member Functions:

1. addEdgeWt(adding edge weight to adjacency matrix)
2. computeEdgeWt(computing edge weight)
3. computeNodeWt(computing node weight)
4. makeGraph (for calling above functions to construct graph)

** The graph has been developed from scratch and each of the preprocessing steps has been performed explicitly with minimal use of external python libraries. **

Implemented KCW model

The KCW model considers frequency, centrality and position of a node to calculate importance of the node. The implementation of the model is divided into four phases: pre-processing, textual graph representation, node weight assignment, and keyword extraction. The flowchart of the proposed model is shown in Figure1.

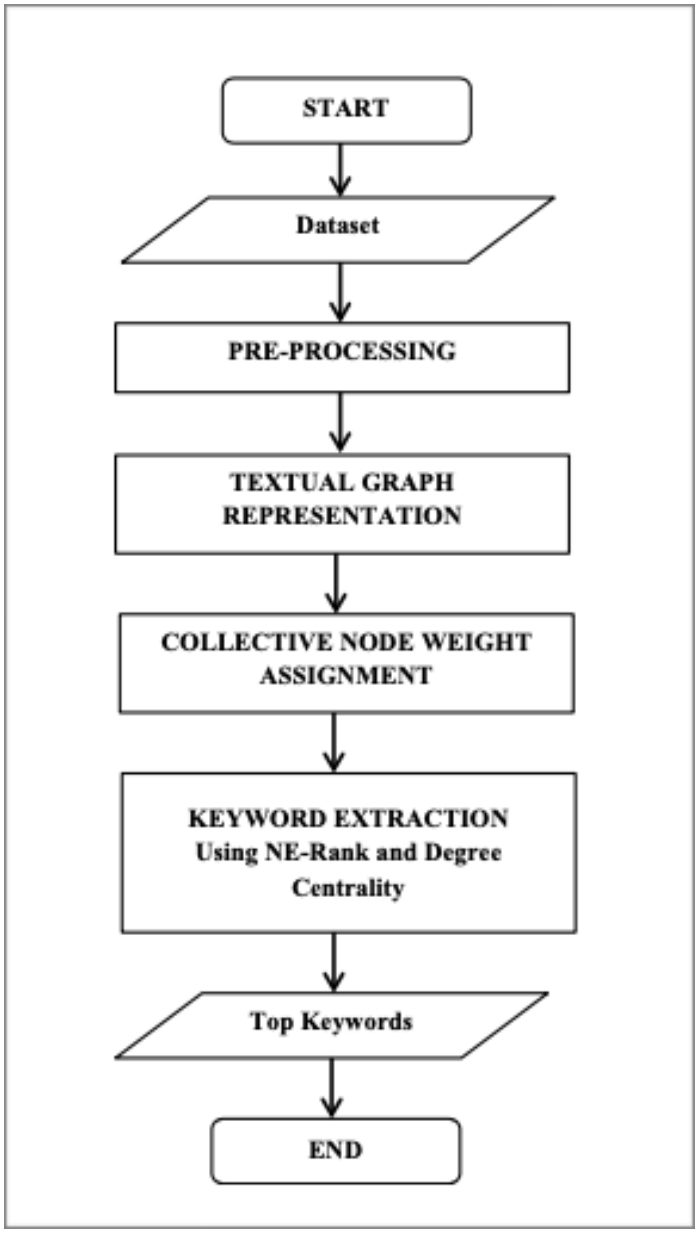


Figure 1

Phase 1: Pre-processing

Text pre-processing in this model involves the following steps-

1. Tokenisation with removal of unwanted characters

Regular expression tokeniser by NLTK has been used to split sentences into sequence of words and removing all unwanted characters and symbols like '@','#','$','/', etc.

2. Lower Casing

All characters have been lower cased in order to avoid multiple nodes with same word in the graph.

3. Stopword Removal and lemmatisation

A standard list of stop words such as about, be, etc. that does not contribute much in the overall meaning has been removed from the tokenised sentences.

4. Removal of tokens with frequency below threshold frequency

An average occurrence frequency(AOF) has been computed and the words with frequency less than that have been removed. Formula for AOF calculation is as follows-

$$\text{AOF} = \frac{\sum \text{Frequency of each token}}{\text{Number of tokens}}.$$

5. Vectorisation

Each token has been assigned a number(vector) and thus a vocabulary has been created on the given corpus.

Phase 2: Textual Graph Representation

Let G = (V, E) be a graph, where V is the set of vertices and E is the set of edges. The textual graph is then represented as described below.

1. Vertex assignment

Each token has been used to create one vertex each. The set of vertices (V) has been created from the set of tokens in the vertex assignment.

2. Edging

If 2 nodes(words) occur in the same 'paper' there is an edge b/w them.

Edge Weight assignment has been done using the following formula:

$$W_c(i,j) = \frac{\text{freq}(i,j)}{\text{freq}(i) + \text{freq}(j) - \text{freq}(i,j)},$$

where freq(i, j) is the number of times node i and j co-occur, and freq(i) and freq(j) are the occurrence frequencies of nodes i and j, respectively.

Phase 3: Node Weight Assignment

In keyword extraction using graph-based model, the weight of a node plays a vital role. Proper node weight evaluation leads to effective and representative keywords determination for tweets/text. Many factors affect the importance of a node. The KCW model considers four different important features to calculate the node weights as discussed below:

1. Position of a node

Hotho et al. (2005) suggested that the position of a term is an important criterion while extracting keywords. Therefore added weight is given to them, which is calculated as follows.

i. If token i is the first word then,

$$F[i] = \frac{n_f}{freq(i)}$$

where, F[i] is the weight of i and n_f is the number of times i is the first word.

ii. Else F[i] =0

i. If token i is the last word then,

$$L[i] = \frac{n_l}{freq(i)}$$

where, L[i] is the weight of i and n_l is the number of times i is the last word.

ii. Else L[i] =0

2. Term frequency

Important keywords tend to occur more frequently in a document. Thus, the frequency of the keywords forms an essential parameter in the node weight determination. Term frequency is defined as the number of times a given term

occurs in a document. Here, the term frequency is the average of number of times the term occurs in the each paper.

3. Selectivity centrality

Selectivity centrality is defined as the average weight on the links of a single node. Centrality measure is an indication of the importance of a node within a graph. It is calculated as follows-

$$SC(v) = \frac{s(v)}{d(v)}$$

$$s(v) = \sum_{u} W_{vu}.$$

where Wuv is the weight b/w node ‘u’ and node ‘v’ and d(v) is the out degree of node ‘v’.

4. Distance from central node:

The closer a node is to the most central node the more probability it has to be an important keyword. Therefore, importance of a node is calculated as the inverse of its distance from the central node. The central node is found using degree of the nodes. The node with the highest degree is considered as the most central. In case there is a tie in the degree of nodes, frequency of the nodes is used to break the tie. If central node is c, then the importance for a node i is determined by

$$D_{C(i)} = \frac{1}{d(c,i)},$$

where d(c, i) is distance of node i from the central node c. This value is normalised to be within the range of 0 and 1. For the central node c, the DC(c) value is set to 1.

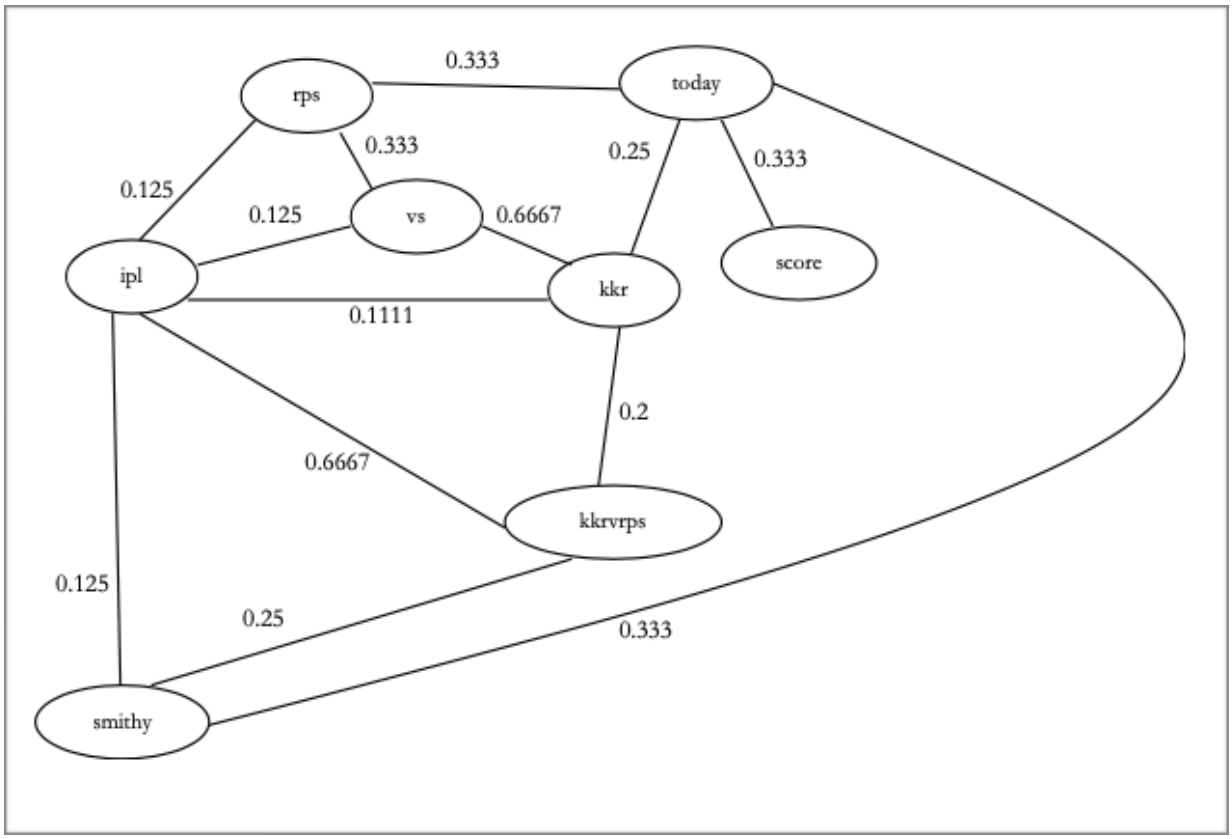
The weight of any node i is calculated by given eq.

$$Node(wt, i) = F[i] + L[i] + TF[i] + SC[i] + D[i]$$

and is normalised to get value between 0 and 1 by the given equation

$$\text{Final}_{\text{weight}(i)} = \frac{\text{Node\_weight}(i) - \text{min\_weight}}{\text{max\_weight} - \text{min\_weight}},$$

The normalised value of i is the final weight of i.



Sample Graph representation of words

Phase 4: Keyword Extraction

The process of identifying keywords from a paper/document that can appropriately represent the subject of the paper/ document is termed keyword extraction. To extract the important keywords, the proposed model uses NE rank and degree centrality.

1. Calculate NE-Rank

For a node vi, the rank/relevance using NE rank is calculated using given equation-

$$R(v_i) = (1-d)\cdot W(v_i) + d\cdot W(v_i)\cdot \sum_{j:v_j\to v_i} \frac{w_{ji}}{\sum_{k:v_j\to v_k} w_{jk}} R(v_j).$$

d is the damping factor which denotes the probability of jumping from a node to the next node and is usually set to 0.85; and (1 – d) denotes the probability of jumping to a new node. W(vi) is the weight of the current node vi. wji is the weight of the edge from the previous vertex vj to the current vertex vi and (∑k:vj→vk wjk) is the summation of all edge weights in the previous node vj. Here, the rank/ relevance of node vj is denoted by R(vj).

2. Calculate degree centrality

Degree centrality of a node is defined as the number of edges incident on the node.

3. Sort keywords

Sort the keywords using the following algorithm.

Let i and j be two terms/nodes such that j occurs immediately after i in the list/text. Then:	
i.	If NE(i) = NE(j)
	If degree(i)<degree(j) Swap (i, j)
ii.	Else if NE(i) < NE(j) Swap (i, j)
iii.	Otherwise No action

4. Select top keywords

Finally, n best ranked terms/nodes are selected as keywords.

Implementation

Because of the large size of the source code it has been uploaded on GitHub

Link-

https://github.com/its7ARC/kewordExtraction/blob/main/_keywordExtraction.ipynb

Result and Conclusion

The constructed KCW model extracts keywords with great accuracy, reinforcing the fact that node weights are as important a parameter as edge weights in automatic keyword extraction.

Printing Top Keywords¶

```
topKeywords = R.argsort()[-20:][::-1]

for i in range(20):
    print(invVocab[topKeywords[i]])
```

```
network
neural
input
use
show
error
model
set
control
figure
weight
time
function
plasma
test
data
cell
value
parameter
result
```

Resources Used

1. Numpy library

2. NLTK library

3. SKLearn

4. Jupyter Notebook

References

1. Monali Bordoloi - KCW Model - https://www.researchgate.net/publication/327616483_Keyword_extraction_from_micro-blogs_using_collective_weight - [8-9-20]

2. Ajay Alex - Keyword extraction techniques - https://medium.com/analytics-vidhya/keyword-extraction-techniques-using-python-edea5fc35678 - [12-9-20]

3. Sowmya Vivek - Automated Keyword Extraction - https://medium.com/analytics-vidhya/automated-keyword-extraction-from-articles-using-nlp-bfd864f41b34 - [12-9-20]

4. Sudalai Raj Kumar - Text Preprocessing Guide - https://www.kaggle.com/sudalairajkumar/getting-started-with-text-preprocessing - [10-10-2020]

5. GeeksForGeeks - Object Oriented Programming - https://www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/ - [12-10-20]

6. GeeksForGeeks - Graph Representation - https://www.geeksforgeeks.org/graph-and-its-representations/ - [14-10-20]

7. GeeksForGeeks - Graph Theory - https://www.geeksforgeeks.org/mathematics-graph-theory-basics-set-1/ - [14-10-20]

8. GeeksForGeeks - Private data members in python - https://www.geeksforgeeks.org/private-variables-python/ - [18-10-20]