A TREE BASED KEYPHRASE EXTRACTION TECHNIQUE FOR ACADEMIC LITERATURE

GOLLAM RABBY

Master of Science

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name	: GOLLAM RABBY
Date of Birth	: 01/01/1996
Title	: A TREE BASED KEYPHRASE EXTRACTION TECHNIQUE FOR ACADEMIC LITERATURE
Academic Session	: SEM 2 2018/2019

I declare that this thesis is classified as:

☐ CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
☐ RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
☑ OPEN ACCESS	I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

_______________	_______________
(Student's Signature)	(Supervisor's Signature)
_______________	_______________
New IC/Passport Number	Name of Supervisor
Date:	Date:

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

SUPERVISOR'S DECLARATION

We hereby declare that We have checked this thesis and, in our opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Master of Science.

____

(Supervisor's Signature)

Full Name	: DR. MD SAIFUL AZAD
Position	: SENIOR LECTURER
Date	:

____

(Co-supervisor's Signature)

Full Name	: DR. MOHD FAAIZIE BIN DARMAWAN
Position	: SENIOR LECTURER
Date	:

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

____

(Student's Signature)

Full Name	: GOLLAM RABBY
ID Number	: MCC17014
Date	:

A TREE BASED KEYPHRASE EXTRACTION TECHNIQUE FOR ACADEMIC LITERATURE

GOLLAM RABBY

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Master of Science

Faculty of Computer Systems and Software Engineering

UNIVERSITI MALAYSIA PAHANG

August 2019

ACKNOWLEDGEMENTS

I am grateful and would like to express my sincere gratitude to my supervisor, Dr. Md Saiful Azad, for his emerging ideas, invaluable guidance, continuous encouragement and constant supports in making this research possible. He has always impressed me with his outstanding professional conduct, his strong conviction for science, and his belief that a master's program is only a start of a life-long learning experience. I appreciate his consistent support from the first day I applied to graduate program to these concluding moments. I am truly grateful for his progressive vision about my training in science, his tolerance of my naive mistakes, and his commitment to my future career. I would like to express very special thanks to my co-supervisor, Dr. Mohd Faaizie Bin Darmawan, for his suggestions and co-operations throughout the study. I would also like to place on record, my sincere thanks to Assoc. Prof. Ts. Dr. Mohamad Fadli Zolkipli, Dean of Faculty of Computer Systems & Software Engineering (FSKKP), for guideline, continues encouragement and providing me with all the necessary facilities for the research. I also sincerely thank him for the time spent for correcting my many mistakes.

I would also like to thank my parents who not only supported me, also funded my study in initial level. Without this support, it would not be easy for me to start the master's course. Many special thanks go to members of SysNet Lab research group for their excellent co-operations, inspirations, and supports during this study.

Next, my appreciations are expressed towards the administrative staffs in the FSKKP, Institute of Postgraduate Studies (IPS) and International Office (IO) for their passionate to handle all forms of official tasks during my master study. The appreciations also go to all members of Bangladeshi community, local and foreign friends in UMP.

Finally, this research work of master's study is funded and supported by Masters Research Scheme (MRS), from IPS, UMP. I would like to thank UMP for funding the research and helping me to concentrate on the study.

ABSTRAK

Teknik pengekstrakan frasa kekunci secara automatik bertujuan untuk mengekstrak frasa kekunci yang berkualiti untuk meringkaskan dokumen pada tahap yang lebih tinggi. Antara teknik yang sedia ada, sebahagiannya adalah domain khusus dan memerlukan pengetahuan domain aplikasi, sebahagiannya lagi berdasarkan kaedah susunan statistik yang lebih tinggi dan sangat mahal, dan sesetengahnya memerlukan data latihan di mana ianya sangat jarang ada dalam kebanyakan aplikasi. Untuk menangani isu-isu ini, penyelidikan ini mencadangkan teknik pengekstrakan frasa kekunci automatik tanpa pengawasan yang baru yang dinamakan TeKET atau Teknik Pengekstrakan Ungkapan Asas berasaskan pokok, yang merupakan domain bebas, memerlukan pengetahuan statistik yang terhad, dan tidak memerlukan data latihan. Teknik yang dicadangkan ini juga memperkenalkan varian baru berstruktur pokok secara binari, yang dikenali sebagai pokok pengekstrakan KeyPhrase (KePhEx) untuk mengekstrak frasa kekunci akhir daripada frasa-frasa kekunci yang terlibat. Struktur Pokok KePhEx adalah bergantung kepada frasa-frasa kekunci yang terlibat, sama ada ianya diperluas, atau menyusut atau dikekalkan. Di samping itu, satu ukuran yang diperolehi, dinamakan sebagai Indeks Kepuasan atau CI, yang menandakan tahap kohesif bagi nod-nod yang diberikan terhadap akar asal yang digunakan untuk mengekstrak kekunci akhir dari pokok yang dihasilkan dalam keadaan yang fleksibel, dan ianya digunakan dalam menanda aras kekunci berserta Tempoh Frekuensi. Keberkesanan teknik yang dicadangkan dinilai menggunakan penilaian eksperimen pada corpus penanda aras, yang dipanggil SemEval-2010 dengan jumlah 244 artikel kereta api dan ujian, dan dibandingkan dengan teknik lain yang tidak terjejas dengan mengambil wakil-wakil dari kedua-dua statistik (seperti Dokumen Invers Frequency-Inverse Frekuensi dan YAKE) dan teknik berasaskan grafik (PositionRank, CollabRank (SingleRank), TopicRank, dan MultipartiteRank). Tiga metrik penilaian, iaitu ketepatan, ingat dan skor F1 diambil kira semasa eksperimen. Sebagai contoh, Hasil yang diperoleh menunjukkan prestasi yang lebih baik dari teknik yang dicadangkan berbanding teknik-teknik serupa yang lain dari segi ketepatan, ingat, dan skor F1.

ABSTRACT

Automatic keyphrase extraction techniques aim to extract quality keyphrases to summarize a document at a higher level. Among the existing techniques some of them are domain-specific and require application domain knowledge, some of them are based on higher-order statistical methods and are computationally expensive, and some of them require large train data which are rare for many applications. Overcoming these issues, this thesis proposes a new unsupervised automatic keyphrase extraction technique, named TeKET or Tree-based Keyphrase Extraction Technique, which is domain-independent, employs limited statistical knowledge, and requires no train data. The proposed technique also introduces a new variant of the binary tree, called KeyPhrase Extraction (KePhEx) tree to extract final keyphrases from candidate keyphrases. Depending on the candidate keyphrases the KePhEx tree structure is either expanded or shrunk or maintained. In addition, a measure, called Cohesiveness Index or CI, is derived that denotes the degree of cohesiveness of a given node with respect to the root which is used in extracting final keyphrases from a resultant tree in a flexible manner and is utilized in ranking keyphrases alongside Term Frequency. The effectiveness of the proposed technique is evaluated using an experimental evaluation on a benchmark corpus, called SemEval-2010 with total 244 train and test articles, and compared with other relevant unsupervised techniques by taking the representatives from both statistical (such as Term Frequency-Inverse Document Frequency and YAKE) and graph-based techniques (PositionRank, CollabRank (SingleRank), TopicRank, and MultipartiteRank) into account. Three evaluation metrics, namely precision, recall and F1 score are taken into consideration during the experiments. The obtained results demonstrate the improved performance of the proposed technique over other similar techniques in terms of precision, recall, and F1 scores.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

χ	Extract Candidate Keyphrases
Z+	All Positive Integer Number
Φ	Final Keyphrase
ρ	Precision
ς	Recall
φ	F1−score
δ	Document
λ	Constant Value
N	Any Positive Integer Number
η	List
γ	Root
σ	Similar candidate keyphrases
w	Word
d	Depth
ℓ	Level
μ	Maturity Index
$\in$	Is an Element Of
Mamu	Minimum Allowable
Ω	Weight
P	Keyphrase

LIST OF ABBREVIATIONS

ML	Machine Learning
TF	Term Frequency
IDF	Inverse Document Frequency
TF-IDF	Term Frequency - Inverse Document Frequency
HAC	Hierarchical Agglomerative Clustering
XML	Extensible Markup Language
POS	Part-Of-Speech
POST	Part-Of-Speech Tagging
SR	SingleRank
PR	PositionRank
TR	TopicRank
MR	MultipartiteRank
PKE	Python Keyphrase Extraction
API	Application Programming Interface
JJ	Adjectives
NN	Nouns
PRP	Personal Pronouns
DT	Determiners
PP	Prepositional Phrases
IN	Proposition/Subordinating Conjunctions
NNS	Noun Plural
NNP	Proper Noun, Singular
NNPS	Proper Noun, Plural
JJR	Adjective, Comparative
JJS	Adjective, Superlative

CHAPTER 1

INTRODUCTION

1.1 Preamble

Automatic keyphrase extraction techniques endeavor to extract quality keyphrases automatically from documents. Generally, these keyphrases provide a high-level summarization of that document. Therefore, they are utilized in many digital information processing applications, such as information retrieval (Zhai & Lafferty, 2017), digital content management (Brown & Duguid, 1998; Vallez, Pedraza-Jimenez, Codina, Blanco, & Rovira, 2015), natural language processing (Huang, Zhang, & Vogel, 2005; Reilly & Sharkey, 2016), contextual advertisement (Sterckx, Demeester, Deleu, & Develder, 2018; Yoo & Eastin, 2017), recommender system (Pudota, Dattolo, Baruzzo, Ferrara, & Tasso, 2010; Ricci, Rokach, & Shapira, 2011). Herein, the concept of information retrieval has been developed to extract desired information from a large collection of textual data. It has been implemented in many practical applications, such as search engines (Tümer, Shah, & Bitirim, 2009), media search, digital libraries (Lawrence, Giles, & Bollacker, 1999), geographic information retrieval (Hariharan, Hore, Li, & Mehrotra, 2007), legal information retrieval (Chor, Gilboa, & Naor, 1997), and others. It is inane explaining the necessity of these systems; since what data can we retrieve without these systems.

Keyphrases play an important role in digital content management or digital library (academic research papers) (Seuring & Gold, 2012). They are utilized for document indexing (Rowley & Hartley, 2017a) to describe or classify the semantic similarity among various documents (a.k.a., document clustering (Steinbach, Karypis, & Kumar, 2000; W. Xu, Liu, & Gong, 2003) or document classification (Manevitz & Yousef, 2001; Wu et al., 2017)); and thereby, can be utilized as recommender systems to improve the browsing experience of digital libraries. Furthermore, document classification and similar concepts are widely used in machine learning (ML), data mining, database discovery, and so on. Some notable applications using these techniques are: news group filtering, target marketing, document organization, health status tracking, and so on (Adeniyi, Wei, & Yongquan, 2016; Das, Datar, Garg, & Rajaram, 2007; Franceschini, Maisano, & Mastrogiacomo, 2016; Kononenko, 2001; Kotler & Roberto, 1989; McCallum, Nigam, et al., 1998). In addition, for any contextual advertising to display advertisements based on user identity and browsing history, keyphrase extraction is a core technique.

1.2 Research Background & Motivation

To support these aforementioned applications, several keyphrase extraction techniques have been proposed (Dashtipour et al., 2016; El-Beltagy & Rafea, 2009a; Freitag, 2000; Herrera & Pury, 2008; Holzinger, 2017; Hulth, 2003a; Litvak & Last, 2008a; Siddiqi & Sharan, 2015; Thomas, Bharti, & Babu, 2016; Vencovsky, Lucas, Mahr, & Lemmink, 2017; Wolf, Zhu, Semret, & Baskin, 2013; Zhang, Xu, Tang, & Li, 2006). Among them, domain specific approaches (Frank, Paynter, Witten, Gutwin, & Nevill-Manning, 1999a) require knowledge of the application domain, and linguistic approaches (Tomokiyo & Hurst, 2003) require expertise of the language, thus are inapplicable in problems from other domains and languages.

Among the Machine Learning based techniques, supervised Machine Learning techniques (Hasan & Ng, 2014; Kosala & Blockeel, 2000) perform better in several domains than other existing techniques. However, they demand a considerable amount of train data to extract quality keyphrases, which is rare at present in many domains including academic literature (Holzinger, 2017). Therefore, these techniques are not considered in this thesis.

Among the unsupervised techniques, statistical techniques (Campos et al., 2018b; El-Beltagy & Rafea, 2009a) are computationally expensive due to their large amount of complex operations, and graph-based unsupervised techniques (Boudin, 2018; Bougouin, Boudin, & Daille, 2013; Florescu & Caragea, 2017b; Sterckx, Demeester, Deleu, & Develder, 2015; Wan & Xiao, 2008) perform poorly due to their incapability in identifying cohesiveness among various words that form a keyphrase (Hasan & Ng, 2014). In light of the aforementioned discussion, the automatic keyphrase extraction remains an important research area to explore (Girardi & Marinho, 2007); and hence, it has been taken into account in this thesis. The limitation of the existing unsupervised techniques are briefly stated in the subsequent section (Section 1.3) to exhibit the research gaps.

1.3 Problem Statement

Web growth and evolution have changed the characteristics of both researcher and academic document/research papers collections. In fact, the participative Web or digital libraries allows a growing number of the researcher to access and populate document collections in a simple way, producing larger and larger collections. As a result, document collections can be explored by a very large set of the researcher, who access the repositories in order to satisfy various personal information needs. Unfortunately, this growing size of the digital information space prevents an effective access to knowledge due to the well-known phenomenon of information overload. Therefore, the evolution of users and document collections require innovative ways to access Web contents. A viable solution to these problems is used keywords or keyphrases, i.e. to first identify (by keywords or keyphrases) and model the specific information needs of the researcher and to subsequently filter Web resources or digital libraries according to the individual researchers. So keyphrases is an important way for information extraction from digital libraries.

Among the existing machine learning based keyphrase extraction techniques, unsupervised techniques are taken into account in this thesis due to their several notable advantages like simplicity in terms of implementation, lower computational complexity, and non-requirement of train dataset, which is rare at present as noted in the preceding section (Section 1.2). However, most of the existing unsupervised techniques also experience several limitations — which inspire this researcher to propose a novel similar technique —

are highlighted below.

KeyGraph (Ohsawa, Benson, & Yachida, 1998) is one of the prominent and advanced unsupervised keyphrase extraction techniques which is content sensitive and domain independent technique and utilizes co-occurrence of different terms for indexing vertices of the graph. However, it fails to recognize the connections between the low-frequency items inside clusters and also ignores fundamental relationships between the clusters; and hence, unable to extract the most representative keyphrases (also known as quality keyphrases) from a document.

PageRank (Page, Brin, Motwani, & Winograd, 1999) is another advanced and important unsupervised keyphrase extraction techniques, which is based on the idea of random walks. Each node of the graph corresponds to a candidate keyphrase and an edge connects two related nodes. However, it is suitable for raking pages on the web and social networks, but not suitable for extracting keyphrases from traditional documents due to lack of consideration of cohesiveness.

PositionRank (Florescu & Caragea, 2017b) is an extension of the PageRank that includes a position of a word along with its frequency to score that word. For that reason, this technique works comparatively well for scientific research articles. However, this technique exhibits considerably limited performance due to ignoring topical coverage and diversity.

For the TextRank (Mihalcea & Tarau, 2004), the scientific documents are represented as a directed or undirected weighted co-occurrence network using co-occurrence windows works depends on variable sizes. It experiences several limitations, such as incapability in capturing cohesiveness, provides the sub-optimal solutions due to selecting only the main cores from clusters whereas it is noticed that a numerous number of valuable keyphases places in the lower levels.

SingleRank (Wan & Xiao, 2008) is the extension of TextRank where weights of an edge are equal to the number of two corresponding words co-occur one another. Depends on its predecessor, it does not extract keyphrases properly by collecting ranked words, instead, only noun phrases are extracted from a document. However, sometimes it assigns higher scores to the long keyphrases but non-significant keyphrases are entices in the ranking procedure.

In TopicRank (Bougouin et al., 2013), which is an enhancement to TextRank, utilizes topics as vertices of a graph, not words. It extracts the noun phrases of a document and clustered them into topics. Although, it considers topical coverage and diversity, but fails to weigh various candidates belonging to a single topic. Moreover, it suffers from the error propagation problem.

MultipartiteRank technique (Boudin, 2018) resolves the error propagation problem of TopicRank by building a complete directed multipartite graph where topics are connected only when they belong to different topics. However, it experiences the clustering error due to which it struggles in selecting the most representative candidates.

The most prominent and the state-of-the-art statistical technique is the Term Frequency - Inverse Document Frequency (TF-IDF) technique (Salton & Buckley, 1988a). Although, it is simple in terms of implementation; however, the computation of Inverse Document Frequency (IDF) is expensive in terms of duration and computational power when a large dataset is considered. On the other hand, keyphrases generated using TF is not representative as the frequency of non-representative keyphrases are observed higher than representative keyphrases.

One of the limitation of TF-IDF is that it favors single term as the frequency of single terms are usually higher. KP-Miner (El-Beltagy & Rafea, 2010) solves this problem of TF-IDF through using two parameters, namely α and β. However, since it utilizes IDF for ranking keyphrases, it inherits the limitations of IDF that are mentioned beforehand.

Conversely, YAKE (Campos et al., 2018a) can resolve the IDF problem by considering five features in selecting keyphrases, namely casing, position and frequency of a word, word relatedness to context, and word in the different sentence to calculate the weight of a keyphrase. However, since it generates candidate keyphrases employing N-grams technique, its computational complexity increases linearly with respect to N-grams (C. Xu, Wu, & Liu, 2017). As N-grams produces a considerably large number of keyprhases, its ranking procedure is enticed.

From the above discussions, it could be concluded that existing unsupervised keyphrase extraction techniques experience several limitations. Moreover, none of the existing technique provides flexibility during keyphrase extraction (Tomokiyo & Hurst, 2003). Flexibility can provide various important keyphrases from a tree depends on researcher interest or needed. Therefore, proposing a novel flexible unsupervised keyphrse extraction technique remains an open research issue; and hence, it is taken into account in this thesis.

1.4 Research Questions

From the above discussions (in Section 1.2 and Section 1.3), three major research questions could be identified, which are mentioned below:

- How to design a unsupervised keyphrase extraction technique — which could extract quality keyphrases flexibly from an article?

- How to design a novel keyphrase ranking technique that could select the most representative $Top-N$ (where $N \in \mathbb{Z}+$) keyphrases from an article?

- How could the performance of the proposed unsupervised keyphrase extraction technique be measured?

1.5 Research Objectives

The primary objective of this research is to design an efficient unsupervised keyphrase extraction for the academic literature. To attain this primary objective, following sub-objectives are needed to accomplish:

- To design an unsupervised keyphrase extraction technique for extracting quality representative keyphrases flexibly from an article.

- To develop a novel ranking technique for selecting the most representative $Top-N$ (where $N \in \mathbb{Z}+$) keyphrases.

- To evaluate the performance of the proposed keyphrase extraction technique through implementing on a benchmark dataset.

1.6 Research Scope

The aspects of the research problem or relevant subject-matter that are tackled (also known as research scope) in this thesis are mentioned below:

- The proposed keyphrase extraction technique is a domain independent approach; and hence, could be applied in most *research domain*, such as computer science, chemical engineering, mathematics, and so forth.

- During the evaluation of the proposed keyphrase extraction technique, only the most relevant existing keyphrase extraction techniques are taken into account.

- During the evaluation, a benchmark research paper dataset is employed, called ”SemEval 2010”. This dataset is the most widely used dataset around the world for analyze automatic keyphrase extraction techniques.

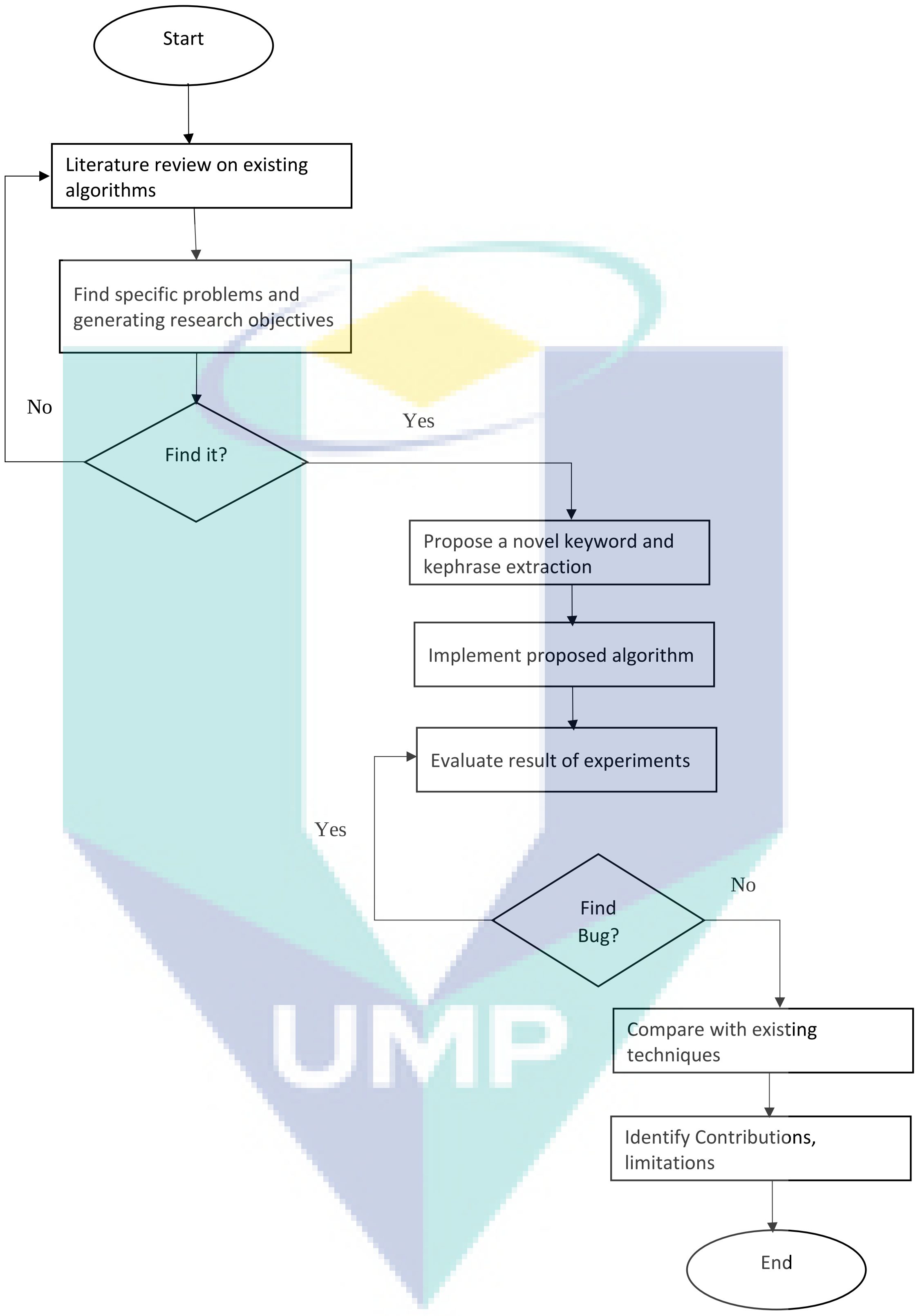
1.7 Research process flow

The research process flow that is employed in this thesis is comprised of several steps (see Figure 1.1), *i* analyzing the existing keyphrase extraction technique and the existing academic recommender system, *ii*) finding specific problems and generating research objectives, *iii*) proposing a novel Keyphrase extraction technique and a novel recommendation algorithm, *iv*) implementing the proposed technique and the algorithm, *v*) performing initial testing on a dataset, and *vi*) concluding by stating the limitations of the proposed scheme and by noting possible future work opportunities. All these steps are briefly detailed below:

Literature review on the existing techniques and algorithms: This thesis work starts with a deep critical investigation on the existing keyphrase extraction techniques. In this process, some nearly relevant techniques are implemented and scrutinized to identify their benefits and limitations. Afterwards, their performances are analyzed and compared with each other to identify their drawbacks and shortcomings. For more details, this reader is requested to read Chapter 2.

Problem statements and research objectives: After investigating the existing systems and algorithms, some major problems are identified, which are mentioned in detail in Section 1.3. In a nutshell, most of the existing keyphrase extraction techniques are not domain independent, fail to extract quality keyphrases, do not offer flexibility during keyphrase extraction, and so on.

Proposing a novel kephrase extraction technique: In this thesis, a novel keyphrase extraction technique is proposed for extracting quality keyphrases from academic literature. The proposed technique employs two prime features, namely popularity and completeness, to find quality keyphrases. In this process, a modified tree data structure and a limited statistical knowledge are utilized. Details of the proposed keyphrase extraction is mentioned in Chapter 3.



Start
Literature review on existing algorithms
Find specific problems and generating research objectives
No
Find it?
Yes
Propose a novel keyword and kephrase extraction
Implement proposed algorithm
Evaluate result of experiments
Yes
Find Bug?
No
Compare with existing techniques
Identify Contributions, limitations
End

Figure 1.1. Research process followed in this thesis

Afterwards, these keyphrases are utilized as the features of an article. Later, they are used to find most relevant references to the recent paper. If the same procedure is applied on 1st level references, such 2nd level reference can be gathered.

Implementing proposed techniques: The implementation is done using using Python programming language tool. The implemented algorithms are applicable for all versions of Python framework. The IDE (also known as *integrated development environment*) used in this implementation is *Jupyter Notebook*.

Experimental Evaluation: For evaluating the effectiveness of the proposed techniques, namely a novel keyphrase extraction technique, they are applied on an actual academic dataset. For this process, a set of testing (*pytest*) is performed to confirm that it is going to the right direction. Changes and modifications are made if the performance do not reach to the satisfactory level. This process continues until the performance reaches to an acceptable level.

Comparing with existing techniques: The proposed techniques are compared with the existing techniques to discover their advantages and disadvantages. In this process, all the compared techniques are either collected or implemented using pyhon; and they are tested on the same dataset with identical parameters, which is utilized to test the proposed scheme. Afterwards, the acquired results are compared with that of the proposed technique. The detail results are demonstrated in Chapter 4.

Identify Contributions, limitations and possible future works: At the end, contributions of the thesis and limitations of it are identified. The notable contributions of the thesis are: i) design and implementation of a novel keyword and keyphrase extraction technique. The notable limitation of the proposed techniques are discovered and noted in Chapter 5. The future works of this thesis also have been identified and stated at the end of the Conclusion chapter.

1.8 Outline of the Thesis

This thesis contains five chapters namely Introduction, Literature Review, Methodology, Results and Discussions, and Conclusion.

Chapter I introduces the work along with the research background, research statements, problem statement, objectives, and research scopes. This chapter portraits a clear motivation/goal behind the current work and the step by step objectives to achieve that goal. It also clarifies where and where not the proposed techniques can be applied and the limitations of them.

Chapter II presents the related works, where most of the description of keywords and keyphrase extraction and describe detailed with the advantages and limitations. In this process these algorithms are keenly analyzed and scrutinized, and then the results of the investigation are summarized in the relevant tables. At the end, a critical analysis is performed to demonstrate the possible research direction which leads us to the current research that is conducted in this thesis.

In chapter III, research design and methodology are mentioned, where the proposed technique is detailed along with necessary algorithms. Besides this, the detail of the experimental setup and survey setup are also elaborated in this chapter. All the performance evaluation parameters are also discussed with adequate detail.

Computational results and discussions on evaluating the concept by choosing the different criteria are given in chapter IV. Results of an experimental data are discussed and indicated a distinct decision based on the results. The data set is extracted from the "*SemEval-2010*". These data are being analyzed according to the scholarly dataset. The validity threats are also discussed, which debates the suitability of utilizing various algorithms.

Finally, in chapter V, the contributions, limitations and conclusion are provided, including the main finding of the work and the outline of the future direction.

CHAPTER 2

LITERATURE REVIEW

2.1 Preamble

Although, a considerable number of keyphrase extraction techniques are proposed in the literature, only relevant ones are discussed in this chapter for providing an extensive overview of the analogous techniques. Alongside this, the relevant techniques are critically analyzed in this chapter to identify their limitations; and thus, identify their research gaps. This process starts with the definition of keyphrase in the subsequent section.

2.2 Definition of Keyphrases

The term ”keyphrase” seems to be a well-understood concept at present due to their utilization in search engines since they are necessary as input to carry out a search. Literally, a keyword means a single word and a keyphrase means a set of separate words that build a phrase. In many applications including academic literature, both terms are considered as synonyms of each other and prefer the term ”keywords” since it is shorter. However, most of the researchers in this area prefer the term ”keyphrase” or ”keyphrases” instead. They can be defined as follows:

”Keyphrases give a high-level description of a documents contents that is intended to make it easy for prospective readers to decide whether or not it is relevant for them.”

— Frank, Paynter, Witten, Gutwin, and Nevill-Manning (1999b)

It also can be defined as:

”Keyphrases provide semantic metadata that summarize and characterize documents.”

— Witten, Paynter, Frank, Gutwin, and Nevill-Manning (2005)

From the aforementioned definitions, it could be synopsized that keyphrases are necessary in summarizing and characterizing documents. Therefore, it is employed in several applications, which are briefly mentioned in Section 2.3. However, the primary challenge in selecting keyphrases from a document lies in the judgment of a term as a keyphrase. To facilitate this process, several keyphrase extraction techniques are proposed, which are discussed elaborately in Section 2.4.

2.3 Applications of keyphrases

As stated earlier in the Section 2.1, keyphrases can support a range of linguistic intelligence tasks, such as information retrieval, contextual advertisement, recommender system, summarization, and so on; which are discussed below:

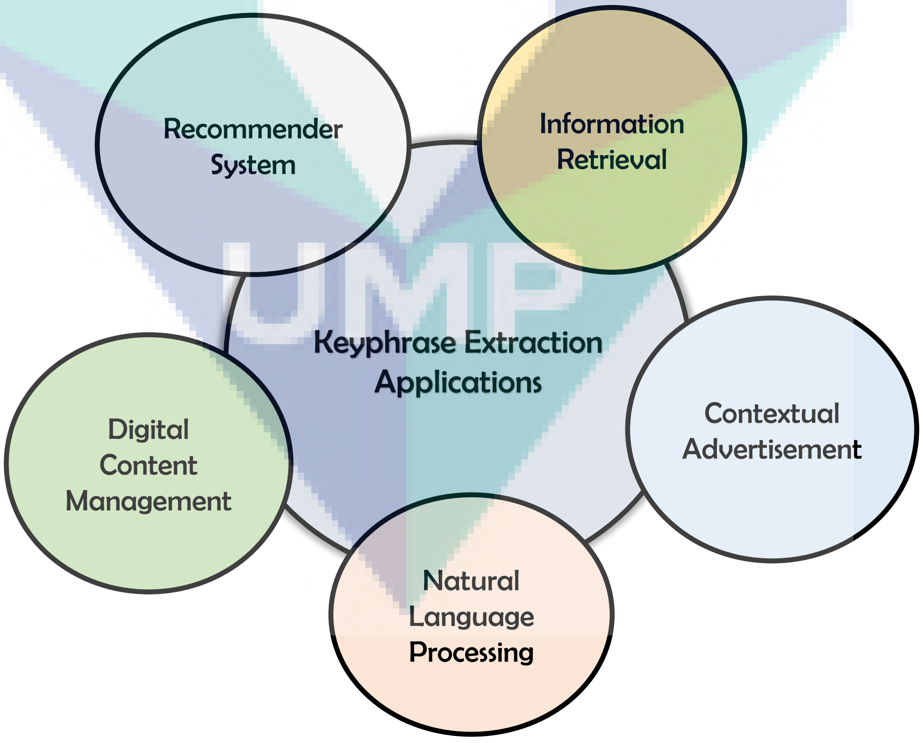

Recommender System
Information Retrieval
Keyphrase Extraction Applications
Digital Content Management
Contextual Advertisement
Natural Language Processing

Figure 2.1. Prominent applications of keyphrase extraction

Recommendation System:

A recommendation system attempts to predict the ”rating” or ”preference” to a user depends on his/her interest. These systems are utilized in various applications, such as movies, musics, news, books, research articles, and so forth. In addition, there are also recommender systems for specialists (Girardi & Marinho, 2007), collaborators (Yager, 2003), jokes (Massa & Avesani, 2007), restaurants (Adomavicius & Tuzhilin, 2005), garments (L. Wang, Zeng, Koehl, & Chen, 2015), financial services (Felfernig & Kiener, 2005), life insurances, and so on.

Generally, a recommender system provides a list of recommendations employing one of the two filtering techniques, namely (*i*) collaborative filtering (Sarwar, Karypis, Konstan, & Riedl, 2001) and (*ii*) content-based filtering (Pazzani, 1999); where, collaborative filtering techniques build models based on users' previous behavioral pattern and content-based filtering methods use a set of discrete characteristics of an item in order to recommend new items with similar properties. In the latter technique, keyphrases are employed to describe the items and linked them with the the user's profiles. A widely used algorithm for content-based filtering is called TF-IDF and the representation of TF-IDF is also called vector space representation. In the former technique, various candidate items are matched with items previously rated or reviewing by the user and the best-matching items are recommended.

Information Retrieval:

Keyphrase extraction plays an important role in the domain of information retrieval. It serves as a minimalistic summary for single documents or document collections, enabling the reader to quickly perceive the main contents of a text.

A collection of documents always indexed by a set of features. In a text based information retrieval system, words, phrase or manually assigned vocabulary items are also use as a feature. When one has chosen a feature set, two approaches to retrieval are exact match methods and ranked methods. In the ranked retrieval methods retrieve a ranked list of documents rather than an unordered set. This allows these methods to take advantage of the fact that some features are batter discriminators than other due to their occurrence statistics. Term Frequency (TF) (Salton & Buckley, 1988b) and Inverse Document Frequency (IDF) (S. Robertson, 2004) are the two useful measures for determining the importance of a feature. The Term Frequency of a word is a function of the number of occurrences of that word in a given document and Inverse Document Frequency of a word that proportion of documents that a word occurs in. TF-IDF is an appropriate fashion, that provide a very useful ranking of words or phrases from documents.

Depends on the TF-IDF, there are two widely used classes for Information retrieval models, probabilistic models (Lafferty, McCallum, & Pereira, 2001) and vector space model (Salton, Wong, & Yang, 1975). Most probabilistic models follow the probability ranking principal (S. Robertson, 2004). This means that documents are ranked according to the probability of being relevant to the information need of the user.

Contextual Advertising:

Contextual advertising is an idea of targeted advertising for advertisements appearing on websites or different media. The advertisements are chosen and followed by automated systems based on the identification of the user and the content displayed. A contextual advertisement method scans the texts from a website for keyphrases and delivers advertisements to the webpage based on those keyphrases. (Dean, Harik, & Bucheit, 2010). They may be displayed on the website or appeared as pop-up ads. Consider as, if the user is visiting a website related to tourism and that website utilizes contextual advertising (Giguere, 2005), the user may see other related advertisements of relevant companies, such as Booking.com, Traveloka, and/or so on. Contextual advertising (Giguere, 2005) is also used by search engines to display ads on their search results pages, which are based on the keyphrase in the user's inquiry. Therefore, it is called as ”In-Text” promotion or ”In-Context” automation.

Contextual advertisements and its dot effect are less critical than conventional advertising. It also influences the users more efficiently. It displays the ads on the area of interest of a user thus increasing the chance of getting a response. The first major contextual advertising network was Google AdSense (Giguere, 2005). It using webmasters with JavaScript code, if entered into web pages, displays related ads from the Google list of promoters. The importance is calculated by a different Google bot called Mediabot. Recently, service providers have developed also complex systems that use a language-independent concurrence pattern finding algorithms to improve matching accuracy (Turtle, 1995). Media.net is also using for contextual advertisement network competing with Google Adsense (Laursen, Olkin, & Porter, 1994).

Digital Content Management:

A digital content management system determines associations between the base content and the relevant content and for publishing the base content and the relevant content on a client browser. Herein, the relevant content is relevant to the user or more specifically, formed of one or more keyphrases in a user profile.

The base content might be served to a user browser along with the relevant content. Embodiments of the invention serve base content to a user via her client system and with the base content, additional content is served that is relevant to the user. Base content generally includes content requested by a user and may be served on a web page visited by the user via the user's client system. The base content might further include the web page on which requested content is published. Additional content might include advertisements and links to content that are placed on the visited web page. Additional content may be deemed relevant to the user if the additional content is substantially similar to attributes of a user profile associated with the user. The attributes might include keyphrases, units, categories and the like that are identified in queries the user uses to query a document corpus and the search results returned to the user. The additional content may be substantially similar to the user profile if the additional content includes or is associated with the same or similar keyphrases, units, categories or the like included in the user profile. An apparatus configured to associate base content and additional content that is relevant to a user is described first below, and a method for associating base content with additional content that is relevant to the user is described thereafterr (Han, 2004; Judd, Brewster, Melia, & Lilly, 2006).

2.4 Keyphrase Extraction Techniques

The existing keyphrase extraction techniques could be broadly classified into several classes, such as domain specific, language specific, statistical, machine learning etc. (Broder, Fontoura, Josifovski, & Riedel, 2007; Chien, 1997; Han, 2004; Hulth, 2003b; Huynh & Hoang, 2012; Mihalcea & Tarau, 2004). Again, most of the machine learning based prominent techniques could be classified into supervised and unsupervised techniques. Furthermore, unsupervised approaches could be divided into graph-based and statistical techniques (see Figure 2.2).

As can be seen in Figure 2.2 is that the initial steps of all the machine learning based keyphrase extraction techniques, i.e., the candidate keyphrase generation process is similar; but the extraction processes are different. In case of supervised techniques, relevant features are extracted from the candidate keyphrases; whereas, this is absent in unsupervised techniques. Again, for unsupervised techniques, train data are not necessary; whereas, for supervised techniques, they are compulsory. Note that train data are manually annotated keyphrases from an application domain. The last phase of any keyphrase extraction technique is ranking or classifying. This is the most important phase since it decides which candidate keyphrases will be finally selected. On many occasions, it has been observed that existing techniques differ from each other based on their ranking and/or classification approaches. Therefore, keyphrase extraction is also accounted as a ranking problem.

Although, there exist several supervised keyphrase extraction techniques; however, they are overlooked in this thesis since the proposed technique is an unsupervised machine learning based technique; and only similar approaches are investigated in this chapter. Again, a brief overview of the existing supervised keyphrase extraction techniques and the justifications of overlooking these techniques are mentioned in the subsequent section.

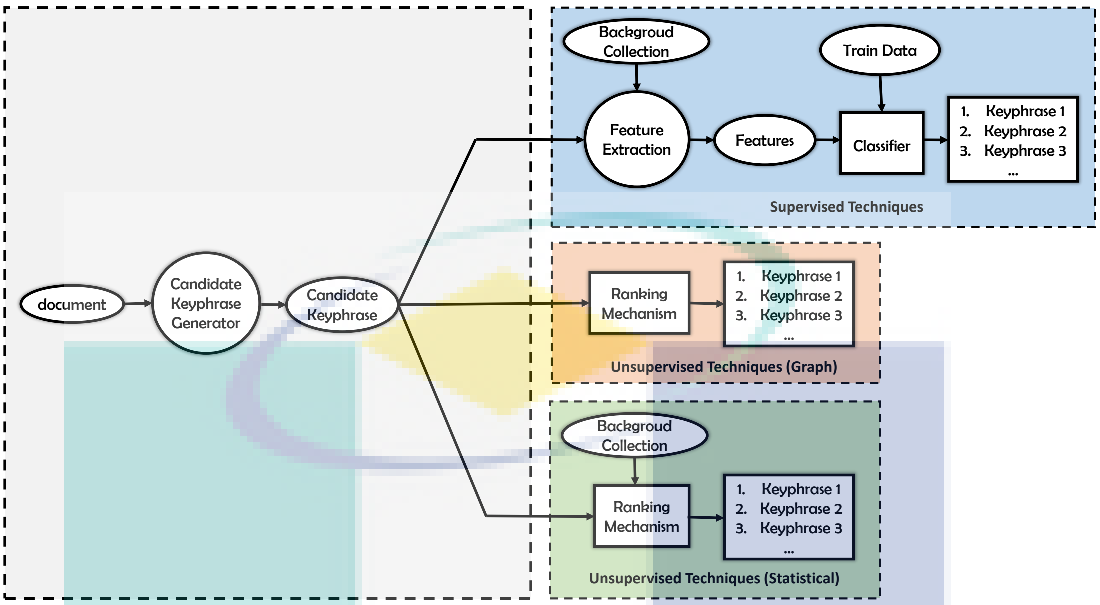


Figure 2.2. Functional details of various machine learning based technique for keyphrase extraction

2.4.1 Supervised Keyphrase Extraction Techniques

As mentioned in the previous section is that there are two classes of machine learning based techniques utilized in keyphrase extraction, namely supervised and unsupervised (Joachims, 1998; Ohsawa et al., 1998). The principal difference between these two techniques is that in supervised (Joachims, 1998) techniques, the output values for sample inputs are determined based on prior knowledge. Conversely, in unsupervised techniques, no labeled outputs are present; and hence, its goal is to understand the physical structure present within a set of data points. In details, supervised machine learning techniques utilize training data as input variables, called features (Pang, Lee, & Vaithyanathan, 2002) and predict the output values based on their features only.

The keyphrase extraction problem using supervised techniques from a document is calculated as a binary classification problem (Joachims, 1998; Narasimhamurthy, 2005; Ohsawa et al., 1998), where some fraction of candidate phrases are classified as keyphrases and the others as non-keyphrases. Support vector machines (Joachims, 1998), naive Bayes

(McCallum et al., 1998) and decision trees (Quinlan, 1986) are prominent methods to solve the classification problem.

It has been observed in several studies (Kotsiantis, Zaharakis, & Pintelas, 2007) that supervised techniques perform better than unsupervised techniques when the systems are trained with a large dataset. However, a large training data is difficult to find for keyphrase extraction due to the presence of enormous number of classes and levels of documentation. Again, the risk of training a model that does not induce to unseen examples need constant guarding; otherwise, would perform unsatisfactorily. Moreover, training dataset could be jeopardized due to the limitations of the human being since humans do not judge keyphrases independently. They just judge certain candidate phrases in an intrinsically relevant knowledge. Furthermore, some supervised approaches employ a wide variety of features to distinguish between keyphrases and non-keyphrases; and in reality, the list of possible features is exponential. With increasing number of features, time complexity of a technique also increase; and therefore, it takes a considerably long time to train the system.

Due to the aforementioned constraints of the supervised machine learning based techniques, in this thesis, the unsupervised machine learning based techniques are taken into account, which are discussed elaborately in the following section.

2.4.2 Unsupervised Keyphrase Extraction Techniques

Unlike supervised machine leaning techniques, unsupervised machine learning techniques do not require any test data. Generally, these techniques recognize commonalities or patterns in the dataset and act based on the appearance or absence of such commonalities in the various new part of the dataset. This class of techniques could be further divided into two groups, namely graph-based techniques and statistical techniques. The prominent techniques of both groups are discussed with adequate details in Section 2.4.3 and 2.4.4.

2.4.3 Graph-based Unsupervised Techniques

The graph-based techniques generate weighted or unweighted graphs from raw research data like text or XML and analyze afterwards to extract keyphrases. In other words, these methods generate a graph based on the words within a document, and then, provide scores based on several criteria, and at the end, rank the words or keyphrases according to their scores. The top ranked keyphrases are returned as the desired keyphrases.

In literature, a considerable number of graph-based techniques are proposed. Among them, KeyGraph (Ohsawa et al., 1998), PageRank (Florescu & Caragea, 2017a), TextRank (Mihalcea & Tarau, 2004), and PositionRank (Florescu & Caragea, 2017b) are the most prominent techniques, which are discussed below.

2.4.3.1 KeyGraph

In KeyGraph, only representing keyphrases are extracted without considering any other external methods, such as natural language processing tools (Reilly & Sharkey, 2016), document corpus etc. Afterwards, a graph is generated depends on the extracted keyphrases. Segmentation of the graph into different clusters is the main theme of this algorithm with characterizing the co-occurrence between words in a document. Each cluster represents an idea of an author(s), and the top-ranked terms in these clusters — which are based on the relationship of each term — are selected as keyphrases.

The main phases of this technique are: document preparation, phase extraction foundations and preparatory concepts that are obtained as clusters, relationships between terms in the document, and keyphrase extraction. In document preparation phrase, at first, non-significant words like stop words (i.e., ”a”, ”and”, ”the”, and so forth) — which contribute a little in a document summarization — are removed (Hulth, 2003b). Afterwards, all the remaining words are stemmed to bring related words to the identical root. For instance, the terms — ”work”, ”working”, ”works” are all reduced to ”work” by using steaming process (Willett, 2006). Then, the chains of words that are connected by non-significant terms and stems convert candidate phrases.

In the second phase, candidate keyphrases are generated from a keyphrase and added in a list. For this, the combination of multiple words and their sequence of appearances are taken into account during the process. For instance, if a keyphrase is *ABCD* with the terms *A*, *B*, *C*, and *D*, following candidate keyphrases will be generated: *ABC*, *AB*, *BCD*, *BC*, and *CD*. Depending on the frequency of these candidate keyphrases, they are sorted and each of them is treated in turn.

If any phrase with the current phrase has a frequency equal to the current one, the current one is removed. In another way, all the phrases including the current one are cast out. This system pulls out larger phrases with higher frequency.

A graph for a document is constructed by nodes of representing terms, and links describing the co-occurrence. Nodes in the graph describe high-frequency words in a document and create the candidates like foundation because words may appear repeatedly for proving basic ideas in the domain. Then, subgraphs corresponding to the fundamental concepts. This construction is depends on the idea that connections among terms within a document and create the semantic coherence or the underlying idea in a portion of the document. This concept that underlying ideas are implemented by the relationship among terms in a document (Lau, Song, Li, Cheung, & Hao, 2009; Ohsawa et al., 1998) has been applied for segmenting a document into semantically coherent portions (Liu, Pennell, Liu, & Liu, 2009), depends on the location of words in the document. However, choosing the most significant portion is only work for scientific documents. Also, several works used the co-occurrence data for indexing (Matsuo & Ishizuka, 2004), but the search engine performance was not completely superior to traditional methods as TF-IDF. It means that using co-occurrences to compare queries with local information in a document is not very meaningful.

Terms in a document D_t are sorted depends on the frequencies in D. High-frequency

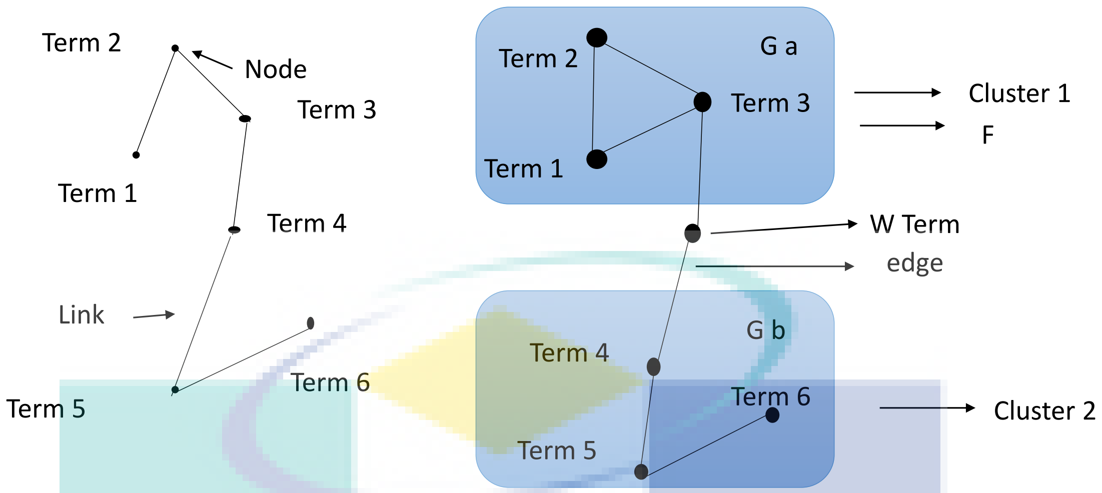


Figure 2.3. Single connected and multiple connected graph G

terms, such as, the group of terms above the $40th$ highest frequency when D_t, consider into G. Terms or words in height frequency F become the nodes in G in Figure 2.3. G is only connected if G_a and G_b must be a connected graph with the number of nodes in G.

In KeyGraph, the relation between terms are defined by the co-occurrence between them (Figure 2.3). A maximal connected subgraph is thinking as a idea based on the author's concept.

Selected terms for the keyphrases represented by the node of term w (Figure 2.3). Defining the tightness of word w holding clusters, they use the following value $key(w)$ to every term w, in a document. $key(w)$ must be a real number between 0 and 1, defined as the probability of term or word w to appear if all the foundations in the graph being considered. $key(w)$ denotes the conditional probability that w is used, for based and neighbors depend on the auxiliary functions. That is,

$$based(\omega,\theta) = \sum_{s\in D} |\omega|_\delta |\theta-\omega|_\delta \qquad 2.1$$

$$neighbors(\theta) = \sum_{s\in D}\sum_{s\in \delta} |\omega|_\delta |\theta-\omega|_\delta \qquad 2.2$$

Next, combine all the high probably candidate phrases as new nodes on the graph. They do not sort terms or words only by the keys, because terms or words which serve foundations completely related to roots are also necessary for summarizing the document. Those especially important foundations for the keys but highly ranked by the total strength of reaching base terms score. Nodes in the graph are sorted by the sum of above from the base terms score. Terms or words described by nodes of the higher score of these sums than a specific threshold are selected as the keyphrases from the document. They select top 12 words or terms. If more then one term are equally ranked, all the same ranked terms or words are extra.

2.4.3.2 PageRank

PageRank (Langville & Meyer, 2011) works as a link analysis algorithm (Ohsawa et al., 1998) and it selects a statistical weight to each element of a hyperlinked set, such as the World Wide Web (Chien, 1997), including its relevant importance within that set. This algorithm has been utilized for any number of individual elements with mutual references and quotations. The statistical weight can be assigned to any element within X and constructed the PageRank of X or $PR(X)$. The web graph algorithm is the main mathematical base for the PageRank and web graph considers all World Wide Web (Chien, 1997) pages as nodes and hyperlinks as edges. Importance of a particular page is indicated by the rank value. Page counts are used as a score for a hyperlink. The PageRank of a page is described recursively and PageRank metric (Langville & Meyer, 2011) is created with all pages that connect to it. A page that is connected with many pages considers as high rank in PageRank (Langville & Meyer, 2011). The PageRank concept is vulnerable for manipulation and a various investigation has been conducted to identify incorrectly modified PageRank rankings.

In Figure 2.4, Page C has a higher PageRank value than Page E, even though there are several links to E; the only one link to C that comes from an important page and hence C is carrying a large value. Web surfers who start on a random page have 85% probability of choosing a random link from the page they are currently visiting, and 15% probability

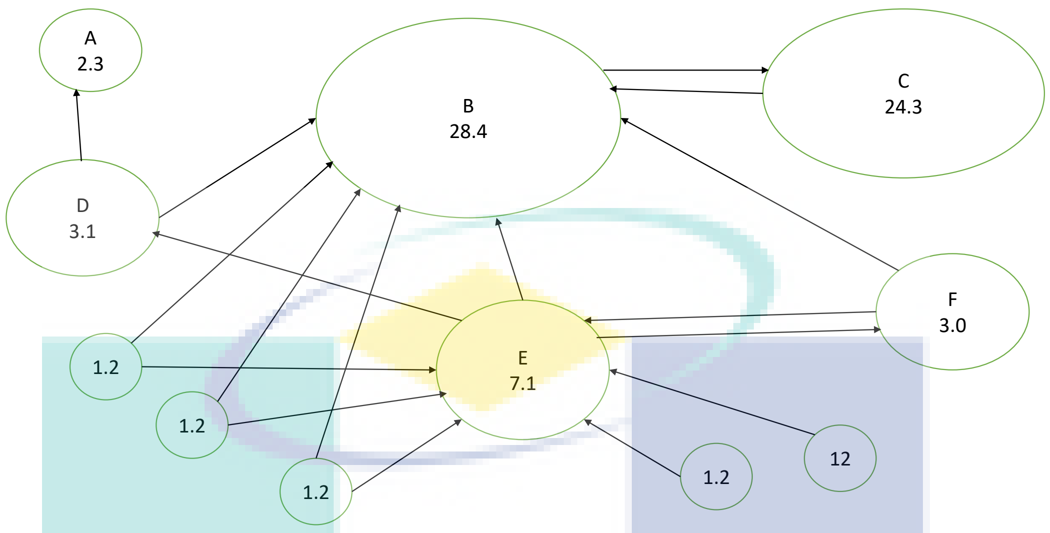


Figure 2.4. Mathematical PageRanks for a simple network

of jumping to a page chosen at random from the whole web, they will reach Page E 8.1% of the time. All web surfers would finally end up on Pages A, B, or C, and all other pages would have PageRank zero. Page A efficiently links to all pages in the web, even though it has no outgoing links of its own.

The PageRank algorithmic rule used the probability distribution to represent the possibility of a web surfer randomly clicking on links will arrive at any particular page. PageRank can be calculated by any size of compendium document. The distribution is evenly divided among all the documents. The PageRank computing iterate through the in gathering to adjust approximate 6 senses of value to more closely predict the theoretical true value. This probability is expressed as a numeric value between 0 and 1. Hence, the PageRank score of 0.5 quintet means there is a 50% chance that a person clicking on a random link will be conducted to the document.

Assume a small phrase of four web page: $A1$, $A2$, $A3$, and $A4$. Links from a web page to itself are ignored and multiple outbound connections from one web page to another page are treated as a single link. The values are initialized for all web pages. The sum of PageRank over all pages was the cumulative number of pages on the web at that time and each page in this discussion would consider an initial value of 1 with the chance of distribution between 0 and 1. If the only links in the system were pages $A2$, $A3$, and $A4$ connected to $A1$ and each link would carry- over 0.25 PageRank value to $A1$ upon the next loop, for a total of 0.75.

$$PR(A1) = PR(A2) + PR(A3) + PR(A4) \tag{2.3}$$

Again, if $A2$ had a connection to pages $A3$ and $A1$, page $A3$ had a connection to page $A1$, and page $A4$ had connections to $A1$, $A2$, $A3$. In the first iteration, page $A2$ would transfer half of its current value, or 0.125, to page $A1$ and the other half, or 0.125, to page $A3$. Page $A3$ would share all of its current value, 0.25, to the single page it links to, $A1$. Since $A4$ had three outbound connections, it would share one-third of its current value, or approximately 0.083, to $A1$. At the end of this iteration, page $A1$ will have a PageRank value will be approximately 0.458.

$$PR(A1) = \frac{PR(A2)}{2} + \frac{PR(A3)}{1} + \frac{PR(A4)}{3} \tag{2.4}$$

We can also say PageRank connects by an outbound connection is similar to the documents. Consider the number of outbound connections L.

$$PR(A1) = \frac{PR(A2)}{L(A2)} + \frac{PR(A3)}{L(A3)} + \frac{PR(A4)}{L(A4)} \tag{2.5}$$

In the general case, the value for any page P can be expressed as:

$$PR(p) = \sum_{v \in B_p} \frac{PR(v)}{L(v)} \tag{2.6}$$

The score for a page p is dependent on the score for every page v included the set B_p where the set including all pages connecting to page p and divided by the number $L(v)$. $L(v)$ is the number of connections from page v. Depends on the PageRank method when a user clicking randomly on links. From the probability in PageRank, a damping factor d will continue at any stage of the calculation. Various experiments have experimented various damping factors for calculation but it is assumed that the damping factor will be 0.85 (Brin

& Page, 1998). For the biological dataset, a Bayesian study discovers the optimal value of d to be 0.31 0.31 (Page, 2001).

$$PR(A1) = \frac{1-d}{N} + d\left(\frac{PR(A2)}{L(A2)} + \frac{PR(A3)}{L(A3)} + \frac{PR(A4)}{L(A4)} + \cdots\right) \qquad 2.7$$

$$PR(A1) = 1-d+d\left(\frac{PR(A2)}{L(A2)} + \frac{PR(A3)}{L(A3)} + \frac{PR(A4)}{L(A4)} + \cdots\right) \qquad 2.8$$

The difference between 2.5 and 2.6 is that, the PageRank values in 2.5 sum to one, while in 2.6 every PageRank (Litvak & Last, 2008b) value is increased by N and the total also converts by N. From the Page and Brin's paper (Brin & Page, 1998) that "the sum of all PageRanks is one" (Page et al., 1999) and claims by different Google representatives (Page et al., 1999) recommend the first modification of the method above. They are confused in 2.5 and 2.6 in their most famous paper "The Anatomy of a Large-Scale Hypertextual Web Search Engine", where they mistakenly declared that the recent formula created a probability distribution across web pages (Brin & Page, 1998). For calculating PageRank score, no outbound connections are allowed to link out to all different pages in the collection and the scores are distributed equally among all different web pages.

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)} \qquad 2.9$$

$p_1, p_2, ..., p_n$ are the web pages under consideration where $M(pi)$ is the collection of web pages that have a connection to p_i and N is the total number of pages. The PageRank score is the entry of the principal right eigenvector of the transformed adjacency matrix rescaled with every column adds up to one. This makes an especially simple metric. The eigenvector is,

$$\mathbf{R} = \begin{bmatrix} PR(p_1) \\ PR(p_2) \\ \vdots \\ PR(p_N) \end{bmatrix} \qquad 2.10$$

$$\mathbf{R} = d\mathcal{M}\mathbf{R} + \frac{1-d}{N}\mathbf{1} \tag{2.11}$$

Then,

$$\mathbf{R} = (\mathbf{I} - d\mathcal{M})^{-1}\frac{1-d}{N}\mathbf{1} \tag{2.12}$$

$\mathbf{I}$ is the identity matrix. The solution exists and is individual for $0 < d < 10 < d < 1$. $\mathcal{M}$ is constructed by a stochastic matrix and hence has an eigenvalue corresponding to one as an outcome of the *Perron Frobenius* theorem.

2.4.3.3 TextRank

TextRank is a graph-based keyphrase extraction technique for text processing and being successfully used in various applications. For the TextRank, two innovative unsupervised techniques are used for keyphrases extraction and found that the results obtained are better than previously published results on established benchmarks.

Graph-based ranking algorithms are the correct way of finding the importance of a vertex within a graph, depends on the global information recursively extracted from the entire graph. The core idea was implemented by a graph-based ranking model for scoring or recommendation. When one vertex connects to another, it basically calculates a score for another vertex. Higher scores demonstrate the importance of the vertex. Furthermore, the importance of the vertex forming the score and consider that, how important the score itself is, and this information is also exerted into account by the ranking method. Hence, the score compared with a vertex and determined based on the scores that are cast for it, and the final score of the vertices casting these scores. The TextRank algorithms described, depends on an algorithm called Googles PageRank (Brin & Page, 1998) and other graph-based ranking algorithms such as HITS (J. C. Miller et al., 2001). Positional Function (Mihalcea, 2004) can be efficiently integrated into the TextRank model (Mihalcea & Tarau, 2004).

Although most of the keyphrase extraction techniques have used directed graphs and also a recursive graph-based ranking algorithm can be applied to the undirected graphs. For this case, the outdegree of a vertex is related to the indegree of the vertex. Sometimes in connected graphs, the number of edges is equivalent to the number of vertices and in the undirected graphs to have more regular convergence curves. In the TextRank model, graphs built are depended on the natural language texts and also can include multiple or partial links between the vertices that are extracted from the text. It is also valuable to indicate and incorporate into the model the ”strength” and the connection between the two vertices as a weight added to the similar edge for connecting the two vertices.

To enable the application of graph-based ranking algorithms to natural language processing, need to build a graph that represents the text and interconnects words or other text items with meaningful connections. Depending on the various applications with text units of different sizes and properties, it can be added as vertices in the graph such as words, entire sentences, collocations or others. Similarly, it handles the type of links that are used to draw the connections between any two vertices, such as contextual overlap, lexical or semantic relations, etc.

The application also manages the relations between any two such vertices, such as contextual overlap, lexical or semantic relations, etc. Also, the types and characteristics of the elements added to the graph and the graph-based ranking algorithms to natural language processing consist of the following main steps: (1) Recognize the text units those are best and add those text units as vertices in the graph. (2) Recognize the relations in the connect (text units), and use those relations to draw edges between the vertices within the graph. Edges can be directed or undirected, weighted or unweighted. (3) Repeat the graph-based ranking algorithm till convergence. (4) Sorting the vertices depends on the final score. The values are assigned to every vertex for selection.

To investigate and judge the method of TextRank with two natural language processing tasks including the ranking of text units: (1) keyphrase extraction is consisting of the selection of keyphrases from a text; (2) A sentence extraction task is selecting the most meaningful sentences from a text. Those sentences are also used to develop summaries the text.

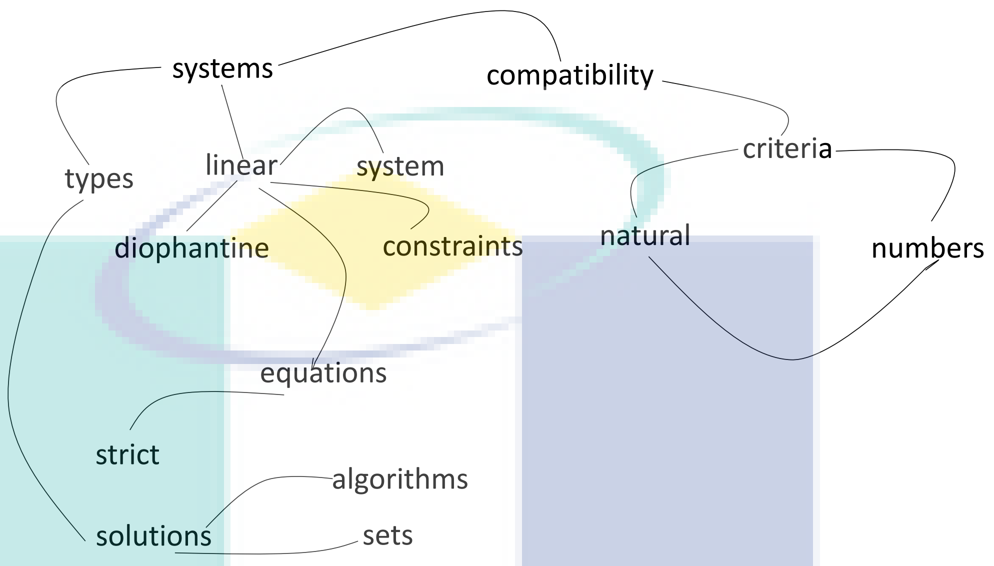


Figure 2.5. Sample graph build for keyphrase extraction in TextRank

In Figure 2.5, presents a graph constituted from an abstract. Most of the time, the length of the abstracts ranges from 50 to 300 words and the average length is 100 words. For example, the lexical units create a higher ”importance” by the TextRank algorithm: linear (2.29), numbers (2.46), equations (2.45), diophantine (2.28), strict (0.77). Notice that ranking is different from the simple word frequency techniques. For the same text, we assume that a word frequency method provides the following top- ranked lexical units: systems (4), types (3), solutions (3), linear (2), equations (2), algorithms (2). Other lexical units contain frequency 1 and it cannot be ranked, but listed.

The responsibility of a keyphrase extraction algorithm is to automatically identify a group of keyphrase. These keyphrases may develop a useful entry for creating automatic indexing for a dataset, also possible to classify a document or provide services as a concise summary for a document. Moreover, a method for automatic identification of relevant terms in a document can be applied for solving the problem of language extraction, and develop of domain-specific dictionaries. The simplest possible approach probably uses a frequency criterion to find the ”important” keyphrase from a document. However, this method was usually found to lead to poor results, thus other methods were explored.

2.4.3.4 PositionRank

PositionRank is an unsupervised graph-based keyphrase extraction technique especially using for scholarly documents such as research papers, documentation, etc and it covers information from all positions and frequency of a word. For the unsupervised keyphrase extraction research, keyphrase extraction is considered as a ranking problem with graph-based ranking methods. All the graph-based keyphrase extraction techniques create a word graph from the document and all the words are considered as nodes and edges corresponding to word association patterns for the graph. Nodes are ranked by using the graph similarity measures such as HITS (Litvak & Last, 2008b) or PageRank (Mihalcea & Tarau, 2004; Mihalcea, Tarau, & Figa, 2004), and the top-ranked candidate phrases are counted as keyphrases.

There are three main steps for the PositionRank algorithm: (1) Graph Building (2) Position-Biased PageRank (3) Create Candidate Phrases.

For building the graph, first apply the parts-of-speech filter applying the NLP Stanford toolkit (Manning et al., 2014) and only select nouns and adjectives as candidate phrase or words (Mihalcea & Tarau, 2004). For building the word graph with each different word those passes the parts-of- speech filter considered as a node. Each node is connected by an edge if the words similar to these nodes co-occur within a window. The weight of every edge is calculated depending on the co-occurrence count of the two words. The graph will be directed or undirected but PositionRank is built by undirected graphs. The main approach of PositionRank is to select larger weights to words that are found beginning in a document and also frequent. Specifically, it provides a higher score to a word found on the 5th position as compared to a word located on the $60th$ position in the corresponding document.

They also provide a score for each candidate word with its inverse position in the document. If a similar word arrives multiple times in the document, then they calculate all of its position's weights. For example, if a word is located in the following locations $5th$, $10th$ and $20th$, its weight is $\frac{1}{5}+\frac{1}{10}+\frac{1}{20}=\frac{35}{100}=0.35$. Calculate all the positions weights for a word aims to grant more confidence to frequently occurring words by taking the position's weight of each occurrence. Those candidate words have nearby positions in a document are concatenated into phrases. They consider only the noun phrases that match using the regular expression $(adjective)(noun)+$ and consider the length up to three using unigrams, bigrams, and trigrams. Finally, those phrases are scored by using the total of scores of individual words that contain in the final keyphrase (Turney, 2000). The top-scoring keyphrases are output as predictions.

2.4.3.5 CollabRank (SingleRank)

Nowadays, the keyphrase extraction task for single document works independently without any intercommunications between each document, under the hypothesis that the documents are considered independent of each other. In this method, keyphrase extraction for a single document by collaborative filtering using the mutual impacts of multiple documents within a cluster. This algorithm is implemented in the clustering method for finding the relevant document clusters and then applying the graph-based ranking for collaborative single document keyphrase extraction within every cluster. Various clustering algorithms have been reviewed and found that this method relies positively on the quality of document clusters. For every single document, they use a document set for keyphrase extraction. CollabRank first applies the clustering algorithm to a set of documents. The documents within the individual cluster are needed to be topic-related and each cluster can be counted as a context for any document in that cluster. For the document clustering, this algorithm depends on the global word relationships in the cluster to judge and rank candidate phrases for every single document in the cluster based on the graph-based ranking algorithm.

The CollabRank algorithm includes two principal steps: (1) Document Clustering (2) Collaborative Keyphrase Extraction.

For document clustering, several clustering algorithms will create different clusters. The documents in a high-quality cluster are normally deemed to be highly topic-related, where documents from a low-quality cluster are normally not topically related such as inappropriate cluster context. The feature of a cluster will influence the reliability of contextual data for estimating the information in the cluster. A number of clustering algorithms are observed for the experiments, including the agglomerative algorithm (Bougouin et al., 2013) including both average-link (Page, 2001) and complete-link (Page, 2001), the divisive algorithm (G. A. Miller, 1995) and the *Kmeans* algorithm (Zha, 2002). In the collaborative keyphrase extraction, all the candidate phrases are in a cluster depends on the graph-based ranking algorithm. The global link graph points to join the cluster-level cooccurrence relations between all the candidate phrases in the documents in a cluster. The final scores of every word are calculated depending on the global affinity graph to show how much knowledge about the main topic in the phrases displays. Depending on the cluster-level phrase scores, the candidate phrases of every individual document is to evaluate and select some outstanding phrases as keyphrases of the document. Cluster depends on the graph-based keyphrase extraction and the ranking algorithm works on all documents in a cluster in order to evaluate the words from a global perspective while the evaluation of candidate phrases is applied on every single document in order to extract keyphrases from a perspective. Extremely remarkable phrases also can be probably included with keyphrases in every document. From their analysis, the keyphrase extraction tasks are conducted in a batch method for every cluster. If clustering is implemented on all single document without considering the cluster context, the method declines the simple TextRank model (Mihalcea & Tarau, 2004). For CollabRank, the graph is constructed based on the whole cluster and it is described as Global Affinity Graph (Ajmani, Ghosh, Mallik, & Chaudhury, 2013). But for the SingleRank it develops a local graph depends on every single document. SingleRank supports most of the graph building rules for the CollabRank but CollabRank builds a graph of all the documents in a cluster and SingleRank designs a graph only based on a single document.

2.4.3.6 TopicRank

TopicRank is a graph-based keyphrase extraction algorithm that applies for topic-based representation of a document. Depending on topics the candidate keyphrases are clustered and used as vertices for building a complete graph. A graph-based ranking algorithm is applied for a given significance score to every topic. Keyphrases are generated from selecting candidate phrases and each of the top- ranked topics. They conducted an experiment on four evaluation datasets of various languages and domains. TopicRank is an enhancement of the TextRank algorithm applied for keyphrase extraction (Mihalcea & Tarau, 2004). In the TextRank algorithm, a document is expressed by a graph where words are applied as vertices and edges describe co-occurrence relations. A graph-based ranking algorithm is always derived from PageRank (Brin & Page, 1998) and applied to indicate a significance score to every word. TopicRank has represented a document as a complete graph where vertices are not words but topics. They define a topic as a cluster of similar single and multiword expressions.

The TopicRank algorithm requires three main steps: (1) Topic Identification (2) graph-based Ranking (3) Keyphrase Selection.

In topic identification, a topic is normally conducted by more than one noun phrase. Some keyphrase candidates are irrelevant in consideration of the topic they represent. Graph-based keyphrase extraction methods such as SingleRank, TextRank, PageRank, etc do not take that factor into account. Candidate keyphrases are normally independent and the knowledge about the topic they represent is distributed throughout the graph. By this way, a set of similar noun phrases works as a single entity in a topic. They recognize that if two candidate keyphrases have at least 25% of overlapping words, they are similar. Candidate keyphrases are stemmed to conquer their inflected word from the root. For automatically collecting a set of similar candidate keyphrase within topics, they use a ”Hierarchical Agglomerative Clustering”(HAC) algorithm (Murtagh & Legendre, 2014). Between the usually used linkage strategies, those are the complete, average and single linkage, they apply the average linkage because it stands as an agreement between complete and single linkage. Using a deeply agglomerative method such as complete the linkage is more likely to group with irrelevant candidate phrases, although an approach such as a single linkage is hidden likely to group topically related candidate phrases. Again, TopicRank also represents a full document by a complete graph where topics are considered as vertices and edges are scored according to the intensity of the semantic relations between the vertices. Also, TextRanks is used to indicate a significance value for each topic.

Let consider, G is a complete and undirected graph where a collection of vertices is V and the edges are E, So a subset is $V * V$. Topics are considered as vertices and the edge between two topics or vertices are t_i and t_j is weighted according to the strength of their semantic relation. t_i and t_j have an influential semantic relation if candidate phrases often close to each other in the document. Therefore, the score of their edge is,

$$\omega_i, j = \sum_{c_i \in t_i} \sum_{c_j \in t_j} dist(c_i, c_j) \qquad 2.13$$

For ranking,

$$S(t_i) = (1 - \lambda) + \lambda * \sum_{t_j \in v_i} \frac{\omega_j i * S(t_j)}{\sum_{t_k \in \text{o}(v_j)} \omega_j k} \qquad 2.14$$

where V_i is the value of the topic for t_i and λ is a damping factor and normally fixed to 0.85 (Brin & Page, 1998).

Using candidate keyphrases as vertices significantly enhance SingleRank on SemEval (Rowley & Hartley, 2017b), WikiNews (Bobadilla, Ortega, Hernando, & Gutiérrez, 2013) and DEFT (Han, 2004). It induces a significant lack of enforcement produced by an important lack of connections that spans connected elements, as shown in Figure 2.6. The graph is split depends on the connected elements and increases the difficulty to select ”fuzzy Bayesian inference techniques” (Yang, 1997) as a keyphrase. For the complete graph, topics are interconnected among each other. The completeness of the graph has the advantage of implementing an extra exhaustive view of the connections between all the

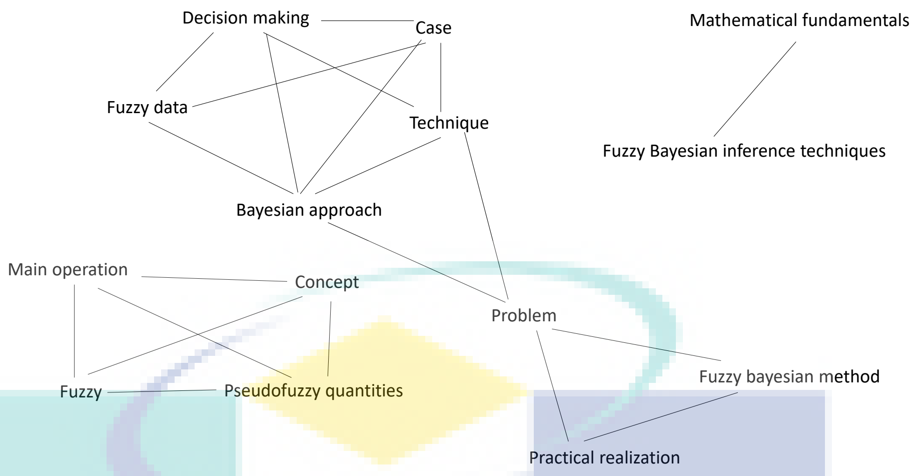


Figure 2.6. Sample graph build by TopicRank

topics. Also, calculating weights depends on the distances between offset, positions avoid the necessity for the manually defined parameter. when the graph is generated, the ranking method TextRank has applied to rank the topics. Depends on the concept of scoring, this method assigns a meaningful score to the topics. High-scoring topics provide more score to the related topics. Keyphrase collection is the ultimate level of TopicRank. Only the most representative candidate phrases are selected from each topic. This selection skips repetition and points to large coverage of the document topics because selecting *n* keyphrases specifically covers *n* topics. To extract the candidate keyphrases which are best for a topic, they suggest three approaches. When a topic is first represented by its general structure, the primary approach is to select the candidate keyphrases that arrives first in the document. The other method assumes that the general form of a topic is the one that is most frequently used and the third strategy is deciding the centroid of the cluster. Those candidate keyphrases that is the most similar to the other candidate keyphrases of the cluster.

2.4.3.7 MultipartiteRank

MultipartiteRank is one of the most important unsupervised keyphrase extraction technique that extracts topical information within a multipartite graph structure. This algorithm expresses keyphrases as candidate phrases with the topics within a single graph and utilizes their complementary relationship to improve candidate phrase scoring. They found a system that incorporates keyphrase selection preferences into the algorithm.

MultipartiteRank algorithm mainly operates in two steps. First, construct a word graph depending on the document and implement a ranking algorithm to allocate a relevance score to each keyphrase. The sequences of adjacent nouns with one or more previous adjectives are used for selecting the candidate keyphrases. Then set into topics depending on the stemming forms the words and using hierarchical agglomerative clustering with average linkage. There are so many different approaches to identify topics, supervised or unsupervised probabilistic topic models is one of them.

Candidate keyphrases are used as nodes to create a complete directed multipartite graph and those candidate keyphrases are connected particularly if they relate to different topics. Weight between two nodes is calculated as the total of the inverse distances between the occurrences of candidates keyphrases. The final graph is a complete $npartite$ graph (Litvak & Last, 2008b) and nodes are distributed into n different independent sets. Here n is the number of topics. The proposed algorithm does not create any hypotheses about the topic collection and provides direct use of any topic decomposition. It implicitly supports the significant number of topics covered in the selected keyphrases by discouraging intra-topic recommendation and captures the mutually reinforcing connection between topics and candidate keyphrases. We can also say that, excluding edges between candidate keyphrases applying to a particular topic proves that the overall recommendation of each topic is divided throughout the whole graph. Also, a benefit of encoding topic relevant candidate phrases differentially is that the ones that best validation for every topic, that is directly given by the model. Deciding the most suitable candidate keyphrases for a specific topic is a complex task, and relying only on their importance in the document is not sufficient (Hasan & Ng, 2014). Between the features introduced to address the problem in the discussion, the position of the candidate keyphrases within the document is most reliable. In order to capture this in the system, they accommodate the incoming edge weights of the nodes communicating to the first occurring candidate keyphrases of all topic. Candidate keyphrases occur at the beginning of the document are developed according to the other candidate keyphrases belonging to the same topic. Note that the selection of the candidate keyphrases is to support, such as the selection heuristic can be adapted to fit other needs such as prioritizing candidate phrases from a dictionary.

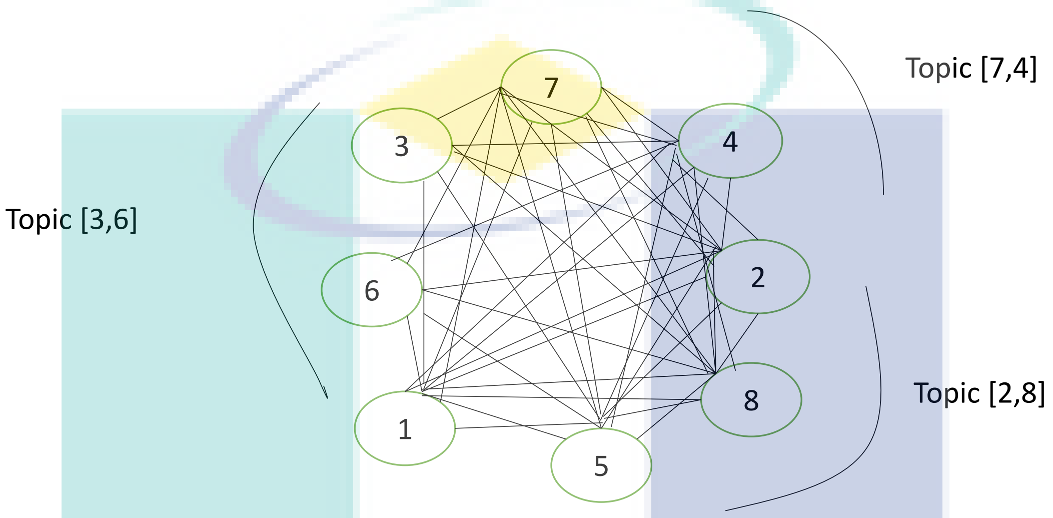


Figure 2.7. Sample graph build for keyphrase extraction in Multipartite graph

In Figure 2.7 a particular graph formation, called multipartite graph describes documents as tightly associated sets of topic associated candidate keyphrases. After the graph built, candidate keyphrases are ordered by a graph-based ranking algorithm, and the top N candidate keyphrases are selected as keyphrases. They also utilize the widely used TextRank algorithm (Mihalcea & Tarau, 2004) which leverages as consider as edge weights:

$$S(b_i) = (1-\lambda) + \lambda . \sum_{b_j \in I(b_i)} \frac{\omega_i j . S(b_j)}{\sum_{b_k \in o(b_i)} \omega_j k} \quad 2.15$$

where $I(b_i)$ is the set of predecessors of b_i, $O(b_j)$ is the set of replacements of b_j , and λ is a damping factor set to 0.85 as in (Mihalcea & Tarau, 2004). Note that different ranking algorithms can also be used. They apply TextRank because it was shown to perform consistently well (Boudin, 2016a).

2.4.4 Statistical Approaches

Mathematical formulas, patterns, and methods are utilized in the statistical analysis of raw research data like text XML. The application of statistical techniques such as (Salton & Buckley, 1988b), KP-Miner (El-Beltagy & Rafea, 2009a), YAKE (Campos et al., 2018b), etc extracts Keyphrases from text or XML data and presents several ways to access the robustness of research products.

2.4.4.1 Term Frequency - Inverse Document Frequency

For the information retrieval system, Term Frequency-Inverse document frequency (TFIDF) (Salton & Buckley, 1988b) is a statistical approach that explains the importance of a word or phrase in a specific document in a corpus or dataset (Salton & Buckley, 1988b). It is usually applied as a weighting factor in the field of information text mining, information retrieval, and user modeling. The TF-IDF score represents the number of times a word or phrase appears in a document and is balanced by the total number of documents in the data set or corpus that contain the word. It helps to adjust to the fact that some words or phrase appear more regularly. TF-IDF is the most widely used term-scoring algorithm and more than 80% of text-based recommender systems in digital libraries use TF-IDF (Salton & Buckley, 1988b). Modifications of the TF-IDF scoring is often used by search engines as a fundamental tool in scoring and ranking documents and importance are given a user query. TF-IDF also applied for stop-words filtering in different fields such as text summarization, text classification, etc.

Think of a group of English text documents or corpus and want to find out the most appropriate document in the ”the recommendation system” query. Fast need to drop the documents those do not contain all three words ”the”, ”recommendation”, and ”system”. It might calculate the number of times each term or word occurs in a document and it is called term frequency. Term frequency $TF(t_1,d_1)$ is the simplest option to use the number of a term or word in a document, consider the number of times that term t1 occurs in document d1 and express the raw count by ft_1,d_1, then the simplistic TF scheme is $TF(t_1,d_1) =$

ft_1, d_1.

$$TF(t_1, d_1) = 0.5 + 0.5 * \frac{ft_1, d_1}{max f'_t, d_1 : t' \in d} \qquad 2.16$$

where boolean frequencies $TF(t_1, d_1)$ = 1 if t_1 occurs in d_1 and 0. Otherwise, term frequency adjusted for document length ft_1,d_1 (number of words in d_1) and logarithmically scaled frequency $TF(t_1, d_1)$ = $\log(1 + ft_1, d_1)$.

Because the word ”the” is so frequent, term frequency will calculate incorrectly to the document where use the word ”the”, without giving enough value to the important terms ”recommendation” and ”system”. Hence an inverse document frequency is included decreases the score of terms or words that use very frequently in the document set and increases the weight of terms or words that occur infrequently.

$$IDF(t_1, D_1) = \log * \frac{N}{|d \in D : t \in d_1|} \qquad 2.17$$

where total number of documents in the corpus $N = |D|$, $|\{d_1 \in D : t \in d_1\}|$ where the term t appears.

Then the $TF - IDF$ is calculation is,

$$\mathrm{TF} - \mathrm{IDF}(t_1, d_1, D) = \mathrm{TF}(t_1, d_1) \cdot \mathrm{IDF}(t_1, D) \qquad 2.18$$

High score in TF-IDF is shifted by a high term frequency with a low document frequency of the term in the full collection of documents. The score manages to separate common terms. Since the proportion inside the IDF 0, the log function is always greater than or equal to 1, the value of IDF is higher than or equal to 0. If a term appears in more documents then the ratio inside the logarithm approaches 1 and bringing the IDF and TF-IDF closer to 0.

2.4.4.2 KP-Miner

KP-Miner (El-Beltagy & Rafea, 2009b) is an unsupervised automatic keyphrase extraction algorithm that can extract keyphrases from English and Arabic documents. The principal goal was to develop a keyphrase extraction method depends on the natural language processing or linguistic tools, that do not require any training documents and can be easily configured by users based on their knowledge of the documents. The primary objective is to build this method is the lack of training data for any domain. KP-Miner applies a set of rules for extract candidate keyphrases. A phrase will unusually contain stop words within it and will never be divided by punctuation marks within text. The first requires a sequence of words or phrase has to present in order to be recognized as a candidate keyphrase and it is not be divided by any punctuation marks or stop words. A total of 187 common stop-words such as the, then, in, above, etc are applied to extract candidate keyphrase. Applying these rules, many candidate keyphrases will be generated for future processing but some of them do not create any meanings to the human readers. For cleaning this, two additional conditions are employed. The first condition is, a phrase has to have performed at least *n* times in a document from which keyphrases are to be extracted. This is called the frequency factor. *n* is dependent on the length of the document and increment or decrement is possible to depend on the length of the document.

The second condition is the position where a candidate keyphrase first appears in a document. By research, it was discovered that phrases occurring for the first time after a given threshold in a long document. If a phrase appears for the first time after a number of words, it is filtered out and ignored for final keyphrases. For the KP-Miner model (El-Beltagy, 2006), this cutoff value is constant (850). An optimum value for this constant is considered as 400 (El-Beltagy & Rafea, 2009b). The keyphrase extraction step described above is carried out in two phases. In the first phase, words are considered until a punctuation mark or a stop word. They also consider all possible n-grams within the confronted sequence where n can vary from 1 to sequence length 1 are stemmed and saved in both original and stemmed forms. Initially, the new phrase or sub-phrases is considered a count of one and it will increase depending on the repetition. Porter stemmer (D. Miller & Friesen, 1986) , a very week streaming algorithm is used for this algorithm. In the second phase, it finds the longest possible sequence that satisfies the conditions mentioned above. After that, this is considered as a candidate keyphrase. The devised algorithm sets no limit on the length of keyphrases, but it was found that extracted keyphrases rarely exceed three terms.

key features achieved from documents by models such as TF-IDF (Salton & Buckley, 1988b) and represent documents from which they use clustering and classification tasks. These models performed very badly (Frank et al., 1999b) when applied to the keyphrase extraction task. In any document experiences of keyphrases are much less frequent than the occurrence of single terms within the same document. When the keyphrase extraction task applied, TF-IDF performs poorly because it does not consider the fact into consideration which appears in a bias towards single words as they occur in larger numbers. So, the preference of single terms in a document is a boosting factor. Consider an input document d from where keyphrases will be extracted, a boosting factor B_d is calculated as follows:

$$B_d = \frac{|N_d|}{|P_d| * \beta} \tag{2.19}$$

$$if B_d > \sigma, B_d = \sigma \tag{2.20}$$

Here $|N_d|$ is the total number of candidate keyphrases in document d, $|P_d|$ is the total number of candidate keyphrases whose length passes one in document d and β and σ are the weight adjustment constants. The values selected for the implemented system are 3 for σ and 2.3 for β. The TF-IDF model in combination with the introduced boosting factor is used to calculate the score of document terms. For applying TF-IDF for a common application rather than a corpus-specific one is that keyphrase sequences do not occur as frequently within a document set. There are two potential methods to address this observation. A large corpus or dataset of various documents can be used to extract keyphrase with frequency information. Any encountered keyphrase is estimated to have developed only once in the corpus or dataset. For the compound keyphrases, frequency within a document as well as the boosting factor are defining its weight as the IDF value for all compound keyphrases will be a permanent c defined by the size of the corpus or dataset used to build frequency knowledge for single terms. The position factor is also used in the calculation for the term weights (El-Beltagy & Rafea, 2009b). In summary, the Next equation is used to compute the score of candidate keyphrases whether single or compound:

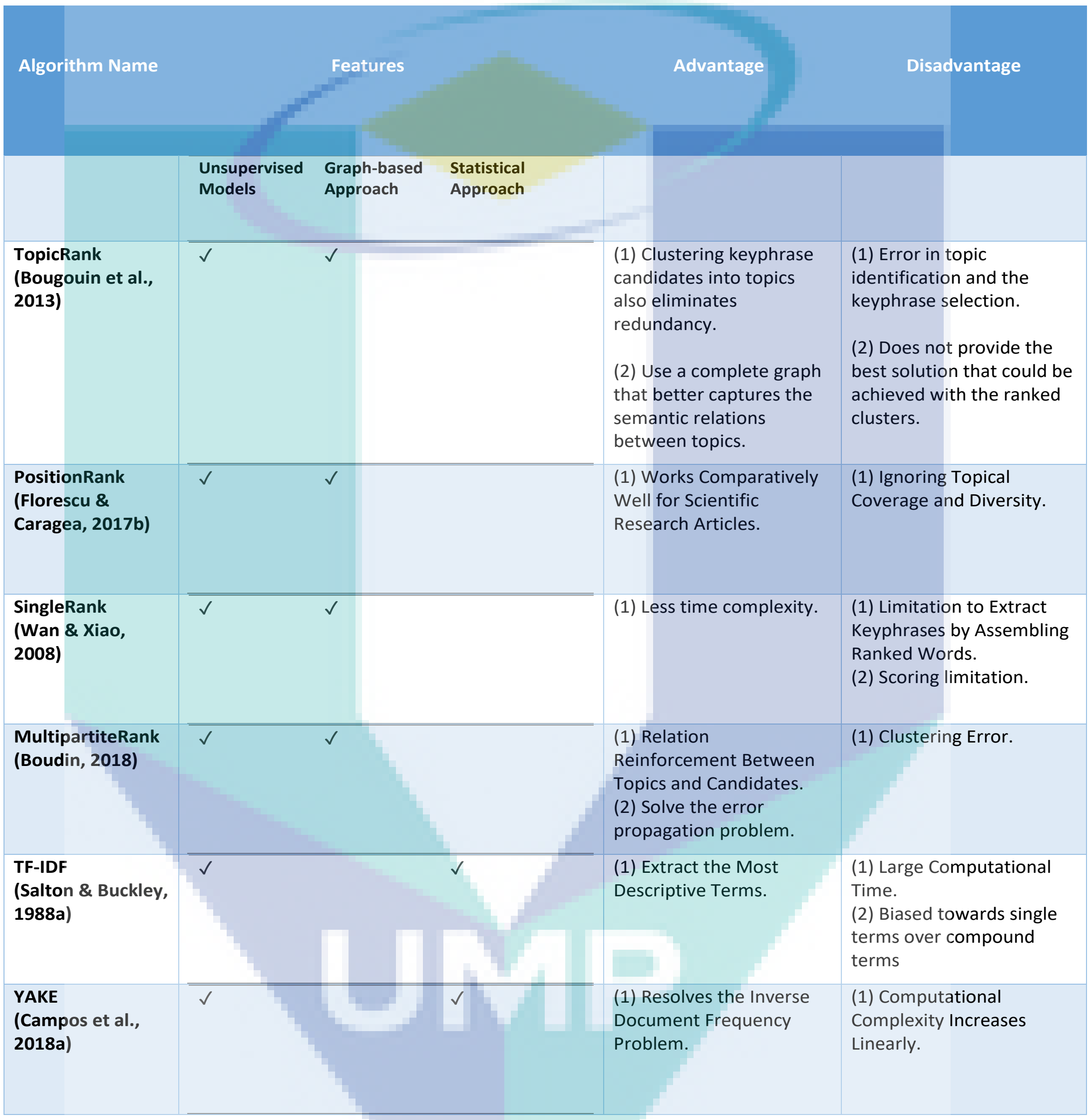
$$W_i j = t f_i j * idf * B_i * P_f \tag{2.21}$$

where $W_i j$ is the score of term t_j in Document D_i, $t f_i j$ frequency of term t_j in Document D_i, idf is the inverse document frequency, B_i is the boosting factor associated with document D_i, P_f is the term position associated factor.

In the KP-Miner algorithm, the user can define a number *n* of keyphrases and return the sorted list top *n* keyphrases defined by the user. The default value of *n* is five. When forming candidate keyphrases, the highest possible sequence of words that are constant by possible keyphrase terminators, are inquired and stored and so are sub-phrases included within that sequence provided that they appear somewhere in the text on their own. To improve outcomes in the KP-Miner algorithm, the top *n* keyphrases are considered to find if any of them is a sub- phrase of another. The count is decremented for the term if any of them are found in the candidate keyphrases. Finally, the weights are recalculated and a final list of keyphrases sorted by weight is produced. The top n key phrases rather than all candidate keyphrases are used in this step is so that lower scored keyphrases do not affect the outcome of the final keyphrase list. The cleaning steps are very important for the algorithm, but their research has shown that in the English version of the method, eliminating this step leads to the production of keyphrase documents that match better with author specified keyphrase.

2.4.4.3 YAKE

YAKE is a unique feature-based algorithm for multi-lingual keyphrase extraction from single document with various sizes, domains or languages. Unlike most of the keyphrase extraction algorithm, YAKE does not rely on references or training data. They follow an unsupervised machine learning approach which creates upon features extracted from the text, making it appropriate to documents written in several languages without any external knowledge. This algorithm is beneficial for a large number of tasks and lots of situations where the training data are limited or restricted. The proposed algorithm has six main parts: (1) Text pre-processing (2) Feature extraction (3) Individual terms score (4) Candidate keyphrases list generation (5) Data Deduplication and (6) Ranking. First, In the pre-processing step, the text splits into unique terms and the empty space or a special character such as brackets, line breaks, period, comma delimiter is found. Second, using a set of five features to capture the property of each individual term. These are (1) Casing (2) Word Positional (3) Word Frequency (4) Word Relatedness to Context and (5) Word Dif-Sentence. Casing shows the character of a word. Word index values provide information about those words occurring at the beginning of a document based on the cockiness that related keyphrases often tend to accumulate more at the beginning of a document. Word Frequency indicates the frequency of the word and also use for scoring those words. The fourth feature, Word Relatedness to Context, calculates the number of different terms or words that occur to the left side of the candidate keyphrases. Various terms or words that co-occur with the candidate word or phrases and more meaningless the candidate word or phrases are likely to be. Finally, Word DifSentence quantifies how frequently candidate keyphrases appear within different sentences. Similar to Word Frequency, Word DifSentence values means those words that often occur in different sentences. Both features, however, are connected with Word Relatedness to Context, meaning that the more they transpire in different sentences the better, as long as they do not transpire regularly with different words or terms on the right or left side. They heuristically link all these features into a single measure such that each term is allocated a score. This score will feed the process of making keyphrases which are to be taken in the fourth step. Here, they analyze a sliding window of 3-grams, thus creating a contiguous pattern of 1, 2 and 3-gram of candidate keyphrases. Each candidate keyphrase will then be given a final score, such that the smaller the score the higher meaningful the keyphrase will be. After calculating the weight of a candidate keyphrases, adding the score with the first term of the candidate keyphrases by the following scores of the remaining terms. This is divided by the total of the calculated scores to average out with respect to the length of the keyphrase. The result is additionally divided by the term frequency of the keyphrases - to penalize less frequent candidate keyphrases. In the final step, they remove similar candidate keyphrases coming from the previous steps. For this, they apply the Levenshtein distance (Yujian & Bo, 2007). Finally, the system will provide a list of relevant keyphrases.



Algorithm Name	Features			Advantage	Disadvantage
	Unsupervised Models	Graph-based Approach	Statistical Approach		
TopicRank (Bougouin et al., 2013)	✓	✓		(1) Clustering keyphrase candidates into topics also eliminates redundancy. (2) Use a complete graph that better captures the semantic relations between topics.	(1) Error in topic identification and the keyphrase selection. (2) Does not provide the best solution that could be achieved with the ranked clusters.
PositionRank (Florescu & Caragea, 2017b)	✓	✓		(1) Works Comparatively Well for Scientific Research Articles.	(1) Ignoring Topical Coverage and Diversity.
SingleRank (Wan & Xiao, 2008)	✓	✓		(1) Less time complexity.	(1) Limitation to Extract Keyphrases by Assembling Ranked Words. (2) Scoring limitation.
MultipartiteRank (Boudin, 2018)	✓	✓		(1) Relation Reinforcement Between Topics and Candidates. (2) Solve the error propagation problem.	(1) Clustering Error.
TF-IDF (Salton & Buckley, 1988a)	✓		✓	(1) Extract the Most Descriptive Terms.	(1) Large Computational Time. (2) Biased towards single terms over compound terms
YAKE (Campos et al., 2018a)	✓		✓	(1) Resolves the Inverse Document Frequency Problem.	(1) Computational Complexity Increases Linearly.

Figure 2.8. Analysis for automatic unsupervised keyphrase extraction

2.5 Critical Analysis

Starting with the relevant graph-based unsupervised keyphrase extraction techniques, the basic idea is to build a graph from an input document and to rank its nodes according to their importance (Brin & Page, 1998). Such a technique is KeyGraph (Ohsawa et al., 1998), which is content sensitive and domain-independent technique that utilizes co-occurrence of various terms for indexing vertices of the graph. However, it fails to detect the relationships among the low-frequency items inside clusters and also ignores direct relationships between the clusters (H. Wang, Xu, Hu, & Ohsawa, 2013).

On the other hand, PageRank (Page et al., 1999) is based on the concept of random walks and is related to eigenvector centrality that tends to favor nodes with many important connections regardless of cohesiveness considerations. Each node of the graph corresponds to a candidate keyphrase from a document and an edge connects two related nodes; and thus, weight of an edge represents the syntactic relevance between the connected candidate keyphrases. This technique is well suited for raking pages on the web and social networks, but not suitable for keyphrase extraction due to lack of consideration of cohesiveness (Mihalcea & Tarau, 2004; J. Wang, Liu, & Wang, 2007).

An extension of PageRank, called PositionRank (Florescu & Caragea, 2017b), incorporates all the positions of a word along with its frequency to score the word; and thus, decides the rank of that word. This way, it outperforms all the techniques that consider only the first position information in the ranking. However, due to ignoring topical coverage and diversity which is not naturally handled by this kind of graphs (Hasan & Ng, 2014), this technique suffers from considerably limited performance.

TextRank (Mihalcea & Tarau, 2004) is one of the most well-known graph-based approaches for keyphrase extraction. Here, the scientific documents are modeled as undirected or directed and weighted co-occurrence networks using a co-occurrence window of variable sizes (Mihalcea & Tarau, 2004). It experiences several limitations, such as its incapability to capture cohesiveness. Again, retaining only the main core is suboptimal since sometimes it is impractical to discover all the gold standard keyphrases within a unique subgraph; whereas, many valuable keyphases may place in the lower levels of the hierarchy (Tixier, Malliaros, & Vazirgiannis, 2016). Moreover, due to selecting or discarding a large group of words at a time reduces the flexibility of the extraction process, and negatively impacts the performance.

An extension of TextRank, named SingleRank (Wan & Xiao, 2008), which weights an edge equal to the number of times the two corresponding words co-occur. Unlike its predecessor, it does not extract keyphrases by assembling ranked words, instead, only noun phrases are extracted from a document. However, sometimes it assigns higher scores to long but non-significant keyphrases which entices the ranking procedure.

Again, in enhancement of TextRank, called TopicRank (Bougouin et al., 2013) the vertices of a graph are topics, not words. It extracts the noun phrases that represent the main topics of a document and clustered them into topics. A notable advantage of this technique is that it considers topical coverage and diversity. However, it equally weighs all candidates belonging to a single topic. In addition, it suffers from the error propagation problem which may occur during topics formation.

To resolve the error propagation problem of TopicRank, the MultipartiteRank technique (Boudin, 2018) utilizes multipartite graph. Here, a complete directed multipartite graph is built that are connected only if they belong to different topics. Since this technique makes good use of relation reinforcement between topics and candidates, it performs better than other graph-based techniques. However, due to clustering error (where candidate keyphrases could be wrongly assigned to a similar topic), it struggles in selecting the most representative candidates.

Although graph-based techniques show acceptable performance in many occasions, they are considerably difficult to implement in comparison to statistical unsupervised keyphrase extraction techniques.

The most prominent and state-of-the-art statistical technique is Term Frequency - Inverse Document Frequency (TF-IDF) (Salton & Buckley, 1988a), which reflects the importance of a keyphrase to a document in a corpus. Among the two terms, TF provides aboutness and IDF provides informativeness. The IDF discriminates between informative and non-informative keyphrases across the documents; whereas, the TF discriminates between popular and non-popular keyphrases in a document. This technique is computationally expensive as IDF is calculated across different documents. Again, many studies report that this technique is biased towards single terms over compound terms (El-Beltagy & Rafea, 2010).

To resolve the problem of favoring single terms, KP-Miner (El-Beltagy & Rafea, 2010) is proposed. It utilizes some heuristics based on TF and positions to identify potential keyphrases which are weighted with TF-IDF score (Jean-Louis, Zouaq, Gagnon, & Ensan, 2014). Although it outperforms TF-IDF, it experiences several limitations such as drop in global ranking performance with increasing length or number of documents (Merrouni, Frikh, & Ouhbi, 2016). In addition, it is computationally expensive due to its dependence on TF-IDF.

Another lightweight technique is YAKE (Campos et al., 2018a) which resolves the IDF problem. It takes five features into consideration, namely casing, word position, word frequency, word relatedness to context, and word in the different sentence to calculate the weight of a keyphrase. Again, due to generating candidate keyphrases employing N-grams technique, its computational complexity increases linearly with respect to N-grams (C. Xu et al., 2017). Again, due to the same reason, a large number of keyphrases are generated, which entices the ranking procedure.

From the above discussions, it is evident that graph-based techniques and statistical techniques have several adverse characteristics, which restrict them in achieving better performance. To overcome the identified shortcomings, in this thesis proposes a tree-based technique to extract quality keyphrases from documents.

2.6 Summary

Figure 2.8 summaries the most common features that have been used in keyphrase extraction systems. Observable features are covered, such as Punctuation, Stop-words, Part-Of-Speech Tags (POS Tags) etc. A feature's type is taken as a basis for categorization. This chapter discusses the background knowledge that is essential to understand this thesis. Initially, we propose a taxonomy where we divide all the existing keyphrase extraction technique for academic literature. Afterward, they are further divided into several subclasses. Since all the subclasses are not in-line, therefore, our further discussions are limited to those classes that are similar to the proposed algorithm. Again, since every class has numerous of an algorithm, so, only prominent algorithm are described, scrutinized, and analyses with their advantages and limitations—which inspires to propose a new algorithm. Furthermore, evaluation metrics for the various algorithm are also mentioned with various comparison tables. In the end, the most related algorithms are deeply investigated and their efficiency and drawbacks are highlighted.

CHAPTER 3

METHODOLOGY

3.1 Preamble

This section states the assumption of the proposed technique followed by formulating the problems of keyphrase extractions. Afterwards, the conceptual frameworks that are taken into account while developing the proposed technique to resolve the aforementioned problems are explained.

Generally, most of the unsupervised keyphrase extraction techniques immediately start ranking after extracting candidate keyphrases. A large number of candidate keyphrases contend for receiving good ranks, which many times entice ranking procedures. It could be compared with a human analogy: too many options make it harder to choose (Scheve, 2018). Conversely, a refinement of candidate keyphrases and generation of final keyphrases from them would reduce the keyphrase quantity to a considerable amount, thus, facilitate the ranking procedure.

3.2 Problem Formulation

Considering a document, δ, which is passed to a keyphrase extraction technique to extract final keyphrases, φ. For this, at first it is necessary to extract candidate keyphrases, χ from δ, which will be processed later to extract φ. There are several techniques to extract χ from δ and the one employed in the proposed technique can be found in Section 3.4.1.

For now, χ has been extracted from δ. Any candidate keyphrase χ_i in χ (i.e., $\chi_i \in \chi$)

is comprised of n number of ordered sequence of words, $\{w_1, w_2, ..., w_m, ..., w_{n-1}, w_n\}$, where $n \in \mathbb{Z}+$. As mentioned earlier, any keyphrase is a coherently connected sequence of words that appear contiguously. Therefore, χ_i could be represented as an ordered set and its segments also could be represented as ordered subsets. Again, when $n = 1$, χ_i contains only one word; otherwise, multiple words. Note that χ_i cannot be empty; and therefore, $|\chi_i| = n \neq 0$.

For extracting a final keyphrase, φ_j (where $\varphi_j \in \varphi$) from a χ_i, the latter is necessary to be processed. For this, following probable cases need to be considered:

Case 1 : χ_i is φ_j, i.e., $\chi_i = \varphi_j$ or $\chi_i \subseteq \varphi_j$ and $\varphi_j \subseteq \chi_i$.

Case 2 : φ_j is a part of χ_i, i.e., $\varphi_j \subset \chi_i$.

Case 3 : Again, χ_i is a part of φ_j, i.e., $\chi_i \subset \varphi_j$.

Case 4 : χ_i is not a final keyphrase.

Although, four probable cases are identified, but it is difficult to determine under which case a certain candidate keyphrase falls. To identify that, in the subsequent section, we discuss some hypotheses and observations.

3.3 Conceptual Framework

The concept of extracting final keyphrases from candidate keyphrases relies on the following hypotheses and observations:

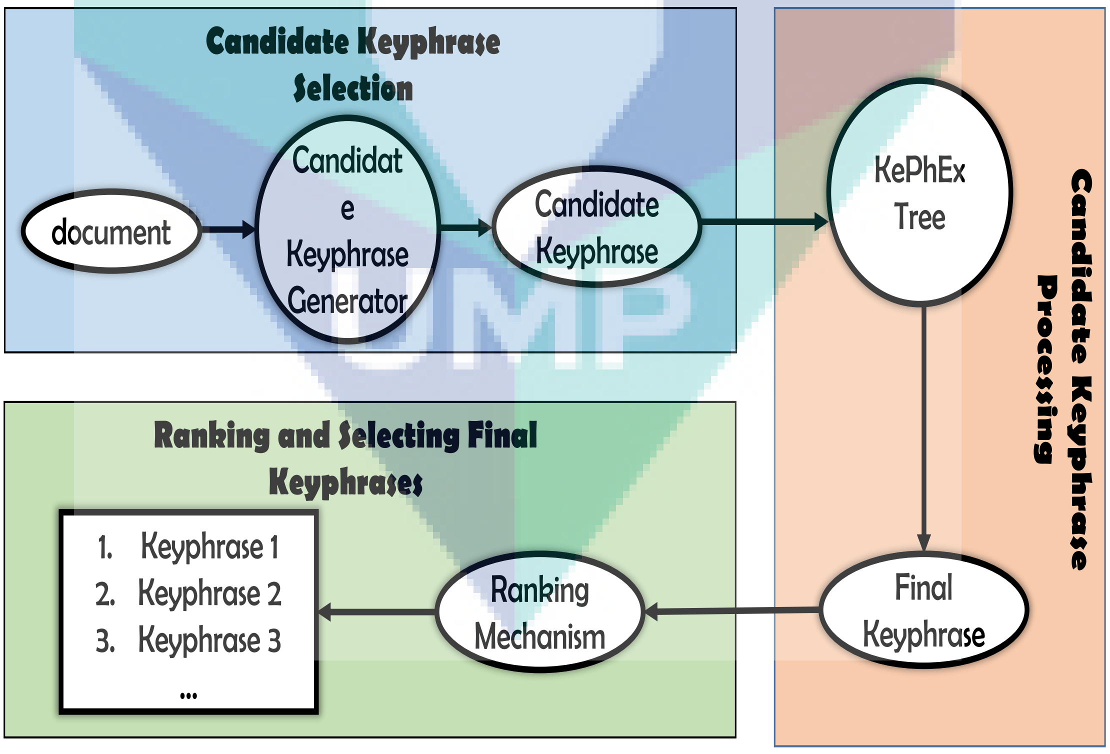
Hypothesis 1 : For any χ_i, *case 1* and *case 4* can be determined by its popularity. In other words, this decision can be taken based on the frequency of χ_i in a document and applying a binary decision strategy.

Hypothesis 2 : For *case 2*, since a part of χ_i — denoted as χ_i' — is a final keyphrase (i.e., $\chi_i' = \varphi_j$), the popularity and the cohesiveness of χ_i' must be higher than that of χ_i. In this case, χ_i need to be appropriately reduced to χ_i'.

Hypothesis 3 : For *case 3*, since χ_i is a part of φ_j, χ_i need to be expanded to χ_i' such that $\chi_i' = \varphi_j$. Again in this case, the popularity and the cohesiveness of χ_i' must be higher than that of χ_i.

Hypothesis 1 is quite straightforward. A simple binary decision strategy could be applied to determine this. For instance, assuming that the frequency of χ_i is ρ in δ. Now, it is compared with λ, which is a constant value, and also known as least seen allowable frequency (lsaf) factor. The value of λ varies from one language to another (El-Beltagy & Rafea, 2009a), and also subject to the length of a document (El-Beltagy & Rafea, 2009a). When $\rho < \lambda$, it is not a final keyphrase, otherwise, it is likely to be a final keyphrase.

Again, for *hypothesis 2* and *hypothesis 3*, the proposed rooted binary tree expands or shrinks based on the candidate keyphrases and keeps track of the cohesiveness of various words in a keyphrase with respect to the root. At the end, the final keyphrases are extracted from the tree as detailed in the subsequent section.



Candidate Keyphrase Selection
document
Candidat e Keyphrase Generator
Candidate Keyphrase
KePhEx Tree
Candidate Keyphrase Processing
Final Keyphrase
Ranking Mechanism
Ranking and Selecting Final Keyphrases
1. Keyphrase 1
2. Keyphrase 2
3. Keyphrase 3
...

Figure 3.1. Functional details of the proposed technique

3.4 Proposed Technique: Tree-based Keyphrase Extraction Technique (TeKET)

The entire process of keyphrase extraction using our proposed technique can be parted into three main phases as exhibited in figure 3.1 : *i*) candidate keyphrase selection or pre-processing, *ii*) candidate keyphrase processing or simply processing, and *iii*) ranking and selecting final keyphrases or post-processing. Briefly, the pre-processing technique selects candidate keyphrases from a document which are processed using the newly proposed technique, and the final keyphrases are ranked based on the new ranking approach (see section 3.4.3) that employs a combination of a value obtained from the tree and the term-frequency as factor. At the end, top-N keyphrases are selected as final keyphrases.

3.4.1 Candidate Keyphrase Selection

Generally, for selecting candidate keyphrases, most of the existing techniques either employ N-gram approach (Kim, Baldwin, & Kan, 2010; Kim, Medelyan, Kan, & Baldwin, 2010; Kumar & Srinathan, 2008) or *Part-Of-Speech (POS) Tagging (POST)* approach (Barker & Cornacchia, 2000; Hulth, 2003a; Ono, Hishigaki, Tanigami, & Takagi, 2001; Yu & Ng, 2018). However, the former one extracts a huge number of candidate keyphrases, which are afterwards reduced employing several conditions, such as removing *stop words* (Bird, Klein, & Loper, 2009; Fox, 1989), removing *punctuation marks* (Bayraktar, Say, & Akman, 1998; R. Wang, Liu, & Mcdonald, 2014), removing keyphrases that fail to satisfy *lsaf* factor, and so on (Hasan & Ng, 2014).

For instance, let us consider that a document has $N = 1000$ words, and 3-gram procedure is applied for selecting candidate keyphrases. In this case, if no condition is applied to clean the keyphrase list, then $N + (N-1) + (N-2)$ candidate keyphrases will be generated; where N for 1-gram, $N-1$ for 2-gram, and so on. Hence, for 1000 words, around 2997 keyphrases will be generated; where many of them are nonsensical keyphrases. On the other hand, due to considering these nonsensical keyphrases, feature space will increase; and hence, the processing complexity will also increase.

For extracting candidate keyphrases, we employ the *Part-Of-Speech (POS) Tagging (POST)* on a given document, *D*; and afterwards, we perform *POS* pattern matching to limit ourselves only to noun phrases (Barker & Cornacchia, 2000; Frank et al., 1999a). In this place, we would like to define noun phrases since it is necessary for clear understanding of the further discussions. According to ("Noun Phrase", (accessed May 3, 2018)), a noun phrase is a phrase that is formed by a noun and functions in a sentence as subject, object, or prepositional object. Therefore, in a sentence, a noun phrase may appear in many different patterns. In (Popova & Danilova, 2014), authors found 56 such patterns (All patterns are not only noun phrase) from a training dataset. However, only some of these patterns could yield quality keyphrases from noun phrase. Therefore, in this thesis, only prominent patterns are chosen for an initial investigation with an intention of selecting only one such pattern for further processing. The patterns that are considered for the investigation are mentioned in Table 3.1.

Table 3.1. POS patterns

	Denoted as	POS Pattern
1	A	$(<NN.*>+)\vert(<NN.*>+<JJ.*>?)\vert(<JJ.*>?<NN.*>+)$
2	B	$<NN.*\vert JJ>*<NN.*>$
3	C	$(<JJ.>\vert<NN.>)*<IN>?(<JJ.>\vert<NN.>)*<NN.>$
4	D	$<PRP>?<JJ.*>*<NN.*>+$
5	E	$<DT\vert PP\$>?<JJ>*<NN.*>+$
6	F	$(<\backslash w+DT>)?(<\backslash w+JJ>)*(<\backslash w+>(<NN\vert NP\vert PRN>))$
7	G	$(<NN.*>+<JJ.*>?)\vert(<JJ.*>?<NN.*>+)$

There are 7 candidate *POS* patterns are asserted in the table, which are denoted with characters from *A* to *G*. Aforementioned expressions are regular expressions, which are written in a simplified format used by NLTK's RegexpParser (*Regular Expression HOWTO*, n.d.); where adjectives are tagged with *JJ*, nouns are tagged with *NN*, personal pronouns are tagged with *PRP*, determiners are tagged with *DT*, prepositional phrases are tagged with *PP*, and preposition/subordinating conjunctions are tagged with *IN*.

’self-adapt applic’, ’grid gosia wrzesinska jason maassen henri e. bal dept’, ’comput systems’, ’universiteit amsterdam’, ’gosia’, ’jason’, ’inher heterogen’, ’dynamic’, ’import problem’, ’grid comput’, ’resourc selection’, ’appropri resourc’, ’application’, ’problem’, ’chang characterist’, ’grid environment’, ’solut’, ’problem requir’, ’perform model’, ’construct such model’...

Figure 3.2. An example of the candidate keyphrases for POS patterns 1.

In POS patterns 1, where nouns are tagged with *NN* and adjectives are tagged with *JJ*. This expression has three parts and is separated by the operator ”|”.The first part selects those noun phrases that must have nouns of any format (i.e., *NN*, *NNS*, *NNP*, *NNPS*). The second part selects those noun phrases that must have nouns of any format (i.e., *NN*, *NNS*, *NNP*, *NNPS*) for at least one time or more followed by adjectives of any format (i.e., *JJ*, *JJR*, *JJS*) once or none and the third part selects those adjectives of any format (i.e., *JJ*, *JJR*, *JJS*) for at least once or none followed by noun phrases that must have nouns of any format (i.e., *NN*, *NNS*, *NNP*, *NNPS*) one time or more. Figure 3.2 shows the example of the POS patterns 1.

’self-adapt applic’, ’grid gosia wrzesinska jason maassen henri e. bal dept’, ’comput systems’, ’vrije universiteit amsterdam’, ’gosia’, ’jason’, ’inher heterogen’, ’dynamic’, ’import problem’, ’grid comput’, ’resourc selection’, ’appropri resourc’, ’application’, ’problem’, ’chang characterist’...

Figure 3.3. An example of the candidate keyphrases for POS patterns 2.

In POS pattern 2, where nouns are tagged with *NN* and adjectives are tagged with *JJ*. The POS pattern selects those noun phrases that have nouns of any format (i.e., *NN*, *NNS*, *NNP*, *NNPS*) or adjectives of any format (i.e., *JJ*, *JJR*, *JJS*) with must have nouns of any format (i.e., *NN*, *NNS*, *NNP*, *NNPS*). Figure 3.3 shows the example of the POS patterns 2.

In POS pattern 3, where nouns are tagged with *NN*, adjectives are tagged with *JJ* and prepositions are tagged with *IN*. The POS pattern selects those adjectives of any format (i.e., *JJ*, *JJR*, *JJS*) or noun phrases that have nouns of any format (i.e., *NN*, *NNS*, *NNP*,

’systems’, ’models’, ’of resources’, ’statistics’, ’needs’, ’bottlenecks’, ’processors’, ’therefor’, ’improvements’, ’networks’, ’techniques’, ’years’, ’problems’, ’parallel’, ’supercomputers’, ’node’, ’environments’. . .

Figure 3.4. An example of the candidate keyphrases for POS patterns 3.

NNPS) with must have a preposition. The preposition must have zero or one occurrence of any format (i.e., *JJ*, *JJR*, *JJS*) or noun phrases that have nouns of any format (i.e., *NN*, *NNS*, *NNP*, *NNPS*) with must have a noun phrase (i.e., *NN*, *NNS*, *NNP*, *NNPS*). Figure 3.4 shows the example of the POS patterns 3.

’self-adapt applic’, ’grid gosia wrzesinska jason maassen henri e. bal dept’, ’comput systems’, ’vrije universiteit amsterdam’, ’gosia’, ’jason’, ’inher heterogen’, ’dynamic’, ’import problem’, ’grid comput’, ’resourc selection’, ’appropri resourc’, ’application’, ’problem’, ’chang characterist’, ’grid environment’, . . .

Figure 3.5. An example of the candidate keyphrases for POS patterns 4.

In POS patterns 4, where nouns are tagged with *NN*, adjectives are tagged with *JJ* and Personal pronouns are tagged with *PRP*. The POS pattern selects those adjectives of any format (i.e., *JJ*, *JJR*, *JJS*) with must have a noun that has nouns of any format (i.e., *NN*, *NNS*, *NNP*, *NNPS*). Figure 3.5 shows the example of the POS patterns 4.

’self-adapt applic’, ’the grid gosia wrzesinska jason maassen henri e. bal dept’, ’comput systems’, ’vrije universiteit amsterdam’, ’gosia’, ’jason’, ’inher heterogen’, ’dynamic’, ’import problem’, ’grid comput’, ’resourc selection’, ’an appropri resourc’, ’the application’, ’anoth problem’, ’the chang characterist’, ’the grid environment’,. . .

Figure 3.6. An example of the candidate keyphrases for POS patterns 5.

In POS patterns 5, where nouns are tagged with *NN*, adjectives are tagged with *JJ*, Determiners are tagged with *DT*. The POS pattern selects those adjectives that have adjectives of any format (i.e., *JJ*, *JJR*, *JJS*) with nouns of any format (i.e., *NN*, *NNS*, *NNP*, *NNPS*) with have once or none determiner. Figure 3.6 shows the example of the POS patterns 5.

’self-adapt applic’, ’grid gosia wrzesinska jason maassen henri e. bal dept’, ’comput systems’, ’vrije universiteit amsterdam’, ’gosia’, ’power’, ’thu’, ’the possibl’, ’veri larg problems’, ’multipl site’, ’the same time’, ’the complex’, ’grid environ’, ’mani time’, ’tradit parallel’, ’cluster’,. . .

Figure 3.7. An example of the candidate keyphrases for POS patterns 6.

In POS patterns 6, where nouns are tagged with *NN*, adjectives are tagged with *JJ*, determiners are tagged with *DT* and Possessive pronoun are tagged with PRN. The POS pattern selects determiners once or none with those adjectives that have adjectives of any format (i.e., *JJ*, *JJR*, *JJS*) with nouns that must have nouns or possessive pronoun of any format (i.e., *NN*, *NNS*, *NNP*, *NNPS*). Figure 3.7 shows the example of the POS patterns 6.

’self-adapt applic’, ’grid gosia wrzesinska jason maassen henri e. bal dept’, ’comput systems’, ’universiteit amsterdam’, ’gosia’, ’jason’, ’inher heterogen’, ’dynamic’, ’important problem’, ’grid comput’, ’resourc selection’, ’appropri resourc’, ’application’, ’problem’, ’chang characterist’, ’grid environment’, ’solut’, ’problem requir’, ’perform model’,. . .

Figure 3.8. An example of the candidate keyphrases for POS patterns 7.

In POS patterns 7, this expression has two parts and are separated by the operator “|”. The first part selects those noun phrases that must have nouns of any format (i.e., *NN*, *NNS*, *NNP*, *NNPS*) for at least one time or more followed by adjectives of any format (i.e., *JJ*, *JJR*, *JJS*) once or none; whereas, the second part selects those keyphrases that have adjectives of any format once or none followed by nouns of any format for at least one time. Figure 3.8 shows the example of the POS patterns 7.

Therefore, the proposed technique employs *POST* approach to extract those keyphrases that make sense. Since keyphrases generally are noun phrases (Turney, 2000), the proposed technique limits the extraction to *noun phrases* (Barker & Cornacchia, 2000; Frank et al., 1999a; Goyvaerts & Levithan, 2012; Loper & Bird, 2002). Consequently, in our proposed technique, following *POS* pattern is employed as it has been demonstrated in (Rabby, Azad, Mahmud, & Zamli, 2018) is that it can extract more suitable candidate keyphrases than other similar prominent patterns.

$$(<NN.*>+<JJ.*>?)|(<JJ.*>?<NN.*>+)$$

After extracting these noun phrases, they are cleaned before passing them to the processing phase. In the cleaning process, candidate keyphrases that are less likely to be final keyphrases are filtered out from the list. For that, following conditions are applied: i) any candidate keyphrase that contains non-alphabetic characters, ii) any candidate keyphrase that contains single alphabetic word(s), and iii) if the frequency of any candidate keyphrase fails to satisfy *lasf* factor. The former two conditions are applied to filter out those candidate keyphrases that make no sense to the human reader in general; and hence, less likely to become final keyphrases. On the other hand, the latter one filter out all non-popular candidate keyphrases from the list. The justification of this assumption is that generally final keyphrases are not rare in a document. Here, note that *lsaf* is decremental with respect to the length of a document (Nguyen & Kan, 2007), and varies from one language to another. For academic literature in English language, *lsaf* value is suggested to be three by several researchers for better cleaning performance (Nguyen & Kan, 2007; Witten et al., 2005; Zhao et al., 2011).

3.4.2 Candidate Keyphrase Processing

In conventional unsupervised keyphrase extraction techniques, candidate keyprhases are not processed; instead, they are sent to the ranking phase immediately after selecting. On the contrary, an intermediate phase between candidate keyphase selection and ranking — which would extract final keyphrases — could release the burden of ranking unnecessary keyphrases, and thus, assist finding more appropriate keyphrases. The proposed *KePhEx* tree takes all the hypotheses (see Section 3.3) into account for extracting final keyphrases. The *KePhEx* tree expands (*hypothesis 3*) or shrinks (*hypothesis 2*) or remains in the same state (*hypothesis 1*) based on the candidate keyphrases. The advantages of employing *KePhEx* tree in keyphrase extraction are threefold: i) extracts quality scalabl grid servic discoveri base, grid servic, servic discoveri mechan, scalabl web servic permiss, distribut grid servic discoveri architectur, servic discoveri architectur, grid discoveri servic, servic discoveri, grid inform servic, servic discoveri grid comput, servic technolog, servic discoveri function, grid servic call registri, web servic version, discoveri servic, servic properti, thi servic, index servic, servic discoveri, web servic commun, . . .

Figure 3.9. An example of similar candidate keyphrases consider *servic* as a root.

keyphases from candidate keyphrases, *ii*) provides flexibility during keyphrase extraction, and *iii*) contributes in ranking by providing a value that represents cohesiveness of a word in a keyphrase with respect to a root.

Although there exist a number of different binary trees, the *KePhEx* tree is different from them in that, the position of a node in terms of the subtree and its level is fixed. Again, all the predecessors of a node at the upper-levels (including root) are also fixed. It is so because a good keyphrase must be a coherently connected sequence of words that appear contiguously in the text (see Section 1.2). Thus, the *KePhEx* tree preserves the characteristics of a good keyphrase intact. For instance, in a candidate keyphrase, *cognitive agent specification language*, *specification* is the root. In the *KePhEx* tree, all the words at the left side of *specification* will become the nodes of the left subtree and all the words at the right side will become the nodes of the right subtree. Again, since *cognitive* is at the left side of *specification*, it must be in the left subtree, and since it is two words away from the root, its level must be two. Moreover, its predecessors must be *agent* and *specification*, respectively.

Every node in a *KePhEx* tree holds a 2-tuple data along with other information, namely a word and its μ value. The latter one provides twofold advantages: *i*) it assists in finding the cohesiveness of various words with respect to the root of a tree, which is employed as a factor in ranking keyphrases, and *ii*) it provides flexibility during keyphrase extraction. Again, note that the value of μ increases or decreases based on the existence of that word in candidate keyphrases.

Root Selection

Before forming a *KePhEx* tree, it is necessary to select the root of that tree. This is important since a poorly selected root may lead to a poor keyphrase. Therefore, in our proposed technique, only nouns are designated as roots. The justification of the choice is that noun phrases are the most likely candidate for final keyphrases; and hence, nouns as roots would increase the chances of extracting quality final keyphrases. In the proposed technique, nouns are selected from the candidate keyphase list, χ and saved them in a list, η.

After selecting the roots, the trees are formed taking these roots into consideration. The entire process from tree formation to final keyphrase extraction could be segmented into three main steps, namely *i*) tree formation, *ii*) tree processing, and *iii*) keyphrase extraction. They are discussed below with necessary examples.

Tree Formation

For forming a *KePhEx* tree, a root, γ is selected from η. Afterwards, the proposed system selects candidate keyphrases that contains the root. Let us denote them as similar candidate keyphrases, which could be defined as follows:

Definition 1. Similar candidate keyphrases, σ are those candidate keyphrases that contain γ in them — irrespective of its position, and $\sigma \subseteq \chi$.

A partial sample of σ for $\gamma =$ *servic* is shown in Fig. 3.9. Among them, the first encountered similar candidate keyphrase, σ_1 (e.g., *scalabl grid servic discoveri base* of Fig. 3.9) is employed to form the *KePhEx* tree and the rest are utilized for tree processing (see Section 3.4.2).

Here, the process of tree formation starts by selecting the position of γ in σ_1; but the tree starts forming once the γ is assigned as the root of the tree and μ value is initialized to 1. Now, all the words at the left side of γ are included in the left subtree and all the words at the right side of γ are included in the right subtree. The algorithm for adding a node in

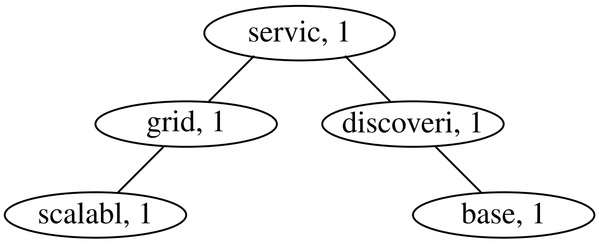


Figure 3.10. A newly created tree using a candidate keyphrase, where $\gamma = servic$.

the *KePhEx* tree is illustrated in Algorithm 1.

As the algorithm describes, when γ is passed to the *AddNode* function, if at that time, the root is found *NULL*, it is designated as the root and μ value is initialized as 1. If it is not the γ, in that case, position of w_i, i.e., w_p is determined to decide in which subtree it would be placed. If position of γ, i.e., γ_p is more than w_p (i.e., $\gamma_p > w_p$), it would be placed in the left subtree; otherwise, the right subtree.

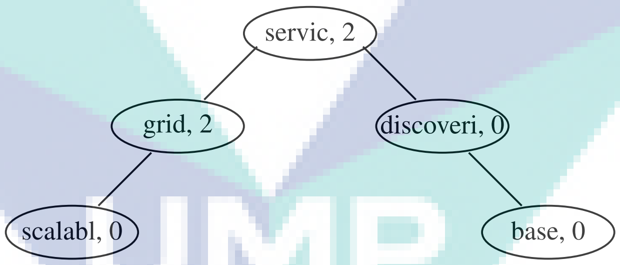
Again, the depth of a word, w_d in a phrase with respect to γ is also necessary to calculate to determine the level of the tree where w would be added, which could be defined as follows:

Definition 2. Depth of a word, w_d in a keyphrase is the distance of that word from γ irrespective of its direction, which is calculated as, $w_d = |\gamma_p - w_p|$.

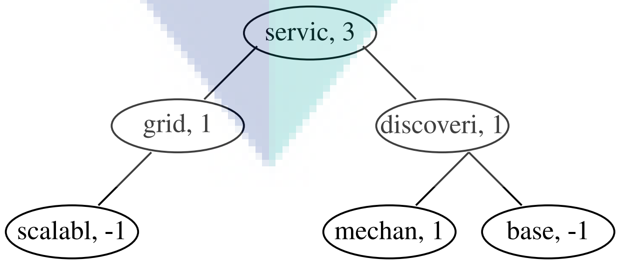
Note that w_d in a candidate keyphrase and the level of that word in the *KePhEx* tree, l is identical; and hence, they are used interchangeably in this thesis. Now, to determine where the new word would be added in the tree, both w_p and w_d is necessary. Here, we would like to mention that w_p and w_d are jointly unique with respect to any σ_i, where $\sigma_i \in \sigma$. Once they are determined, the next condition to be satisfied is that all the predecessors must be in their respective places. This can be tracked by adding all the nodes in the tree sequentially from higher level to lower level and by ensuring that all the words between the depth 0 to $d-1$ are placed in their respective levels in the tree. Once these constraints are satisfied, w is qualified for adding in the tree at the level l. For adding w in the tree, a node is created incorporating it and initializing μ to 1. Afterwards, it is added in the tree in its respective place.

Once all the words at the left side of γ is added in the left subtree; then the words at the right side of γ are added in the right subtree following the same procedure. The tree formation ends when all the words of σ_1 are added in the tree. A sample tree is depicted in Fig. 3.10.

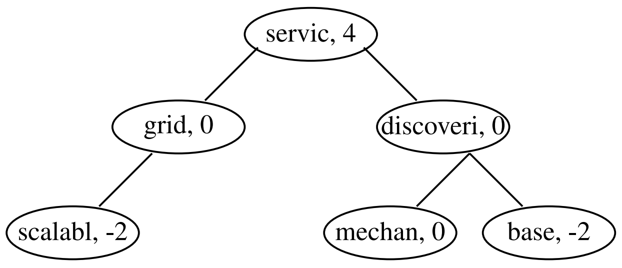
Considering σ_1 mentioned in Fig. 3.9, i.e., *scalabl grid servic discoveri base*, *servic* is the root. Now, the tree formation starts by adding *servic* in the tree and initializing μ of the node to 1. Afterwards, all the words at the left side (i.e., *grid* and *scalabl*) are added in the left subtree in their respective levels, where levels are calculated based on their respective depths in σ_1. For instance, since $grid_d = 1$, *grid* is added at level 1 in left subtree, whereas, since $scalabl_d = 2$, *scalabl* is added at level 2 in left subtree. Again, when *grid* is added in the tree, it is tracked that its predecessor is *servic* in the tree. Similarly, when *scalabl* is added in the tree, it is tracked that *grid* and *servic* are its predecessors, respectively. Once all the words at the left side of *servic* are added in the tree, the words at the right side (i.e., *discoveri* and *base*) are added in the right subtree employing a similar procedure as the left subtree.



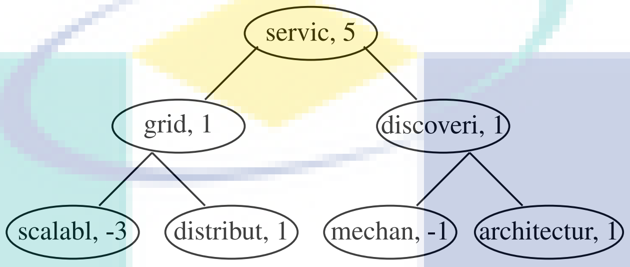
(a) σ'_1 = grid servic



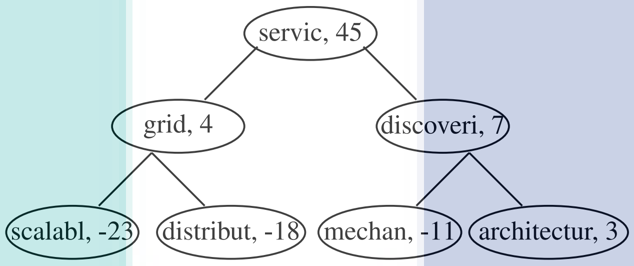
(b) σ'_2 = servic discoveri mechan



(c) σ_3' = scalabl web servic permiss



(d) σ_4' = distribut grid servic discoveri architectur



(e) σ_n' = grid servic discoveri

Figure 3.11. Several tree processing steps, where $\gamma = servic$.

Tree Processing

After forming the tree employing σ_1, the rest of the similar candidate keyphrases, σ', where σ' = $\{\sigma_2, \sigma_3, ..., \sigma_n\}$ are utilized to process the tree to extract final keyphrases. Following the cases in processing the newly formed *KePhEx* tree (see Section 3.1), for *case 1* no processing is needed; for *case 2* the tree must be trimmed properly to remove unnecessary parts; and for *case 3* it must be expanded to put on necessary parts from all the similar candidate keyphrases in σ'. This process has been described in Algorithm 1.

Let us fetch a similar candidate keyphrase, σ_i' from σ', and utilize it for processing the *KePhEx* tree. At first, γ_p in σ_i' need to be determined. Afterwards, all the words are

Algorithm 1 Adding a Node in the KePhExTree

Input: word, wordIndex, root, rootIndex, termFreq, phrase
Output: return **TRUE** if a node is added; otherwise **FALSE**

```
 1: if word is None then
 2:   return
 3: if root is None then
 4:   create a node and assign it as root
 5:   return TRUE
 6: depth $\leftarrow$ abs(wordIndex - rootIndex)
 7: count $\leftarrow$ 0
 8: newNode $\leftarrow$ root
 9: if wordIndex $<$ rootIndex then
10:   while count $<$ depth - 1 or newNode is not None do
11:     if newNode.word equals word then
12:       return TRUE
13:     else if newNode.prev is not None and newNode.prev.word equals phrase[rootIndex - count -1] then
14:       assign newNode.prev to newNode
15:       increase count by 1 and continue
16:     else if newNode.next is not None and newNode is not root and newNode.next.word equals phrase[rootIndex - count -1] then
17:       assign newNode.next to newNode
18:       increase count by 1 and continue
19:     else
20:       if newNode.prev is None then
21:         create a node and assign to newNode.prev
22:         return TRUE
23:       else if newNode.next is None and newNode is not root then
24:         create a node and assign to newNode.next
25:         return TRUE
26:       else if newNode.prev is not None and newNode.prev.word is not phrase[rootIndex - count -1] then
27:         if newNode.prev.termFreq $<$ termFreq then
28:           create a node and assign to newNode.prev
29:           return TRUE
30:         else
31:           return FALSE
32:       else if newNode.next is not None and newNode is not root and newNode.next.word is not phrase[rootIndex - count -1] then
33:         if newNode.next.termFreq $<$ termFreq then
34:           create a node and assign to newNode.prev
35:           return TRUE
36:         else
37:           return FALSE
38:       else
39:         return FALSE
40:   return FALSE
41: if wordIndex $>$ rootIndex then
42:   while count $<$ depth - 1 or newNode is not None do
43:     if newNode.word equals word then
44:       return TRUE
45:     else if newNode.next is not None and newNode.next.word equals phrase[rootIndex + count +1] then
46:       assign newNode.next to newNode
47:       increase count by 1 and continue
48:     else if newNode.prev is not None and newNode is not root and newNode.prev.word equals phrase[rootIndex + count +1] then
49:       assign newNode.prev to newNode
50:       increase count by 1 and continue
51:     else
52:       if newNode.next is None then
53:         create a Node and assign to newNode.next
54:         return TRUE
55:       else if newNode.prev is None and newNode is not root then
56:         crete a Node and assign to newNode.prev
57:         return TRUE
58:       else if newNode.next is not None and newNode.next.word is not phrase[rootIndex + count + 1] then
59:         if newNode.next.termFreq $<$ termFreq then
60:           create a Node and assign to newNode.next
61:           return TRUE
62:         else
63:           return FALSE
64:       else if newNode.prev is not None and newNode is not root and newNode.prev.word is not phrase[rootIndex + count + 1] then
65:         if newNode.prev.termFreq $<$ termFreq then
66:           create a Node and assign to newNode.next
67:           return TRUE
68:         else
69:           return FALSE
70:       else
71:         return FALSE
72:   return FALSE
```

passed to the *AddNode* function (see Algorithm 1.) for possible inclusion. This function returns *true* for successful inclusion of a word in the tree or if that word already exists in the tree. On the contrary, returns *false*, which stops processing of the rest of the words in the same direction and moves to the alternative side if that is not yet processed. For instance, if the function returns *false* for a word at the left side of γ, and the right side is not yet processed; it stops processing the other words at the left side and moves to the words at the right side and vice-versa.

Like tree formation, the tree processing also starts from γ followed by the words at the left side of γ and then, right side. Since γ is present in every σ_i', the *AddNode* function always returns *true* after increasing the μ value by 1. Considering a word ($w \in \sigma_i'$) at position w_p is passed to the function for possible inclusion in the tree. If $w_p < \gamma_p$, it is added to the left subtree; otherwise, the right subtree. Again, depth (w_d) is calculated to determine at which level w is added in the subtree. Furthermore, all the predecessors must be checked with the ones in σ_i' before inclusion. For that, an iterative procedure is applied which compares all the words from depths 0 to $d-1$ in σ_i' with their respective equivalent levels. The iterator increases when the word in a certain depth is same as the word in the equivalent level in the tree; otherwise, returns *false*.

When the iterator reaches at level l, three events can occur: i) there is no node in l, ii) there is only one node in l, and iii) there is two nodes in l. In case of the first event, a node is created incorporating w and initializing μ to 1; and then, add it as a left child for the left subtree or as a right child for the right subtree. For the second event, if the word in the node is same as w; then no node is added. Otherwise, a node is created incorporating w and initializing μ to 1; and add it as the child that is *NULL* at present at that level in the tree. Last event but not least, if both children already exist at that level; then, the node created with w replaces the node whose word has the lowest TF with respect to w. The reason is that any word with higher TF is highly likely to form a quality final keyphrase since generally keyphrases are not rare in a document. For that, if the lower TF node is a leaf node, the new node will replace it. Otherwise, if it is a root of a subtree, then the subtree is deleted from the tree and the new node is added in that position. This process is deemed complete when all the words of σ'_i are taken into account.

Update μ Values

The process of updating μ values starts immediately after the process of node addition is over for any σ'_i. This process is demonstrated in Algorithm 2.

Like tree formation and tree processing, this process also starts by determining γ_p in σ'_i. If γ_p is 0 but the root in the tree has a left child then, μ values of all the nodes in the left subtree are decreased. Similarly, if γ_p is $|\sigma'_i| - 1$, i.e., the rightmost node in the tree, but the root has a right child then, μ values of all the nodes in the right subtree are decreased. Afterwards, the μ value of the root is increased and the tree is traversed and compared starting from the left subtree followed by the right subtree using iterative procedures.

For any w when it reaches to its own level, l, three events may occur: *i*) w is absent in l, *ii*) w is present as a left child, and *iii*) w is present as a right child. For the first event, μ values of all the nodes in the left and right subtree are decreased. In the second case, μ value of the left child is increased; whereas, they are decreased for all nodes in the right subtree, and then, move to the next level. In case of the last event, μ value of the right child is increased; whereas, they are decreased for all nodes in the left subtree, and then, move to the next level. This procedure continues until all the words are taken into account.

An example of tree processing and updating μ values are demonstrated in Fig. 3.11. In this example, the tree shown in Fig. 3.10 and σ in Fig. 3.9 are utilized. Again, the tree is formed using σ_1 in σ, and the rest (i.e., σ') are utilized to process the tree. As in Fig. 3.11.a, since σ'_1 is *grid servic*, and both the words already exist in the tree in sequence, the tree remains in the same state as before. However, μ values of the nodes that contain *grid* and *servic* are increased, and all others are decreased. In Fig. 3.11.b, among the three words, only *mechan* does not exist in the right subtree at level 2 and it is added as left child. Afterwards, μ values are increased based on σ'_2. Similarly, the tree keeps amending with

Algorithm 2 Update μ values

```
Input: root, phrase

 1: find rootPosition in the phrase
 2: if root not found then
 3:     return
 4: else if rootPosition equals 0 and root.prev is not None then
 5:     decrease $\mu$ of all Nodes in left subree
 6: else if rootPosition equals len(phrase) - 1 and root.next is not None then
 7:     decrease $\mu$ of all Nodes in right subree
 8: increase $\mu$ value of root by 1
 9: assign root.prev to newNode
10: for i from rootPosition - 1 to 0 do
11:     if newNode is None then
12:         break
13:     if newNode.word equals phrase[i] then
14:         increase $\mu$ value of newNode by 1
15:     if newNode.prev is not None and $i - 1 > -1$ and newNode.prev.word equals phrase[i - 1] then
16:         if newNode.next is not None then
17:             decrease $\mu$ of all Nodes in right subree
18:             assign newNode.prev to newNode and continue
19:         else if newNode.next is not None and $i - 1 > -1$ and newNode.next.word equals phrase[i - 1] then
20:             if newNode.prev is not None then
21:                 decrease $\mu$ of all Nodes in left subree
22:                 assign newNode.next to newNode and continue
23:             else
24:                 decrease $\mu$ of all Nodes in left subree
25:                 decrease $\mu$ of all Nodes in right subree
26:                 break
27:     else
28:         decrease $\mu$ of all the nodes in the subtree where newNode is the root
29: assign root.next to newNode
30: for i from rootPosition + 1 to len(phrase) - 1 do
31:     if newNode is None then
32:         break
33:     if newNode.word equals phrase[i] then
34:         increase $\mu$ value of newNode by 1
35:     if newNode.next is not None and $i + 1 <$ len(phrase) and newNode.next.word equals phrase[i + 1] then
36:         if newNode.prev is not None then
37:             decrease $\mu$ of all Nodes in left subtree
38:             assign newNode.next to newNode and continue
39:         else if newNode.prev is not None and $i + 1 <$ len(phrase) and newNode.prev.word equals phrase[i + 1] then
40:             if newNode.next is not None then
41:                 decrease $\mu$ of all Nodes in right subtree
42:                 assign newNode.prev to newNode and continue
43:             else
44:                 decrease $\mu$ of all Nodes in left subree
45:                 decrease $\mu$ of all Nodes in right subree
46:                 break
47:     else
48:         decrease $\mu$ of all the nodes in the subtree where newNode is the root
```

every encountered σ_i' and μ values are also updated accordingly. This process keeps continuing until all the keyphrases in σ' are processed. Although this example demonstrated only expansion or no change of tree state, the shrinkage occurs in the keyphrase extraction phase.

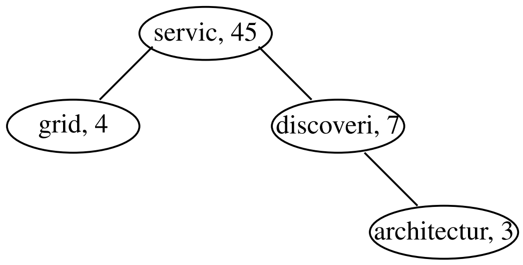


Figure 3.12. The resultant tree for *mamu* $= 2$ for the *KePhEx* tree in Fig. 3.11.

Keyphrase Extraction

In this step, final keyphrases are extracted from the *KePhEx* tree. This process is initiated by pruning the weak nodes from the tree. Here, weak nodes are selected based on their cohesiveness with respect to γ with an assumption that they may not be the parts of final keyphrases. For that, a constant value, named *minimum allowable* μ *(mamu)*, is utilized. A node whose μ value is lower than *mamu* is pruned from the tree. Hence, if that node is a root of a subtree than that entire subtree is also erased from the tree with the assumption that a weak root would form a weak subtree. Again, a *mamu* value must be selected with considerable attention as because a smaller *mamu* value results in many and/or long keyphrases; whereas a large *mamu* value results in lower an/or abbreviated keyphrases. Therefore, it is essential to find a suitable *mamu* value for improved performance of the system. Hence, this thesis utilizes an experiment to find a suitable *mamu* value (see Section 3.4.2). Again, this *mamu* value also provides flexibility during keyphrase extraction. Such a tree is depicted in Fig. 3.12, where *mamu* is considered as 2.

Afterwards, all paths from the root to the leaves are extracted to discover final keyphrases. Since this procedure is dissimilar to any conventional tree traversal technique (namely *preorder*, *inorder*, and *postorder*), they are not directly applicable in this case. Hence, *inorder* tree traversal technique is enhanced to perform the task, which is explained in Algorithm 3.

This algorithm extracts all the paths from root to leaf and separates them in left paths (paths from left subtree) and right paths (paths from right subtree), which are later

Algorithm 3 Find paths from leaf to root

```
    Input: root, nodeList
    Output: return leftPaths, rightPaths

Procedure ROOTTOLEAFPATHS(root, nodeList, leftPaths, rightPahts)
 1: if root is None then
 2:     return
 3: append root in nodeList
 4: ROOTTOLEAFPATHS(root.prev, nodeList, leftPaths, rightPahts)
 5: if node.prev is None and node.next is None then
 6:     create an empty container, x
 7:     for j from 0 to length(nodeList) - 1 do
 8:         append nodeList[j] in x
 9:     if length(nodeList) > 1 then
10:         if nodeList[0].prev equals nodeList[1] then
11:             append x in leftPaths
12:         else
13:             append x in rightPaths
14:     else
15:         append x in leftPaths
16:     return
17: ROOTTOLEAFPATHS(root.next, nodeList, leftPaths, rightPaths)
18: pop an element from the nodeList
```

processed to generate final keyphrases individually (one final keyphrase from one path) or collectively (by joining a path from the left subtree and a path from the right subtree) as demonstrated in Algorithm 4.

Now, in case of left paths, since they are extracted from root to leaf, they are unlikely to be the final keyphrases as they are aligned in reverse direction; and hence, misses the coherent relationship. Therefore, all left paths are reversed before extracting final keyphrases. Afterwards, all the words are acquired from each path and a keyphrase is formed. Then, its presence (entirely) is checked in χ as a candidate keyphrase or a part of candidate keyphrase. A similar technique is followed to extract keyphrases from right paths with an exception is that the paths are not reversed since they are already satisfying the coherent relationship conditions. After acquiring all the final keyphrases from the left and right paths,

Algorithm 4 Find Keyphrases from the KePhExTree

```
Input: $\mu$, finalPhraseList, root
Output: finalPhraseList

1:  prune the tree according to the $\mu$ value
2:  if root is None then
3:      return finalPhraseList
4:  else if root.$\mu$ ¡ $\mu$ then
5:      return finalPhraseList
6:  create empty lists, such as nodeList, leftPaths, rightPaths
7:  call the procedure at Algo. 3
8:  create two empty lists, namely leftPhraseList, rightPhraseList
    /* Append from leftPaths */
9:  for i from 0 to length(leftPaths) - 1 do
10:     create an empty list, named newPhrase
11:     for l from 0 to length(leftPaths[i] - 1 do
12:         create an empty list, named morePhrase
13:         insert leftPaths[i][l] in newPhrase at position 0
14:         copy newPhase in morePhrase
15:         append morePhrase in leftPhraseList
16:         if morePhrase does not exist in finalPhraseList and it
            is found in candidatePhraseList then
17:             append morePhrase to finalPhraseList
    /* Append from rightPaths */
18: for j from 0 to length (rightPaths) - 1) do
19:     create an empty list, named newPhrase
20:     for m from 0 to length(rightPahts[j]) - 1 do
21:         create an empty list, named morePhrase
22:         append rightPaths[j][m] in newPhrase
23:         copy newPhase in morePhrase
24:         append morePhrase in rightPhraseList
25:         if morePhrase does not exist in finalPhraseList and it
            is found in candidatePhraseList then
26:             append morePhrase to finalPhraseList
    /* Combine leftPhraseList and rightPhraseList */
27: for i from 1, length(leftPhraseList) - 1 do
28:     create an empty list, named newPhrase
29:     if length(leftPhraseList[i]) > 1 then
30:         for l from 0, length(leftPhraseList[i]) - 1 do
31:             append leftPhraseList[i][l] to newPhrase
32:     else
33:         continue
34:     for j from 0, len(rightPhraseList) - 1 do
35:         if length(rightPhraseList[j]) > 1 then
36:             create an empty list, named morePhrase
37:             copy newPhase in morePhrase
38:             for r from 1, length(rightPhraseList[j]) do
39:                 append rightPhraseList[j][r] in morePhrase
40:             if morePhrase does not exist in finalPhraseList and
                it is found in candidatePhraseList then
41:                 append morePhrase is finalPhraseList
42: return finalPhraseList
```

they are concatenated to generate more long and meaningful keyphrases. Again, these keyphrases will qualify as final keyphrases if they are entirely found in χ as candidate keyphrases or part of candidate keyphrases.

Flexibility during Keyphrase Extraction

The proposed technique offers flexibility in keyphrase extraction via employing the *mamu* values. As an example, the Table 3.2 is generated using the tree in Fig. 3.12. As expected, for different *mamu* values different final keyphrases are generated. These keyphrases also differ in length and quantity. For instance, the longest keyphrase generated by *mamu* values from 1 *to* 3 is 4, whereas, it is 3 for *mamu* value 4, 2 for *mamu* values from 5 *to* 7 and so on. On the other hand, for *mamu* values from 1 *to* 4, 3 final keyphrases are extracted; whereas, it is only 1 for *mamu* values from 5 *to* 45 and 0 afterwards. From

Table 3.2. Final Keyphrases from the resultant tree in Fig 3.12

	mamu value	Final Keyphrase
1	1 to 3	grid servic
2	1 to 3	servic discoveri architectur
3	1 to 3	grid servic discoveri architectur
4	4	grid servic
5	4	servic discoveri
6	4	grid servic discoveri
7	5 to 7	servic discoveri
8	8 to 45	servic
9	≥ 46	—

here, we can conclude that a greedy approach may choose a lower *mamu* value and hence, would get considerably many and/or lengthy keyphrases; but the quality would be a little bit compromised. On the other hand, a conservative approach may choose a large *mamu* value which will in turn provide considerably lower and/or mostly abbreviated keyphrases. Hence, to receive a desired level of performance, *mamu* value must be set properly. To realize this, an experimental evaluation is performed in Section 3.4.2 and the results are analyzed with detail evidences.

After extracting all the final keyphrases from the tree for a γ, the next γ is chosen from the list η and the same procedure is repeated again. It continues until all the nouns in η are considered as γ. After finish extracting all the final keyphrases, they are passed for ranking and selecting.

3.4.3 Ranking and Selecting Final Keyphrases

Generally, various applications, such as academic literature, document indexing, and so on, utilize a certain number of top keyphrases only. Therefore, an automatic keyphrase extraction technique must offer the most relevant top-*N* keyphrases to these applications. Hence, most of the automatic keyphrase extraction techniques irrespective of supervised or unsupervised approaches employ one or more classification or ranking strategies, respectively.

Again, most of the techniques in a certain group differ from each other based on only ranking or classifying. Therefore, keyphrase extraction is also accounted as a ranking problem.

In the proposed ranking technique, only TF is employed along with the μ value as follows to calculate weight, ω of a keyphrase p:

$$\omega_p = \sum_{k=1}^{N} tf_k \times \sum_{k=1}^{N} \mu_k \qquad 3.1$$

Here, N is the number of words in a keyphrase, p. In Eq. 3.1, instead of averaging each factor, summation is performed to eliminate the biasness towards the single terms. The first factor is to identify the popularity of that particular keyphrase in a document and the second factor is for realizing the cohesiveness of every word in that keyphrase to γ.

After performing calculation of ω values for all keyphrases, they are sorted to arrange them in rank. Since the quantity of final keyphrases is limited, any sorting algorithm is suitable. In the proposed system, the *quick sort* algorithm is applied to perform the task rapidly. After ranking, these keyphrases are ready for the selection. Now, when a user or an application seeks for any N keyphrases, the system will provide top-N keyprhases from the rank 1 to N, respectively.

3.5 Experimental Setup

Since the proposed technique is an unsupervised machine learning based technique, its performance is compared it with other relevant unsupervised techniques. For this, both statistical (TD-IDF and YAKE) and graph based (singleRank (SR), positionRank (PR), topicRank (TR), and multipartiteRank (MR)) keyphrase extraction techniques are taken into account. All of these techniques are evaluated under a uniform experimental setup (see Section 3.5.2) taking a benchmark dataset into consideration (see Section 3.5.1).

Table 3.3. Number of documents per topic in the four ACM document classifications

Dataset	Total	Document Topic			
		C	H	I	J
Train	144	34	39	35	36
Test	100	25	25	25	25

Table 3.4. Keyphrase distribution of gold standard in different datasets

Dataset	Author-assigned	Reader-assigned	Combined
Training	559	1824	2223
Test	387	1217	1482

3.5.1 Corpus Details

The proposed technique was tested on the *SemEval-2010* benchmark dataset (Kim, Medelyan, et al., 2010). This dataset is comprised of a train and a test dataset along with other datasets that are collected from the *ACM Digital Library*. In the corpus, to ensure the variability in terms of topics, all the papers are clustered in four groups following four 1998 ACM classifications: C2.4 — Distributed Systems, H3.3 — Information Search and Retrieval, I2.11 — Distributed Artificial Intelligence — Multiagent Systems, and J4 — Social and Behavioral Sciences — Economics. The distribution of documents in the corpus is mentioned in Table 3.3.

All the documents in the corpus are in plain text and the average length of these documents is about 2000 words. Although, the *XML* version of this dataset exists, we prefer text dataset since the former one is heavy, verbose, and rare. For comparison, gold standard keyphrases were employed that came along with the dataset and comprised of author-assigned keyphrases and reader-assigned keyphrases. Table 3.4 exhibits the distribution of author- and reader-assigned keyphrases in the corpus.

3.5.2 Evaluation Metrics

Three prominent and relevant metrics, namely, *precision (ρ)*, *recall (ς)*, and *F1-score (ϕ)* have been used for comparing the proposed technique's performance with other considered techniques. Here, ρ is the ratio of correctly predicted positive values with respect to the total predicted values. It can be calculated using the following formula:

$$Precision(\rho) = \frac{\kappa_{correct}}{\kappa_{extract}} \tag{3.2}$$

where, $\kappa_{correct}$ is the number of correctly matched keyphrases with gold standard keyphrases and $\kappa_{extract}$ is the number of extracted keyphrases from a document, i.e., value of N in case of extracting top-N keyphrases.

On the other hand, ς is the ratio of correctly predicted positive values with respect to the actual positive values; and can be calculated as follows:

$$Recall(\varsigma) = \frac{\kappa_{correct}}{\kappa_{standard}} \tag{3.3}$$

where, $\kappa_{standard}$ is the number of keyprhases in gold standard keyphrase list for that particular document. Again, ϕ is the weighted average of ρ and ς, which can be calculated using the following formula:

$$F1-score(\phi) = \frac{2 \times \rho \times \varsigma}{\rho + \varsigma} \tag{3.4}$$

This metric is much more sophisticated than conventional accuracy metric since it takes both false positives and false negatives into consideration.

3.5.3 Implementation Details

The proposed technique was implemented using Python3 and utilized several other necessary packages, such as PorterStemmer (Kantrowitz, Mohit, & Mittal, 2000; Karaa & Gribâa, 2013), Sent_tokenize, Word_tokenize of Natural Language Tool Kit (Bird & Loper, 2004; Sugiyama & Kan, n.d.), Regular Expression (Barker & Cornacchia, 2000; *Regular Expression HOWTO*, n.d.), and so on. Note that all the words are stemmed initially before passing them to the processing phase employing porterStemmer. Again, for gold standard keyphrases, no such processing is required since they are already stemmed.

For other compared techniques, Python Keyphrase Extraction (PKE) (Boudin, 2016b) — which is an open source python-based keyphrase extraction toolkit — is utilized. Here, we would like to mention that for all the experiments, whatsoever, an uniform experimental environment is offered to ensure a level playing ground for all the techniques. For the compared techniques, Application Programming Interface (API) of those techniques are employed to acquire top *N* keyprhases. Afterwards, these acquired keyphrases are compared with the gold standard keyphrases; and then, metrics are calculated accordingly. All experiment codes will be uploaded in GitHub (*GitHub*, n.d.) for further reference (Rabby & Azad, 2018).

3.6 Summary

This chapter has addressed the topic of keyphrase extraction from scientific publications, going through an account of previous efforts, the variety of keyphrase features currently in use, as well as the presentation of a novel approach. The approach is Tree-based, utilizing only a simplistic set of features derived from the input document to carry out the extraction task. It identifies, filters and merges keyphrase candidates using linguistic methods, such as POS tagging, Streaming etc. It then weighs the candidates using TF and mamu value, to yield a dynamic ranking of their significance within the document. Finally, it provides the flexibility to extract keyphrases that occur in every part of a document. The system implementing the proposed method, TeKET, was evaluated in one setting: a benchmark against the state-of-the-art. This scenario yielded promising outcomes.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Preamble

This section includes the results that are acquired from the experiments along with their detail analyses. Again, since there are some parameters that have direct influence on the performance of the proposed technique; hence, relevant experiments are performed to select suitable values for those parameters. This action is performed before comparing the performance of the proposed technique with other considered techniques and described in the subsequent section in details.

4.2 Parameter Value Selection

Among various parameters of the proposed technique, two parameters have definite impacts on the performance, which are: lsaf and mamu. Here, the former parameter is utilized to filter out all non-popular candidate keyphrases from the list and the latter plays an important role in extracting keyphrases from the resultant tree. As mentioned earlier, a lower mamu value would result in many but low quality keyphrases, whereas, a high mamu value would result in few but abbreviated keyphrases. Therefore, it is necessary to determine, which mamu value would give the superlative performance.

For determining the suitable lsaf value, it is varied from 0 to 5 in our conducted experiments for two arbitrarily selected mamu values. The experiments are performed on test dataset by acquiring top-N keyphrases, where $N = 5$, 10, and 15. The acquired results are demonstrated in Table 4.1. The highest performance shown for F1 value is 15.6 for top-15

Table 4.1. Performance of proposed technique for various lsaf values for two arbitrarily selected μ values on test dataset

lsaf	μ	Top 5			Top 10			Top 15		
		P	R	F1	P	R	F1	P	R	F1
1	0	20.3	7.1	**10.5**	16.2	11.5	**13.3**	13.5	14.1	**13.7**
1	2	20.5	7.2	**10.6**	16.8	11.9	**13.8**	14.0	14.6	**14.2**
2	0	21.3	7.6	**11.2**	17.2	12.3	**14.2**	14.3	15.2	**14.6**
2	2	21.9	7.8	**11.5**	17.0	12.1	**14.1**	14.4	15.3	**14.7**
3	0	21.3	7.6	**11.1**	17.7	12.6	**14.6**	14.9	15.8	**15.3**
3	2	21.3	7.6	**11.1**	17.8	12.6	**14.6**	15.3	16.1	**15.6**
4	0	21.6	7.7	**11.28**	17.9	12.71	**14.74**	15.2	16.1	**15.5**
4	2	21.6	7.7	**11.28**	17.9	12.71	**14.75**	15.3	16.2	**15.6**
5	0	21.6	7.7	**11.28**	17.9	12.71	**14.74**	15.1	16	**15.4**
5	2	21.6	7.7	**11.28**	17.9	12.71	**14.75**	15.2	16.1	**15.5**

Table 4.2. Performance of proposed technique for various μ values on test dataset

μ	Top 5			Top 10			Top 15		
	P	R	F1	P	R	F1	P	R	F1
0	21.3	7.6	**11.1**	17.7	12.6	**14.6**	14.9	15.8	**15.3**
1	21.3	7.6	**11.1**	17.8	12.6	**14.6**	15.3	16.2	**15.6**
2	21.3	7.6	**11.1**	17.8	12.6	**14.6**	15.3	16.1	**15.6**
3	21.5	7.7	**11.2**	17.7	12.5	**14.5**	15.1	15.9	**15.4**
4	21.7	7.8	**11.4**	17.9	12.7	**14.7**	15.0	15.9	**15.3**
5	21.3	7.6	**11.2**	17.8	12.6	**14.6**	14.7	15.6	**15.0**

Table 4.3. Performance of proposed technique for various μ values on train dataset

μ	Top 5			Top 10			Top 15		
	P	R	F1	P	R	F1	P	R	F1
0	17.6	6.0	**8.8**	14.8	9.9	**11.7**	13.3	13.6	**13.2**
1	17.6	6.0	**8.8**	14.7	9.9	**11.6**	13.1	13.5	**13.1**
2	17.9	6.1	**9.0**	14.5	9.8	**11.5**	13.2	13.6	**13.2**
3	17.6	5.9	**8.8**	14.4	9.7	**11.4**	13.0	13.3	**13.0**
4	17.6	5.9	**8.8**	14.5	9.8	**11.5**	13.1	13.4	**13.0**
5	17.4	5.9	**8.7**	14.4	9.7	**11.5**	13.1	13.5	**13.1**

keyphrases by $lsaf$ values 3 and 4; whereas, the lowest performance shown is 7.1 for top-5 keyphrases by $lsaf$ value 1. It is because, a lower value of $lsaf$ incorporates non-popular keyphrases during ranking; and thus, entice ranking approach. From the results, it is evident that with increasing lsaf value, F1 value increases for any mamu value until $lsaf = 3$; afterwards, it becomes almost steady. Hence, 3 could be considered as the threshold value of lsaf; and is utilized in other experiments.

Again, to select a suitable mamu value, we conduct experiments varying mamu values from 0 to 5, fixing lsaf to 3, and taking test and train datasets of the corpus into consideration. For both datasets, results are acquired for top N keyphrases, where $N =$ 5, 10, and 15. All the acquired results are stated in Table 4.2 and 4.3 for test and train datasets, respectively. They are also plotted using contour graphs in Fig. 4.1 and 4.2 for more depictions.

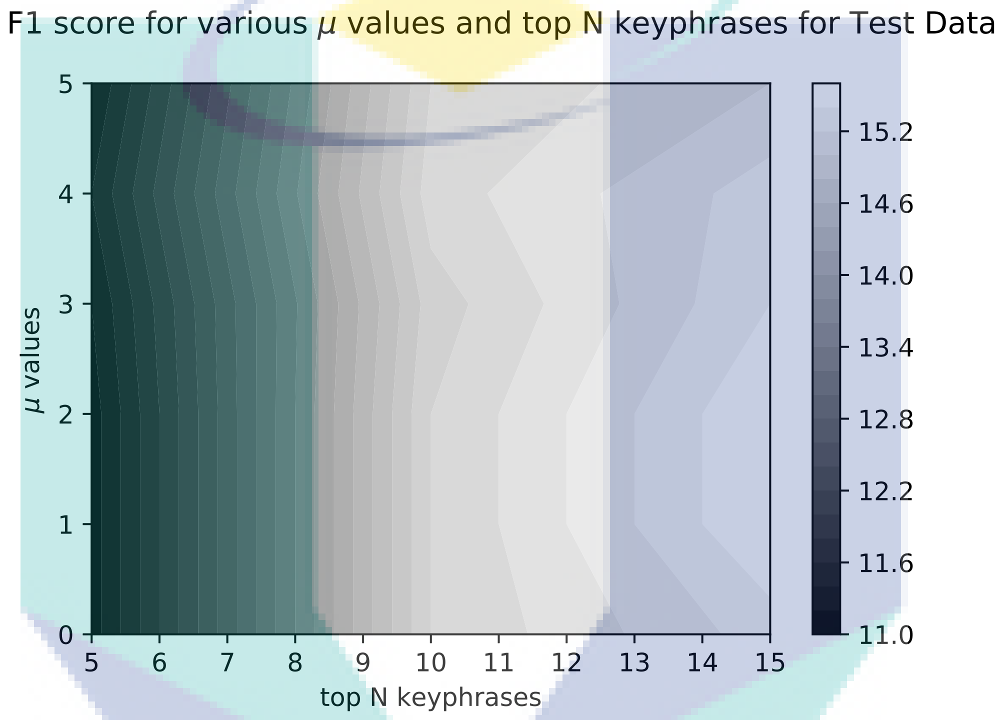


Figure 4.1. Performance of the proposed technique for various μ values on test dataset

As could be observed from the tables as well as from the figures is that performance differences in several mamu values are not as evident as lsaf values since we have already filtered out non-popular keyphrases by selecting $lsaf = 3$. The highest F1 achieved is 15.6 for test data and 13.2 from train data, and both cases, it is achieved by $mamu = 2$. Again, as could be observed that for most of the cases with increasing mamu values performance increases to a certain point, and afterwards, it decreases. In our case, $mamu = 2$ is that threshold for both the datasets. The reason is that it maintains the trade-off between the keyphrase length and quantity. On the other hand, smaller mamu values produce considerably many and/or lengthy keyphrases; but the quality is a little bit compromised; whereas, higher mamu values attain considerably lower and/or mostly abbreviate keyphrases. In the

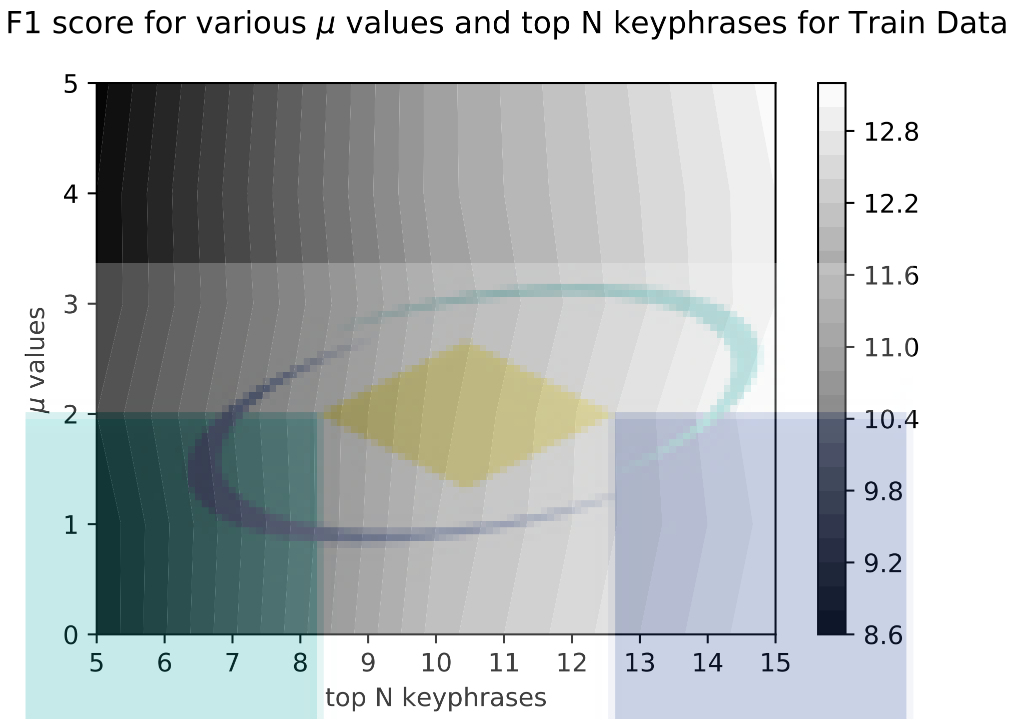


Figure 4.2. Performance of the proposed technique for various μ values on train dataset

latter case, since lengthy keyphrases are ignored; therefore, the performance is also a little bit compromised. Hence, $mamu = 2$ is locked for the rest of the experiments.

4.3 Results Analyses

Here, we would like to note that the performance of all the technique would have improved if 15% of the reader-assigned keyphrases that are absent would have appeared in the text, and if 19% of the author-assigned keyphrases that are absent would have appeared in the text. Hence, all the results in this thesis are based on 85% and 81% for the reader- and author-assigned keyphrases, respectively.

For all the techniques, three experiments are performed for each dataset with a target of extracting top-N keyphrases, where $N = 15$ is preferred in many literatures Kim, Medelyan, et al. (2010); Kim, Medelyan, Kan, and Baldwin (2013); and hence, is our choice. Again, once we have top-15 keyphrases, we can derive top-5 and top-10 keyphrases from there. These experiments are performed for — i) reader-assigned keyphrases, ii) author-assigned keyphrases, and iii) combined keyphrases (combines reader- and author-

Table 4.4. Performance of different unsupervised machine learning based keyphrase extraction techniques for reader-assigned keyphrases on test dataset

Approach	Technique	Top 5			Top 10			Top 15		
		P	R	F1	P	R	F1	P	R	F1
Graph-base	TopicRank	9.4	4.0	**5.5**	7.8	6.6	**7.1**	6.6	8.4	**7.3**
	PositionRank	3.8	1.6	**2.3**	4.4	3.8	**4.0**	3.9	5.1	**4.4**
	SingleRank	1.9	0.8	**1.2**	1.8	1.5	**1.6**	1.7	2.2	**1.9**
	Multipartite Rank	11.3	4.8	**6.7**	9.2	7.8	**8.4**	8.0	10.3	**8.9**
Statistical-base	TF-IDF	11.1	4.7	**6.6**	7.4	6.4	**6.8**	6.9	8.9	**7.5**
	YAKE	12.7	5.5	**7.7**	12.0	10.4	**11.1**	10.9	14.1	**12.2**
Tree-base (proposed)	TeKET	16.5	7.2	**10.0**	14.5	12.6	**13.4**	12.5	16.1	**13.9**

Table 4.5. Performance of different unsupervised machine learning based keyphrase extraction techniques for author-assigned keyphrases on test dataset

Approach	Technique	Top 5			Top 10			Top 15		
		P	R	F1	P	R	F1	P	R	F1
Graph-base	TopicRank	6	7.2	**6.3**	3.9	9.5	**5.4**	3.1	11.2	**4.7**
	PositionRank	1.2	1.8	**1.3**	1.5	3.9	**2.0**	1.5	6.8	**2.4**
	SingleRank	0.4	0.4	**0.3**	0.3	0.7	**0.4**	0.2	1.2	**0.41**
	Multipartite Rank	6.4	8.1	**6.9**	4.4	11.0	**6.1**	3.7	13.7	**5.7**
Statistical-base	TF-IDF	6.4	7.9	**6.8**	4.9	11.9	**6.8**	4.3	16.4	**6.7**
	YAKE	7.8	10.3	**8.6**	6.8	18.6	**9.8**	6.2	2.4	**9.8**
Tree-base (proposed)	TeKET	8.8	11.6	**9.7**	7.3	19.8	**10.4**	6.1	24.2	**9.6**

Table 4.6. Performance of different unsupervised machine learning based keyphrase extraction techniques for combined keyphrases on test dataset

Approach	Technique	Top 5			Top 10			Top 15		
		P	R	F1	P	R	F1	P	R	F1
Graph-base	TopicRank	12.3	4.2	**6.3**	9.4	6.5	**7.6**	8	8.3	**8.1**
	PositionRank	4.2	1.4	**2.1**	5.1	3.6	**4.1**	4.6	4.9	**4.7**
	SingleRank	2.2	0.7	**1.1**	1.9	1.2	**1.5**	1.8	1.8	**1.8**
	Multipartite Rank	13.9	4.8	**7.1**	11.1	7.8	**9.1**	9.5	10.1	**9.7**
Statistical-base	TF-IDF	14.3	5.1	**7.5**	10.3	7.4	**8.6**	9.4	10.1	**9.6**
	YAKE	16.9	6.0	**8.83**	14.9	10.6	**12.3**	13.5	14.3	**13.8**
Tree-base (proposed)	TeKET	21.3	7.6	**11.1**	17.8	12.6	**14.6**	15.3	16.1	**15.6**

assigned keyphrases). The acquired results for test dataset are shown in Table 4.4, 4.5, and 4.6 for reader-assigned, author-assigned, and combined keyphrases, respectively.

From the tables, it could be observed that generally, statistical-base techniques performed better than graph-based techniques. It is because, graph-based techniques are not good in capturing the cohesiveness of words in a keyphrase.

On the other hand, statistical-based techniques are simple to implement and utilize basic features, like term frequency, inverse document frequency, word positions, word relatedness to a context, and so on, to extract the most descriptive terms in a document. Although, these techniques utilize some basic statistical knowledge to find top-*N* keyphrases, they perform better is because — aforementioned statistical characteristics of top keyphrases repeat over and over in most of the documents.

Again, among all the graph-based techniques, SR performs the worst in terms of all the considered metrics. The highest F1 achieves is only 1.9 for top-15 in the case of reader-assigned gold standard keyphrases; whereas, the lowest is 0.3 for top-5 keyphrases in the case of author-assigned gold standard keyphrases. It is because, SR assigns higher scores to long but non-significant keyphrases. In details, SR assign the weights of the edges with the number of co-occurrences. Afterwards, keyphrases are extracted in the form of noun phrases and then, raked based on the sum of the significance of the words they contain. Therefore, non-significant long keyphrases receive higher scores than abbreviated keyphrases.

With respect to SR, PR outperforms the former in terms of all the metrics and for all top-*N* keyphrases. This happens because it incorporates the position information of a word and its occurrences into a biased PageRank to score words. It receives an average F1 score of 3.57 for reader-assigned keyphrases, 1.9 for author-assigned keyphrases, and 3.63 for combined keyphrases for all top-*N* cases that we considered in this thesis. However, it fails to ensure topical coverage and diversity that are not naturally handled by this kind of graphs.

On the other hand, due to taking the topical coverage into account, TR overpowers PR technique for any metric or any parameter, which was absent in the latter technique. Here, topic relations are accounted to find the semantic relatedness between the candidate keyphrases they instantiate. It demonstrates an average performance improvement of 93.55% over PR for reader-assigned keyphrases, 216.82% for author-assigned keyphrases, and 119.24% for combined keyphrases. Again, F1 value of top-5 keyphrases contributes more in these performance differences — around 140% for reader-assigned, 385% for author-assigned, and 200% for combined keyphrases. Although, it maximizes the topical coverage, but it suffers from several limitations. For instance, all candidates under a single topic are considered equally; and therefore, post-ranking heuristics are necessary to select the most representative keyphrases from each topic. Again, if any error occurs while forming topics, it will propagate throughout the model and thus, negatively impacts its performance.

Since MR resolves the issue of error propagation, it performs superiorly over TR, and thus over SR and PR. To resolve this issue, MR utilizes the multipartite graph; and hence, the name, which connects sets of topic related candidates tightly. The average F1 receives for reader-assigned keyphrases is 8; whereas, it is for author-assigned keyphrases is 5.47, and combined keyphrases is 7.33. However, it struggles with selecting the most representative candidates due to clustering errors, where candidate keyphrases could be wrongly assigned to the same topic.

Among the statistical-based approaches, TF-IDF performs comparably to MR for all the metrics and attributes. For instance, it receives an average F1 of 7 for reader-assigned keyphrases, 6.8 for author-assigned keyphrases, and 8.57 for combined keyphra-ses. In TF-IDF, IDF provides informativeness and TF provides aboutness. Here, TF discriminates the non-popular keyphrases from the popular keyphrases in a document; whereas, IDF discriminates between informative and non-informative keyphrases across the documents. A keyphrase receives high IDF when it is rare along the collections. However, it favors single terms or bias towards single terms over compound terms and hence, demonstrates considerably lower performance over YAKE on test dataset.

In case of YAKE, it takes five features into account, namely casing, word position, word frequency, word relatedness to Context, and word in the different sentences, to rank keyphrases. Since many quality keyphrases pursue these statistical features unconsciously, it shows better performance over TF-IDF technique. It receives an average performance enhancement of 47.53% for reader-assigned keyphrases, 38.95% for author-assigned keyphrases, and 34.83% for combined keyphrases. However, since candidate keyprhases are generated using *N*-grams technique, where *N* is 1-, 2-, and 3-grams, a considerably large number of keyphrases are generated, which entices ranking procedure.

In terms of any metric and any attribute, TeKET outperforms the other techniques that are considered in this evaluation significantly. For instance, it outperforms YAKE by 21.51% for F1 measure on an average in case of reader-assigned keyphrases, 5.61% in case of author-assigned keyphrases, and 20.49% in case of combined keyphrases. Again, our proposed technique receives the highest F1 value among all the techniques, i.e., 15.6, for top-15 keyphrases in case of combined gold standard keyphrase list. One of the reasons of its excellent performance is that it extracts final keyphrases from candidate keyphrases using the KePhEx tree, and hence, consider most likely keyphrases during ranking. In addition, it utilizes two factors (TF and μ) in ranking, where the preceding factor is utilized to discriminate non-popular keyphrases from popular keyphrases and the latter factor is utilized to find the cohesiveness of various words in a keyphrase with respect to the root. Again, in the calculation, summation is preferred over average to facilitate longer keyphrases.

In Fig. 4.3, 4.4, and 4.5, F1 scores of various techniques for top-5, -10, and -15 keyphrases are shown in case of reader-assigned, author-assigned, and combined gold standard keyphrases. Like the table, SR demonstrates the substandard performance. Although, PR outperforms SR, but it falls short in front of TR for a considerably larger margin. Again, MR and TF-IDF demonstrate comparable performance in case of all three top-*N* values.

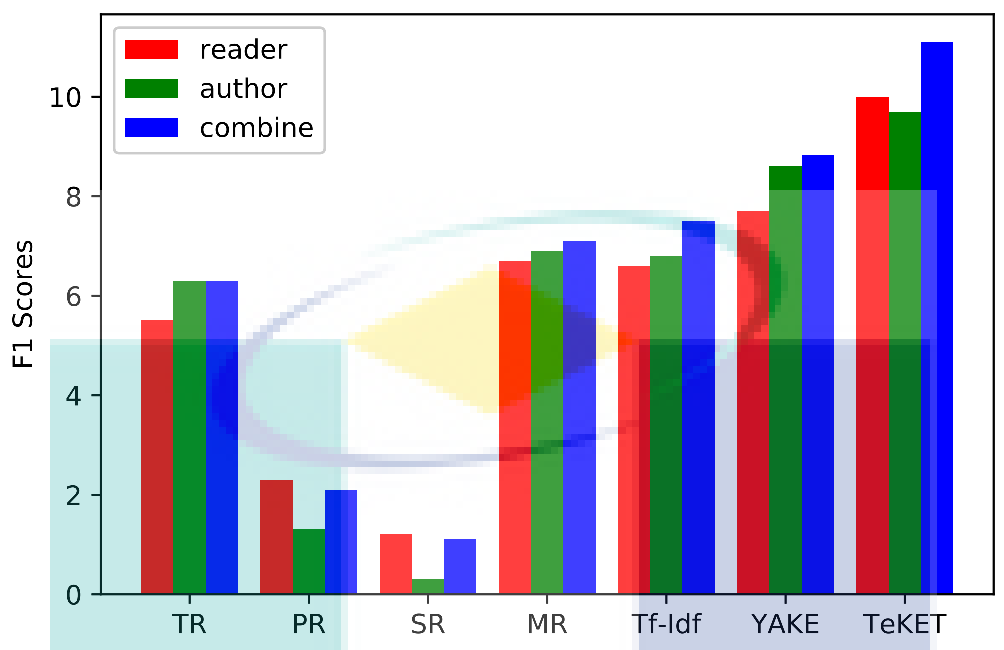


Figure 4.3. F1-Scores of various unsupervised keyphrase extraction techniques for Top-5 keyphrases employed on test dataset

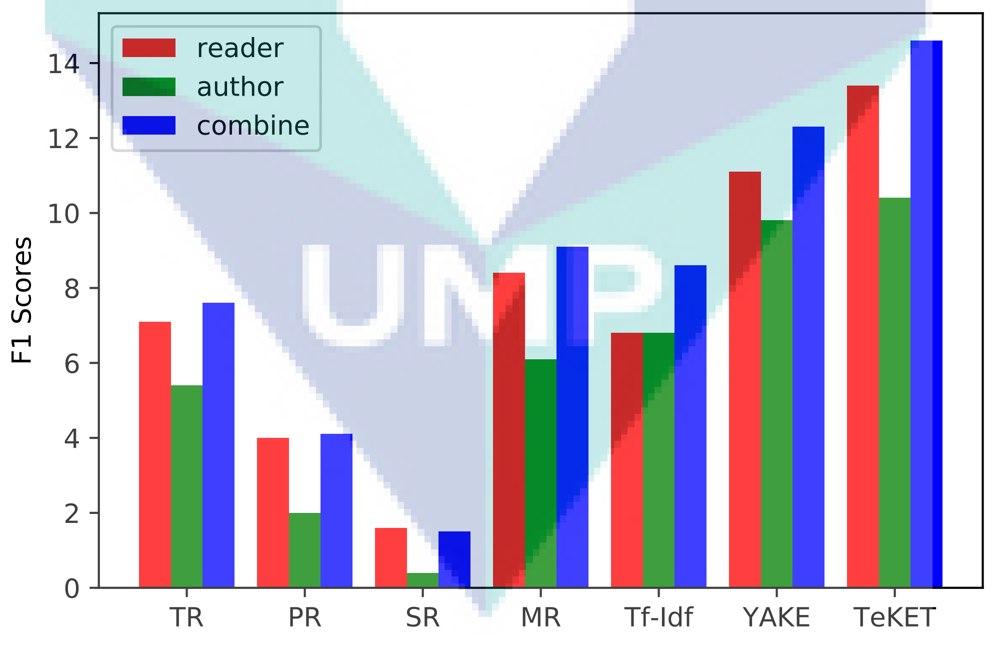


Figure 4.4. F1-Scores of various unsupervised keyphrase extraction techniques for Top-10 keyphrases employed on test dataset

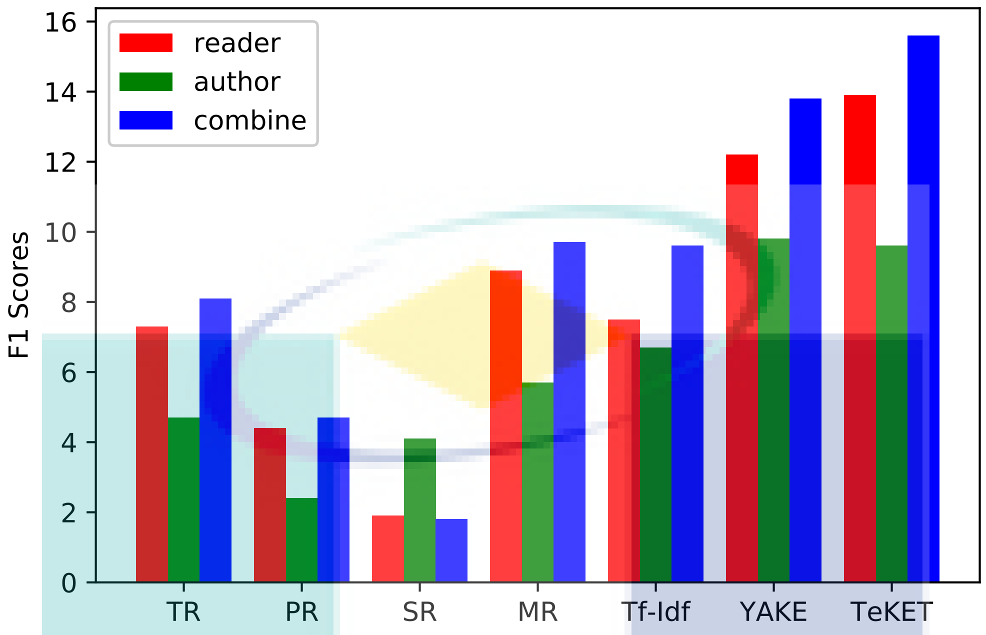


Figure 4.5. F1-Scores of various unsupervised keyphrase extraction techniques for Top-15 keyphrases employed on test dataset

Although, YAKE performs better over other considered keyphrase extraction techniques, but our proposed technique overpowers all others. The reasons of their performance differences are same as before.

Table 4.7. Performance of different unsupervised machine learning based keyphrase extraction techniques for reader-assigned keyphrases on train dataset

Approach	Technique	Top 5			Top 10			Top 15		
		P	R	F1	P	R	F1	P	R	F1
Graph-base	TopicRank	8.0	3.1	**4.5**	6.3	5.0	**5.5**	5.4	6.5	**5.8**
	PositionRank	3.6	1.4	**2.0**	3.2	2.6	**2.9**	2.9	3.6	**3.2**
	SingleRank	1.5	0.58	**0.84**	0.97	0.77	**0.85**	1.2	1.4	**1.3**
	Multipartite Rank	8.8	3.5	**5.0**	7.5	6.0	**6.6**	6.3	7.7	**6.8**
Statistical-base	TF-IDF	7.5	3.1	**4.4**	5.9	4.9	**5.3**	4.7	5.8	**5.2**
	YAKE	7.4	3.0	**4.2**	7.0	5.6	**6.1**	6.7	8.1	**7.2**
Tree-base (proposed)	TeKET	14.0	5.7	**8.0**	11.0	9.0	**9.8**	10.1	12.7	**11.1**

The acquired results for train data are plotted in Table 4.7, 4.8, and 4.9 for reader-assigned, author-assigned, and combined keyphrases respectively. Alike train data, all the results are acquired for three metrics and compared with top-N keyphrases, where $N = 5$,

Table 4.8. Performance of different unsupervised machine learning based keyphrase extraction techniques for author-assigned keyphrases on train dataset

Approach	Technique	Top 5			Top 10			Top 15		
		P	R	F1	P	R	F1	P	R	F1
Graph-base	TopicRank	3.4	4.7	**3.9**	2.7	7.6	**3.8**	2.2	9.4	**3.5**
	PositionRank	1.8	2.4	**2.0**	1.7	5.0	**2.5**	1.3	5.8	**2.2**
	SingleRank	0.4	0.6	**0.4**	0.6	1.8	**0.9**	0.6	2.6	**1.0**
	Multipartite Rank	4.4	6.2	**5.0**	3.4	9.8	**5.0**	3.0	12.7	**4.7**
Statistical-base	TF-IDF	4.2	5.9	**4.6**	3.1	8.0	**4.3**	2.6	10.0	**4.0**
	YAKE	5.4	7.2	**6.0**	4.8	12.8	**6.8**	4.5	17.9	**7.1**
Tree-base (proposed)	TeKET	8.4	11.3	**9.4**	6.7	18.1	**9.6**	6.1	24.4	**9.6**

Table 4.9. Performance of different unsupervised machine learning based keyphrase extraction techniques for combined keyphrases on train dataset

Approach	Technique	Top 5			Top 10			Top 15		
		P	R	F1	P	R	F1	P	R	F1
Graph-base	TopicRank	9.5	3.0	**4.5**	7.5	4.8	**5.8**	6.5	6.3	**6.3**
	PositionRank	4.8	1.5	**2.3**	4.4	2.9	**3.4**	3.8	3.8	**3.8**
	SingleRank	1.6	0.5	**0.7**	1.3	0.9	**1.0**	1.7	1.6	**1.6**
	Multipartite Rank	11.2	3.6	**5.4**	9.3	6.1	**7.3**	8.1	7.9	**7.9**
Statistical-base	TF-IDF	10.1	3.4	**5.0**	8.1	5.4	**6.4**	6.4	6.4	**6.3**
	YAKE	10.8	3.5	**5.3**	10.0	6.6	**7.9**	9.3	9.2	**9.1**
Tree-base (proposed)	TeKET	17.9	6.1	**9.0**	14.5	9.8	**11.5**	13.2	13.6	**13.2**

10, and 15. The average F1 scored by the SR technique for all cases is 1.05, which is the lowest among all. On the other hand, it is 2.7, 4.84, and 6.03 for PR, TR, and MR, respectively. Due to utilizing multipartite graph, it is extracting more gold standard keyphrases than others. Again, the average F1 scores for TF-IDF and YAKE are 5.06 and 6.63, respectively. Unlike test data, TF-IDF fails to achieve comparable performance to MR for train data; however, the latter almost catches YAKE in terms of F1 score. Conversely, our proposed technique overpowers all the considered techniques with an average F1 score of 10.13 for the reasons that are stated before.

The F1 scores of various gold standard keyphrase classes (reader, author, and combined) for train data are shown in Fig. 4.6, 4.7, and 4.8 for top-5, -10, and -15 keyphrases, respectively. Like the previous cases, performances of SR, PR, and TR remain in the same

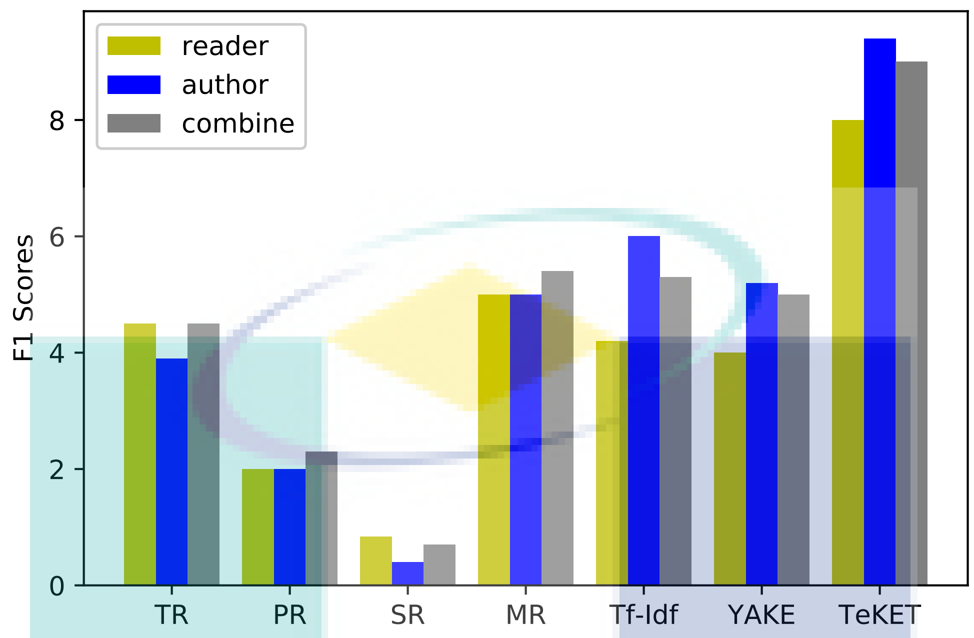


Figure 4.6. F1-Scores of various unsupervised keyphrase extraction techniques for Top-5 keyphrases employed on train dataset

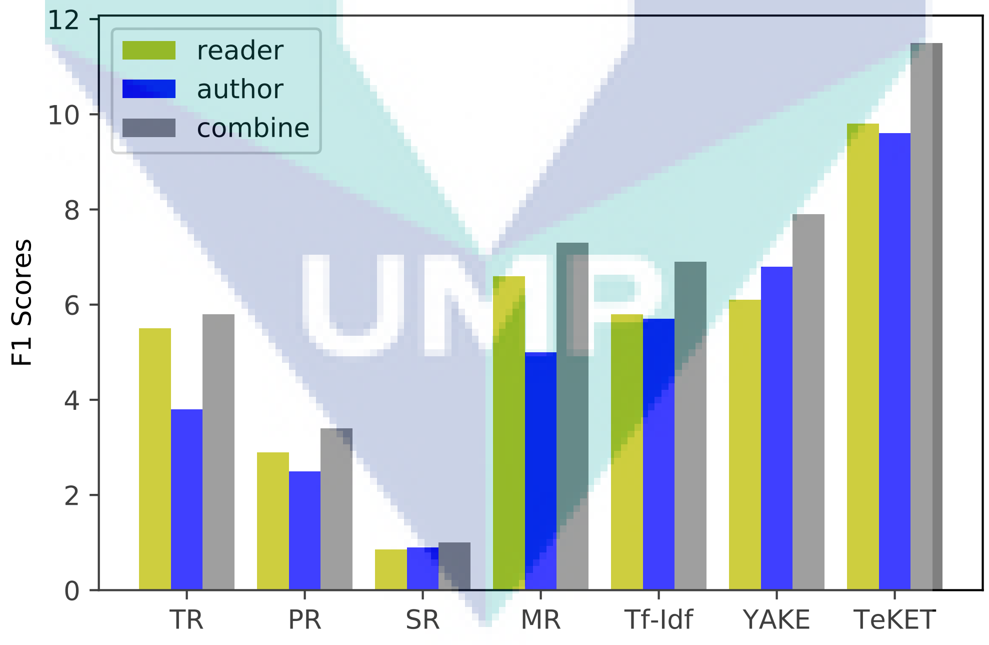


Figure 4.7. F1-Scores of various unsupervised keyphrase extraction techniques for Top-10 keyphrases employed on train dataset

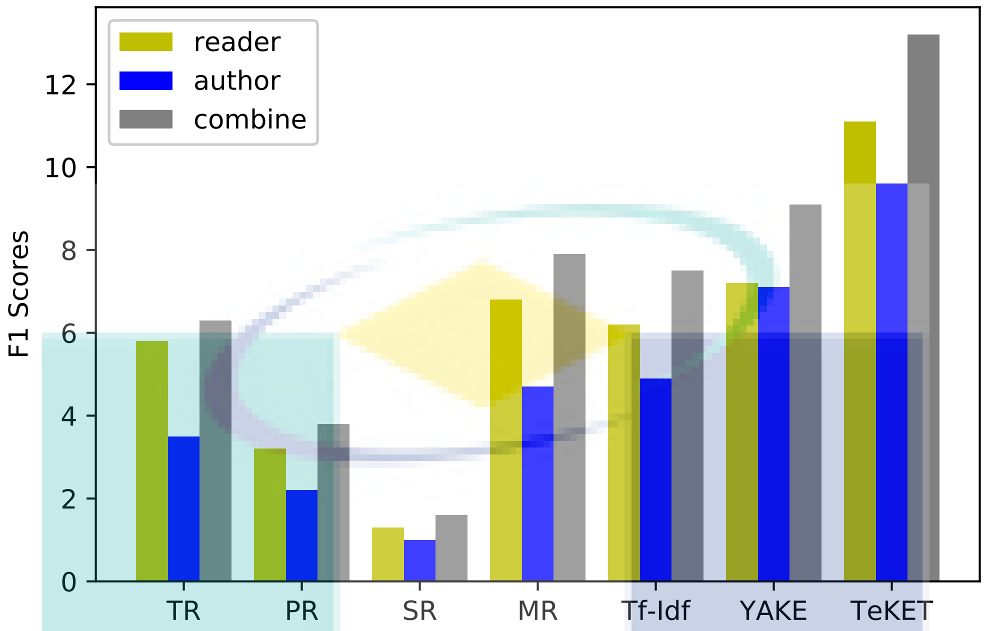


Figure 4.8. F1-Scores of various unsupervised keyphrase extraction techniques for Top-15 keyphrases employed on train dataset

increasing order. However, F1 scores of MR, TF-IDF, and YAKE are comparable for all top-N keyphrases, unlike test data where YAKE outperforms the other two. In any case, our proposed technique overpowers the rest of the compared techniques.

4.4 Summary

This chapter analysis the performance of the proposed algorithm based on three different parameters, namely, precision (ρ), recall (ς), and F1-score (ϕ) for Test and Train data. In the precision (ρ), recall (ς), and F1-score (ϕ) analysis, the proposed algorithm is tested by, Top 5, 10 and 15 keyphrases for reader-assigned, author-assigned and combined keyphrases of Test and Train data, where the proposed algorithom shown an acceptable resilience against each unsupervised machine learning based keyphrase extraction technique. Another advantage of this algorithm is flexibility depends on suitable mamu value, we conduct experiments varying mamu values from 0 to 5, fixing lsaf to 3, and taking test and train datasets of the corpus into consideration(Figure 4.1 and 4.2). Afterwards, the algorithm is compared deeply with the most relavent unsupervised keyphrase extraction algorithm based on different parameters. In most case, the proposed algorithm is found effective than the other related algorithm. At the end, the validity threats of proposed algorithm are discussed briefly.

CHAPTER 5

CONCLUSIONS

5.1 Preamble

This thesis proposes a novel automatic keyphrase extraction algorithm, which is implemented and tested according to the design. Then an in-lab experiment is performed to discover that it works better than other unsupervised keyphrase extraction algorithm and an extensive comparison using SemEval-2010 benchmark dataset conducted to evaluate the usability of it. In the end, contributions, limitations and the future work are discussed, following by a brief summary of the entire research.

5.2 Concluding Remarks

In this thesis, a new unsupervised automatic keyphrase extraction technique, named Tree-based Keyphrase Extraction Technique (TeKET) is proposed, which is domain independent, employs limited statistical knowledge, but no train data are required. It introduces a new variant of binary tree, called KeyPhrase Extraction (KePhEx) tree, for extracting final keyphrases from candidate keyphrases. The proposed tree is formed using a candidate keyphrase and processed with other similar candidate keyphrases of a certain root. In the end, final keyphrases are extracted from the resultant tree employing the mamu value. Afterward, all the final keyphrases are ranked taking TF and μ factors into account where TF is the popularity of that particular keyphrase in a document and μ is considered for realizing the cohesiveness of every word in that keyphrase, and then, sorted. At last, top-*N* keyphrases are selected from the sorted list and returned.

Our proposed technique is compared with other prominent unsupervised keyphrase extraction techniques on a uniform experimental setup. The results are acquired for two datasets, namely test and train, for the SemEval-2010 corpus. According to the acquired results, TeKET outperforms the rest of the compared techniques in terms F1 scores for all considered parameters.

5.3 Contributions

The main contributions of the thesis are described below:

1. A new domain-independent flexible unsupervised keyphrase extraction technique called *TeKET* is introduced in this thesis. A new variant of a binary tree called KePhEx tree is also introduced for extracting final keyphrases from candidate keyphrases.

2. A new ranking technique is introduced in this thesis that can select most relevant top-N keyphrases from a list of final keyphrases. Here, a final keyphrase is scored by TF and μ values where TF identifies the popularity of that particular keyphrase in a document and μ realizes the cohesiveness of every word in that keyphrase.

3. The proposed technique is implemented and tested on a benchmark dataset called SemEval-2010 and also compared with other unsupervised keyphrase extraction techniques (PositionRank, CollabRank (SingleRank), TopicRank, MultipartiteRank, Term Frequency-Inverse Document Frequency, and YAKE).

5.4 Limitations

Although the proposed algorithm is effective, domain independent and not required any train data, still it has some limitations, such as *TeKET* offer huge flexibility and provide good quality keyphrases, but this flexibility does not have a strong influence on the immensely small length data. For the small length data, it works as like as *TF-IDF* because of the less repetition.

5.5 Future works

This work lays the foundation for a flexible unsupervised automatic keyphrase extraction. It also introduced the use of tree in an unsupervised keyphrase extraction algorithm. The next upgradation of this algorithom can be done by integrating *TeKET* in supervised machine learning approach as a feature. A novel supervised approach can be proposed using *TeKET* and incorporate it with the former algorithms, especially for training purposes. This algorithom will fall under the class of supervised learning. Afterwards, *TeKET* can be used to identify meaningful keyphrases with the help of Deep Learning or Hierarchical Learning (Meng et al., 2017), which work on deep semantic meaning basis, where a system is taught by a specific algorithm to recognize any specific items even if they come up with different attributes. Likewise in this case, the keyphrases of every domain will definitely not the same. Therefore, Deep Learning will be a good choice where different set of keyphrase can be fed into the algorithm to train the system to recognize semantic meaning properly.

5.6 Summary

This section concludes the thesis by giving a brief summary of the entire research. The contributions of the thesis are discussed, where the three phases of the proposed algorithom and the comparison of the proposed algorithom with other existing unsupervised keyphrase extraction algorithom are enlightened. Afterwards, the limitations of the proposed algorithom are detailed, such as, text must have a minimum length, and so on. The chapter ends by giving a vision for the future works, where some hints are given; how a the proposed algorithom can be transformed into as a feature of a supervised keyphrase extraction algorithom based on Deep Learning or Hierarchical Learning.

REFERENCES

Adeniyi, D., Wei, Z., & Yongquan, Y. (2016). Automated web usage data mining and recommendation system using k-nearest neighbor (KNN) classification method. *Applied Computing and Informatics*, *12*(1), 90–108.

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge & Data Engineering*(6), 734–749.

Ajmani, S., Ghosh, H., Mallik, A., & Chaudhury, S. (2013). An ontology based personalized garment recommendation system. In *Proceedings of the 2013 ieee/wic/acm international joint conferences on web intelligence (wi) and intelligent agent technologies (iat)-volume 03* (pp. 17–20).

Barker, K., & Cornacchia, N. (2000). Using noun phrase heads to extract document keyphrases. In *Conference of the canadian society for computational studies of intelligence* (pp. 40– 52).

Bayraktar, M., Say, B., & Akman, V. (1998). An analysis of english punctuation: The special case of comma. *International Journal of Corpus Linguistics*, *3*(1), 33–57.

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Bird, S., & Loper, E. (2004). Nltk: the natural language toolkit. In *Proceedings of the acl 2004 on interactive poster and demonstration sessions* (p. 31).

Bobadilla, J., Ortega, F., Hernando, A., & Gutie´rrez, A. (2013). Recommender systems survey. *Knowledge-based systems*, 46, 109–132.

Boudin, F. (2016a, December). pke: an open source python-based keyphrase extraction toolkit. In *Proceedings of coling 2016, the 26th international conference on computational linguistics: System demonstrations* (pp. 69–73). Osaka, Japan. Retrieved from http://aclweb.org/anthology /C16-2015.

Boudin, F. (2016b, December). pke: an open source python-based keyphrase extraction toolkit. *In Proceedings of coling 2016, the 26th international conference on computational linguistics: System demonstrations* (pp. 69–73).

Osaka, Japan: The COLING 2016 Organizing Committee. Retrieved from http://aclweb.org/anthology/C16-2015.

Boudin, F. (2018). Unsupervised keyphrase extraction with multipartite graphs. *arXiv preprintarXiv:1803.08721*.

Bougouin, A., Boudin, F., & Daille, B. (2013). Topicrank: Graph-based topic ranking for keyphrase extraction. In *International joint conference on natural language processing (ijcnlp)* (pp. 543–551).

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, *30*(1-7), 107–117.

Broder, A., Fontoura, M., Josifovski, V., & Riedel, L. (2007). A semantic approach to contextual advertising. In *Proceedings of the 30th annual international acm sigir conference on research and development in information retrieval* (pp. 559–566).

Brown, J. S., & Duguid, P. (1998). Organizing knowledge. *California management review*, *40*(3), 90–111.

Campos, R., Mangaravite, V., Pasquali, A., Jorge, A. M., Nunes, C., & Jatowt, A. (2018a). A text feature based automatic keyword extraction method for single documents. In *European conference on information retrieval* (pp. 684–691).

Campos, R., Mangaravite, V., Pasquali, A., Jorge, A. M., Nunes, C., & Jatowt, A. (2018b). Yake! Collection independent automatic keyword extractor. In *European conference on information retrieval* (pp. 806–810).

Chien, L.-F. (1997). Pat-tree-based keyword extraction for chinese information retrieval. In *Acm sigir forum* (Vol. 31, pp. 50–58).

Chor, B., Gilboa, N., & Naor, M. (1997). *Private information retrieval by keywords*. Citeseer. Das, A. S., Datar, M., Garg, A., & Rajaram, S. (2007). Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on world wide web* (pp. 271–280).

Dashtipour, K., Poria, S., Hussain, A., Cambria, E., Hawalah, A. Y., Gelbukh, A., & Zhou, Q. (2016). Multilingual sentiment analysis: state of the art and independent comparison of techniques. *Cognitive computation*, *8*(4), 757–771.

Dean, J. A., Harik, G. R., & Bucheit, P. (2010, May 11). *Methods and apparatus for serving relevant advertisements*. Google Patents. (US Patent 7,716,161).

El-Beltagy, S. R. (2006). Kp-miner: A simple system for effective keyphrase extraction. In *Innovations in information technology, 2006* (pp. 1–5).

El-Beltagy, S. R., & Rafea, A. (2009a). Kp-miner: A keyphrase extraction system for english and arabic documents. *Information Systems*, *34*(1), 132–144.

El-Beltagy, S. R., & Rafea, A. (2009b). Kp-miner: A keyphrase extraction system for english and arabic documents. *Information Systems*, *34*(1), 132–144.

El-Beltagy, S. R., & Rafea, A. (2010). Kp-miner: Participation in semeval-2. In *Proceedings of the 5th international workshop on semantic evaluation* (pp. 190–193).

Felfernig, A., & Kiener, A. (2005). Knowledge-based interactive selling of financial services with fsadvisor. In *Proceedings of the national conference on artificial intelligence* (Vol. 20, p. 1475).

Florescu, C., & Caragea, C. (2017a). A position-biased pagerank algorithm for keyphrase extraction. In *Aaai* (pp. 4923–4924).

Florescu, C., & Caragea, C. (2017b). Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)* (Vol. 1, pp. 1105–1115).

Fox, C. (1989). A stop list for general text. In *Acm sigir forum* (Vol. 24, pp. 19–21).

Franceschini, F., Maisano, D., & Mastrogiacomo, L. (2016). Empirical analysis and classification of database errors in scopus and web of science. *Journal of Informetrics*, *10*(4), 933–953.

Frank, E., Paynter, G. W., Witten, I. H., Gutwin, C., & Nevill-Manning, C. G. (1999a). Domain-specific keyphrase extraction. In *16th international joint conference on arti- ficial intelligence (ijcai 99)* (Vol. 2, pp. 668–673).

Frank, E., Paynter, G. W., Witten, I. H., Gutwin, C., & Nevill-Manning, C. G. (1999b). Domain-specific keyphrase extraction. In *16th international joint conference on arti- ficial intelligence (ijcai 99)* (Vol. 2, pp. 668–673).

Freitag, D. (2000). Machine learning for information extraction in informal domains. *Machine learning*, *39*(2-3), 169–202.

Giguere, E. (2005). *Make easy money with google: using the adsense advertising program*. Peachpit Press.

Girardi, R., & Marinho, L. B. (2007). A domain model of web recommender systems based on usage mining and collaborative filtering. *Requirements Engineering*, *12*(1), 23–40.

GitHub. (n.d.). https://github.com/. (Accessed: 20108-10-11)

Goyvaerts, J., & Levithan, S. (2012). *Regular expressions cookbook*. O'reilly.

Han, Y. (2004). Digital content management: the search for a content management system. *Library Hi Tech*, *22*(4), 355–365.

Hariharan, R., Hore, B., Li, C., & Mehrotra, S. (2007). Processing spatial-keyword (sk) queries in geographic information retrieval (gir) systems. In *Scientific and statistical database management, 2007. ssbdm'07. 19th international conference on* (pp. 16–16).

Hasan, K. S., & Ng, V. (2014). Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 1: Long papers)* (Vol. 1, pp. 1262–1273).

Herrera, J. P., & Pury, P. A. (2008). Statistical keyword detection in literary corpora. *The European Physical Journal B*, *63*(1), 135–146.

Holzinger, A. (2017). *Introduction to machine learning & knowledge extraction (make)*. Multidisciplinary Digital Publishing Institute.

Huang, F., Zhang, Y., & Vogel, S. (2005). Mining key phrase translations from web corpora. In *Proceedings of the conference on human language technology and empirical methods in natural language processing* (pp. 483–490).

Hulth, A. (2003a). Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on empirical methods in natural language pro- cessing* (pp. 216–223).

Hulth, A. (2003b). Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on empirical methods in natural language pro- cessing* (pp. 216–223).

Huynh, T., & Hoang, K. (2012). Modeling collaborative knowledge of publishing activities for research recommendation. In *International conference on computational collective intelligence* (pp. 41–50).

Jean-Louis, L., Zouaq, A., Gagnon, M., & Ensan, F. (2014). An assessment of online semantic annotators for the keyword extraction task. In *Pacific rim international conference on artificial intelligence* (pp. 548–560).

Jo, T., Lee, M., & Gatton, T. M. (2006). Keyword extraction from documents using a neural network model. In *Hybrid information technology, 2006. ichit'06. international confer- ence on* (Vol. 2, pp. 194–197).

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning* (pp. 137–142).

Judd, D. T., Brewster, J. A., Melia, P. M., & Lilly, D. J. (2006, March 21). *Content management and transformation system for digital content.* Google Patents. (US Patent 7,016,963)

Kantrowitz, M., Mohit, B., & Mittal, V. (2000). Stemming and its effects on tfidf ranking (poster session). In *Proceedings of the 23rd annual international acm sigir conference on research and development in information retrieval* (pp. 357–359).

Kara, W. B. A., & Griba, N. (2013). Information retrieval with porter stemmer: a new version for english. In Advances in computational science, engineering and information technology (pp. 243–254). Springer.

Kim, S. N., Baldwin, T., & Kan, M.-Y. (2010). Evaluating n-gram based evaluation metrics for automatic keyphrase extraction. In *Proceedings of the 23rd international conference on computational linguistics* (pp. 572–580).

Kim, S. N., Medelyan, O., Kan, M.-Y., & Baldwin, T. (2010). Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th international workshop on semantic evaluation* (pp. 21–26).

Kim, S. N., Medelyan, O., Kan, M.-Y., & Baldwin, T. (2013). Automatic keyphrase extraction from scientific articles. *Language resources and evaluation*, *47*(3), 723–742.

Kononenko, I. (2001). Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in medicine*, *23*(1), 89–109.

Kosala, R., & Blockeel, H. (2000). Web mining research: A survey. *ACM Sigkdd Explorations Newsletter*, *2*(1), 1–15.

Kotler, P., & Roberto, E. L. (1989). *Social marketing. strategies for changing public behavior*. Free Press.

Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, *160*, 3–24.

Kumar, N., & Srinathan, K. (2008). Automatic keyphrase extraction from scientific documents using n-gram filtration technique. In *Proceedings of the eighth acm symposium on document engineering* (pp. 199–208).

Lafferty, J., McCallum, A., & Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Langville, A. N., & Meyer, C. D. (2011). *Google's pagerank and beyond: The science of search engine rankings*. Princeton University Press.

Lau, R. Y., Song, D., Li, Y., Cheung, T. C., & Hao, J.-X. (2009). Toward a fuzzy domain ontology extraction method for adaptive e-learning. *IEEE transactions on knowledge and data engineering*, *21*(6), 800–813.

Laursen, A., Olkin, J., & Porter, M. (1994). Oracle media server: providing consumer based interactive access to multimedia data. In *Acm sigmod record* (Vol. 23, pp. 470–477).

Lawrence, S., Giles, C. L., & Bollacker, K. (1999). Digital libraries and autonomous citation indexing. *Computer*, *32*(6), 67–71.

Litvak, M., & Last, M. (2008). Graph-based keyword extraction for single document summarization. In *Proceedings of the workshop on multisource multilingual information extraction and summarization* (pp. 17–24).

Liu, F., Pennell, D., Liu, F., & Liu, Y. (2009). Unsupervised approaches for automatic keyword extraction using meeting transcripts. *In Proceedings of human language technologies: The 2009 annual conference of the north american chapter of the association for computational linguistics* (pp. 620–628).

Loper, E., & Bird, S. (2002). Nltk: The natural language toolkit. In *Proceedings of the acl-02 workshop on effective tools and methodologies for teaching natural language processing and computational linguistics-volume 1* (pp. 63–70).

Manevitz, L. M., & Yousef, M. (2001). One-class svms for document classification. *Journal of machine Learning research*, *2*(Dec), 139–154.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., & McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd* *annual meeting of the association for computational linguistics: system demonstrations* (pp. 55–60).

Massa, P., & Avesani, P. (2007). Trust-aware recommender systems. In *Proceedings of the 2007 acm conference on recommender systems* (pp. 17–24).

Matsuo, Y., & Ishizuka, M. (2004). Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, *13*(01), 157–169.

McCallum, A., Nigam, K., et al. (1998). A comparison of event models for naive bayes text classification. In *Aaai-98 workshop on learning for text categorization* (Vol. 752, pp. 41–48).

Meng, R., Zhao, S., Han, S., He, D., Brusilovsky, P., & Chi, Y. (2017). Deep keyphrase generation. *arXiv preprint arXiv:1704.06879*.

Merrouni, Z. A., Frikh, B., & Ouhbi, B. (2016). Automatic keyphrase extraction: An overviewof the state of the art. In *Information science and technology (cist), 2016 4th ieee international colloquium on* (pp. 306–313).

Mihalcea, R. (2004). Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the acl 2004 on interactive poster and demonstration sessions* (p. 20).

Mihalcea, R., & Tarau, P. (2004). Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

Mihalcea, R., Tarau, P., & Figa, E. (2004). Pagerank on semantic networks, with application to word sense disambiguation. In *Proceedings of the 20th international conference on computational linguistics* (p. 1126).

Miller, D., & Friesen, P. H. (1986). Porter's (1980) generic strategies and performance: an empirical examination with american data: part i: testing porter. *Organization studies*, *7*(1), 37–55.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, *38*(11), 39–41.

Miller, J. C., Rae, G., Schaefer, F., Ward, L. A., LoFaro, T., & Farahat, A. (2001). Modifications of kleinberg's hits algorithm using matrix exponentiation and web log records. In *Proceedings of the 24th annual international acm sigir conference on research and development in information retrieval* (pp. 444–445).

Murtagh, F., & Legendre, P. (2014). Ward's Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward's Criterion? *Journal of classification*, *31*(3), 274– 295.

Narasimhamurthy, A. (2005). Theoretical bounds of majority voting performance for a binary classification problem. *IEEE Transactions on Pattern Analysis and Machine Intelli- gence*, *27*(12), 1988–1995.

Nguyen, T. D., & Kan, M.-Y. (2007). Keyphrase extraction in scientific publications. In *International conference on asian digital libraries* (pp. 317–326).

Noun phrase [Computer software manual]. ((accessed May 3, 2018)). Retrieved from https://en.oxforddictionaries.com/definition/us/noun phrase/

Ohsawa, Y., Benson, N. E., & Yachida, M. (1998). Keygraph: Automatic indexing by co- occurrence graph based on building construction metaphor. In *Research and technology advances in digital libraries, 1998. adl 98. proceedings. ieee international forum on* (pp. 12–18).

Ono, T., Hishigaki, H., Tanigami, A., & Takagi, T. (2001). Automated extraction of information on protein–protein interactions from the biological literature. *Bioinformatics*, *17*(2), 155–161.

Page, L. (2001, September 4). *Method for node ranking in a linked database.* Google Patents. (US Patent 6,285,999)

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). *The pagerank citation ranking: Bringing order to the web.* (Tech. Rep.). Stanford InfoLab.

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the acl-02 conference on empirical methods in natural language processing-volume 10* (pp. 79–86).

Pazzani, M. J. (1999). A framework for collaborative, content-based and demographic filter- ing. *Artificial intelligence review*, *13*(5-6), 393–408.

Popova, S., & Danilova, V. (2014). Keyphrase extraction. abstracts instead of full papers. In *25th international workshop on database and expert systems applications* (pp. 241– 245).

Pudota, N., Dattolo, A., Baruzzo, A., Ferrara, F., & Tasso, C. (2010). Automatic keyphrase extraction and ontology mining for content-based tag recommendation. *International Journal of Intelligent Systems*, *25*(12), 1158–1186.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, *1*(1), 81–106.

Rabby, G., & Azad, S. (2018). *Teket.* https://github.com/corei5/TeKET. GitHub.

Rabby, G., Azad, S., Mahmud, M., & Zamli, R. M. M., Kamal Z. (2018). A flexible keyphrase extraction technique for academic literature. In *Procedia compute science* (Vol. 135, pp. 653–663). Elsevier.

Regular expression howto. (n.d.). Retrieved from https://docs.python.org/3 /howto/regex.html.

Reilly, R. G., & Sharkey, N. (2016). *Connectionist approaches to natural language processing*. Routledge.

Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to recommender systems handbook. In *Recommender systems handbook* (pp. 1–35). Springer.

Robertson, S. (2004). Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation*, *60*(5), 503–520.

Robertson, S. E. (1977). The probability ranking principle in ir. *Journal of documentation*, *33*(4), 294–304.

Rowley, J., & Hartley, R. (2017a). *Organizing knowledge: an introduction to managing access to information*. Routledge.

Rowley, J., & Hartley, R. (2017b). *Organizing knowledge: an introduction to managing access to information*. Routledge.

Salton, G., & Buckley, C. (1988a). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, *24*(5), 513-523. Retrieved from http://www.sciencedirect.com/science/article/pii/ 0306457388900210 doi: https://doi.org/10.1016/0306-4573(88)90021-0

Salton, G., & Buckley, C. (1988b). Term-weighting approaches in automatic text retrieval. *Information processing & management*, *24*(5), 513–523.

Salton, G., Wong, A., & Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, *18*(11), 613–620.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on world wide web* (pp. 285–295).

Scheve, T. (2018). *Why does having too many options make it harder to choose?* Retrieved from https://science.howstuffworks.com/life/ choose-options.htm

Seuring, S., & Gold, S. (2012). Conducting content-analysis based literature reviews in supply chain management. *Supply Chain Management: An International Journal*, *17*(5), 544– 555.

Siddiqi, S., & Sharan, A. (2015). Keyword and keyphrase extraction techniques: a literature review. *International Journal of Computer Applications*, *109*(2).

Steinbach, M., Karypis, G., & Kumar, V. (2000). A comparison of document clustering techniques. In *In kdd workshop on text mining.*

Sterckx, L., Demeester, T., Deleu, J., & Develder, C. (2015). Topical word importance for fast keyphrase extraction. In *Proceedings of the 24th international conference on world wide web* (pp. 121–122).

Sterckx, L., Demeester, T., Deleu, J., & Develder, C. (2018). Creation and evaluation of large keyphrase extraction collections with multiple opinions. *Lang Resources & Evaluation*, *52*, 503–532.

Sugiyama, K., & Kan, M.-Y. (2017). *Scholarly paper recommendation datasets.* Retrieved from http://www.comp.nus.edu.sg/~sugiyama/ SchPaperRecData.html

Thomas, J. R., Bharti, S. K., & Babu, K. S. (2016). Automatic keyword extraction for text summarization in e-newspapers. In *Proceedings of the international conference on in- formatics and analytics* (p. 86).

Tixier, A., Malliaros, F., & Vazirgiannis, M. (2016). A graph degeneracy-based approach to keyword extraction. In *Proceedings of the 2016 conference on empirical methods in natural language processing* (pp. 1860–1870).

Tomokiyo, T., & Hurst, M. (2003). A language model approach to keyphrase extraction. In *Proceedings of the acl 2003 workshop on multiword expressions: analysis, acquisition and treatment-volume 18* (pp. 33–40).

Tumer, D., Shah, M. A., & Bitirim, Y. (2009). An empirical evaluation on semantic search performance of keyword-based and semantic search engines: Google, yahoo, msn and hakia. *In Internet monitoring and protection, 2009. icimp'09. fourth international conference on* (pp. 51–55).

Turney, P. D. (2000). Learning algorithms for keyphrase extraction. *Information retrieval*, *2*(4), 303–336.

Vallez, M., Pedraza-Jimenez, R., Codina, L., Blanco, S., & Rovira, C. (2015). A semi-automatic indexing system based on embedded information in html documents. *Library Hi Tech*, *33*(2), 195–210.

Vencovsky, F., Lucas, B., Mahr, D., & Lemmink, J. (2017). Comparison of text mining techniques for service aspect extraction. In *Ecsm 2017 4th european conference on social media* (p. 297).

Wan, X., & Xiao, J. (2008). Collabrank: towards a collaborative approach to single-document keyphrase extraction. In *Proceedings of the 22nd international conference on computational linguistics-volume 1* (pp. 969–976).

Wang, H., Xu, F., Hu, X., & Ohsawa, Y. (2013). Ideagraph: a graph-based algorithm of mining latent information for human cognition. In *Systems, man, and cybernetics (smc), 2013 ieee international conference on* (pp. 952–957).

Wang, J., Liu, J., & Wang, C. (2007). Keyword extraction based on pagerank. *In Pacific-asia conference on knowledge discovery and data mining* (pp. 857–864).

Wang, L., Zeng, X., Koehl, L., & Chen, Y. (2015). Intelligent fashion recommender system: Fuzzy logic in personalized garment design. *IEEE Trans. Human-Machine Systems*, *45*(1), 95–109.

Wang, R., Liu, W., & Mcdonald, C. (2014). How preprocessing affects unsupervised keyphrase extraction. In *International conference on intelligent text processing and computational linguistics* (pp. 163–176).

Willett, P. (2006). The porter stemming algorithm: then and now. *Program*, *40*(3), 219–223. Witten, I. H., Paynter, G. W., Frank, E., Gutwin, C., & Nevill-Manning, C. G. (2005). Kea: Practical automated keyphrase extraction. In *Design and usability of digital libraries: Case studies in the asia pacific* (pp. 129–152). IGI Global.

Wolf, N., Zhu, Z., Semret, N., & Baskin, J. (2013, August 20). *Providing product recommendations through keyword extraction from negative reviews.* Google Patents. (US Patent 8,515,828)

Wu, Z., Zhu, H., Li, G., Cui, Z., Huang, H., Li, J., Xu, G. (2017). An efficient wikipedia semantic matching approach to text document classification. *Information Sciences, 393*, 15–28.

Xu, C., Wu, Y., & Liu, Z. (2017). Multimodal fusion with global and local features for text classification. In *International conference on neural information processing* (pp. 124– 134).

Xu, W., Liu, X., & Gong, Y. (2003). Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international acm sigir conference on research and development in informaion retrieval* (pp. 267–273).

Yager, R. R. (2003). Fuzzy logic methods in recommender systems. *Fuzzy Sets and Systems, 136*(2), 133–149.

Yang, C. C. (1997). Fuzzy bayesian inference. In *Systems, man, and cybernetics, 1997. computational cybernetics and simulation., 1997 ieee international conference on* (Vol. 3, pp. 2707–2712).

Yoo, S.-C., & Eastin, M. S. (2017). Contextual advertising in games: Impacts of game context on a players memory and evaluation of brands in video games. *Journal of Marketing Communications*, *23*(6), 614–631.

Yu, Y., & Ng, V. (2018). Wikirank: Improving keyphrase extraction based on background knowledge. *arXiv preprint arXiv:1803.09000*.

Yujian, L., & Bo, L. (2007). A normalized levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, *29*(6), 1091–1095.

Zha, H. (2002). Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In *Proceedings of the 25th annual international acm sigir conference on research and development in information retrieval* (pp. 113–120).

Zhai, C., & Lafferty, J. (2017). A study of smoothing methods for language models applied to ad hoc information retrieval. In *Acm sigir forum* (Vol. 51, pp. 268–276).

Zhang, K., Xu, H., Tang, J., & Li, J. (2006). Keyword extraction using support vector machine. In *International conference on web-age information management* (pp. 85–96).

Zhao, W. X., Jiang, J., He, J., Song, Y., Achananuparp, P., Lim, E.-P., & Li, X. (2011). Topical keyphrase extraction from twitter. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1* (pp. 379–388).

APPENDIX A

Research Papers

1. **Rabby, G.**, Azad, S., Mahmud, M., Zamli, K.Z. and Rahman, M.M., (2018). A Flexible Keyphrase Extraction Technique for Academic Literature. Procedia Computer Science, 135, pp.553-563.

2. Yong, T.F., Azad, S., Rahman, M.M., Zamli, K.Z. and **Rabby, G.**, 2018. A Highly Accurate PDF-To-Text Conversion System for Academic Papers Using Natural Language Processing Approach. Advanced Science Letters, 24(10), pp.7844-7849.

3. **Rabby, G.**, Azad, S., Mahmud, M., Zamli, K. Z., and Rahman, M. M. (n.d.). TeKET: A Tree-based Keyphrase Extraction Technique for Decision Support Systems. Applied Soft Computing(Under Review).

Awards

1. GOLD in CITREX-2019, UMP, MALAYSIA, with project **”TeKET: A Tree-based Keyphrase Extraction Technique.”**

2. SILVER in CITREX-2018, UMP, MALAYSIA, with project **”A Highly Accurate PDF- to-Text Conversion System for Academic Papers using Natural Language Process- ing Approach.”**

APPENDIX B

Sample Experiment Data

Scalable Grid Service Discovery Based on UDDI* * Authors are listed in alphabetical order. Sujata Banerjee$, Sujoy Basu$, Shishir Garg , Sukesh Garg , Sung-Ju Lee$, Pramila Mullan , Puneet Sharma$ $ HP Labs 1501 Page Mill Road Palo Alto, CA, 94304 USA +1- 650-857-2137 sujata.banerjee, sujoy.basu, sungju.lee, puneet.sharma@hp.com France Telecom R&D Division 801 Gateway Blvd, # 500 South San Francisco, CA, 94080 USA +1 650 -875- 1500 shishir.garg, sukesh.garg, pramila.mullan@francetelecom.com ABSTRACT Efficient dis- covery of grid services is essential for the success of grid computing. The standardization of grids based on web services has resulted in the need for scalable web service discovery mech- anisms to be deployed in grids Even though UDDI has been the de facto industry standard for web-services discovery, imposed requirements of tight-replication among registries and lack of autonomous control has severely hindered its widespread deployment and usage. With the advent of grid computing the scalability issue of UDDI will become a roadblock that will prevent its deployment in grids. In this paper we present our distributed web-service dis- covery architecture, called DUDE (Distributed UDDI Deployment Engine). DUDE leverages DHT (Distributed Hash Tables) as a rendezvous mechanism between multiple UDDI registries. DUDE enables consumers to query multiple registries, still at the same time allowing orga- nizations to have autonomous control over their registries.. Based on preliminary prototype on PlanetLab, we believe that DUDE architecture can support effective distribution of UDDI registries thereby making UDDI more robust and also addressing its scaling issues. Further- more, The DUDE architecture for scalable distribution can be applied beyond UDDI to any Grid Service Discovery mechanism. Categories and Subject Descriptors C2.4 [Distributed Systems] General Terms Design, Experimentation, Standardization. 1. INTRODUCTION Efficient discovery of grid services is essential for the success of grid computing..........

Sample Candidate Phrases

’scalabl grid servic discoveri base’, ’list’, ’alphabet order’, ’sujata banerje’, ’shishir garg’, ’sukesh garg’, ’pramila mullan’, ’puneet’, ’hp lab’, ’page mill road palo alto’, ’d divi’, ’gateway blvd’, ’san francisco’, ’grid servic’, ’success’, ’grid comput’, ’standard’, ’grid base’, ’web servic ha result’, ’need’, ’scalabl web servic discoveri mechan’, ’uddi ha’, ’industri stan- dard’, ’web-serv discoveri’, ’impo requir’, ’registri’, ’lack’, ’autonom control ha sever hinder’, ’widespread deploy’, ’usag’, ’advent’, ’scalabl issu’, ’uddi’, ’roadblock’, ’grid’, ’thi paper’, ’distribut web-serv discoveri architectur’, ’dude’, ’uddi deploy engin’, ’dude leverag dht’, ’distribut hash tabl’, ’rendezv mechan’, ’multipl uddi registri’, ’dude’, ’consum’, ’multipl registri’, ’same time’, ’organ’, ’autonom’, ’registries.. base’, ’preliminari prototyp’, ’planetlab’, ’architectur’, ’effect distribut’, ’uddi registri therebi’, ’scale issu’, ’dude architectur', 'scalabl distribut', 'ser- vic discoveri mechan', 'categori', 'subject descriptor c2.4 [distribut system] gener term de- sign', 'experiment', 'standard', 'introduct effici discoveri', 'grid servic', 'success', 'grid com- put', 'standard', 'grid base', 'web servic ha result', 'need', 'scalabl web servic permiss', 'digit', 'hard copi', 'part', 'thi work', 'person', 'classroom use', 'fee provid', 'copi', 'profit', 'advan- tag', 'bear thi', 'full citat', 'first page', 'server', 'list', 'prior specif permiss', 'fee', 'mgc', '28- decemb', 'grenobl', 'franc discoveri mechan', 'grid', 'grid discoveri servic', 'abil', 'resourc', 'servic', 'grid', 'abil', 'inform', 'addit', 'threshold trap', 'specif chang', 'exist condit', 'state', 'data', 'soft state', 'recent inform', 'inform gather need', 'system', 'purpo', 'grid', 'summari inform', 'fundament problem', 'need', 'huge amount', 'data', 'multipl sourc', 'web servic com- mun ha', 'need', 'servic discoveri', 'befor grid', 'industri standard call uddi', 'uddi ha', 'industri standard', 'web-serv discoveri', 'impo requir', 'registri', 'lack', 'autonom control', 'other thing ha sever hinder', 'widespread deploy', 'usag [', 'advent', 'scalabl issu', 'uddi', 'roadblock', 'grid', 'thi paper', 'scalabl issu', 'way', 'multipl registri', 'distribut web servic discoveri archi- tectur', 'distribut uddi function', 'multipl way', 'corba', 'dce', 'thi paper', 'distribut hash tabl', 'dht', 'technolog', 'scalabl distribut', 'servic discoveri architectur', 'dht', 'p2p', 'distribut sys- tem', 'structur overlay', 'more effici rout', 'underli network', 'crucial design choic', 'factor', 'first motiv factor', 'inher simplic', 'dht provid', 'top', 'dht', 'abstract', 'distribut applic', 'other distribut', 'provid more function', 'higher overhead', 'complex',..........

Sample Keyphrases

('uddi registri ', 26322), ('registri ', 25600), ('servic ', 11236), ('proxi registri ', 10908), ('dht ', 9604), ('uddi ', 7396), ('local registri ', 6916), ('servic name ', 5040), ('dht node ', 4026), ('servic discoveri ', 3360), ('servic inform ', 2196), ('proxi ', 1764), ('queri ', 1296), ('thi ', 1156), ('key ', 1156), ('hash tabl ', 832), ('thi work ', 768), ('search ', 676), ('grid ', 576), ('node ', 576), ('client ', 484), ('local ', 484), ('scalabl ', 400), ('hash ', 400), ('name ', 400), ('comput ', 256), ('architectur ', 256), ('inform ', 256), ('http ', 256), ('discoveri ', 196), ('work ', 196), ('top ', 196), ('section ', 196), ('proceed ', 196), ('tabl ', 144), ('abil ', 144), ('valu ', 144), ('rout ', 144), ('prefix ', 144)

Sample Match with Gold Standard Keyphrases

Match Keyphrase:

uddi registry

dht

Match in Gold Standard Keyphrases keyphrases: 2

Top *n* keyphrase: 5

Lenth of the Gold Standard Keyphrases: 19

Match Keyphrase:

uddi registri

dht

uddi

Match in Gold Standard Keyphrases keyphrases: 3

Top *n* keyphrase: 10

Lenth of the Gold Standard Keyphrases: 19

Match Keyphrase:

uddi registri

dht

uddi

queri

Match in Gold Standard Keyphrases keyphrases: 4

Top *n* keyphrase: 15

Lenth of the Gold Standard Keyphrases: 19