Table of Contents

Chapter 1 Introduction

As the Fourth Industrial Revolution unfolds, a succession of emerging digital and artificial intelligence technologies give impetus to new forms of data generation and analysis. The recent breathtaking storm set off by the language model ChatGPT is undoubtedly an impressive one, ushering people into a new world of AI empowered content generation and interaction. Particularly, the penetration of emerging digital and artificial intelligence technologies into the scientific domain has a transcendent impact on methodological paradigms of traditional research, fostering the rising development of intersectional studies of language and computer science. The current thesis exerts itself to such kind of arduous effort, i.e. the incorporation of linguistic knowledge into the design of keyword extraction algorithm, grounded in the specific context of scholarly discourse.

This chapter is dedicated to an introduction to the background and significance of keyword extraction in the scientific domain, the objectives the thesis seeks to achieve, and the major contributions of the paper.

1.1 Research Background and Significance

With the leaps and bounds of information technology, the human craving for content condensation and data organization becomes increasingly intense. Whether it be in natural science or social science, a kind of ‘‘data deluge’’ is exceeding the capacity of current research methods and tools. Large quantities of scholarly documents that are important for the sharing and dissemination of scientific discoveries, including “journal and conference papers, dissertations and theses, books, technical reports and working papers” (Khabsa & Giles, 2014, p.1), become just a click of a button away. This brings convenience and diversity to scientific research, but also causes unprecedented difficulties in handling such rapid proliferation of literature. Given the massive amounts of information, researchers are troubled with searching and sorting relevant papers, especially for those working on systematic reviews and surveys.

Thus, people urgently hope to summarize or compress information of a research paper through several index terms, which are called **keywords** in practice, as shown in Table 1.1 as an example. While large segments of text are too long to read and easy to forget, these keywords can present the main content of the document in a highly condensed and concise way. In fact, a large part of the important ideas and knowledge in scholarly documents are conveyed through these abstractive terms or phrases. To recognize the keywords and capture their underlying meaning is one of the keys to acquiring information from the scientific literature.

Title: **Token frequency** as a determinant of **morphological change**
This paper demonstrates that **morphological change** tends to involve the replacement of low **frequency** forms in inflectional paradigms by innovative forms based on high **frequency** forms, using **Greek** data involving the diachronic reorganisation of verbal inflection classes. A computational procedure is outlined for generating a possibility space of **morphological changes** which can be represented as **analogical** proportions, on the basis of synchronic paradigms in ancient **Greek**. I then show how supplementing **analogical** proportions with **token frequency** information can help to predict whether a hypothetical change actually took place in the language's subsequent development. Because of the crucial role of inflected surface forms serving as **analogical** bases in this model, I argue that the results support theories in which inflected forms can be stored whole in the lexicon. Keywords: **analogy**, **Greek**, **morphological change**, **morphology**, **token frequency**

Table 1.1: The title, abstract and the author-input keywords of a research article by Rendle et al. (2010) from the *Journal of Linguistics*. Bold phrases marked with different colors represent (or part of) the gold-standard keywords for the document.

Traditionally, keywords are manually assigned by authors themselves or by editors. However, only a small portion of research papers have been designated keywords (Hulth, 2003). Taking the journals related to the field of linguistics as an example, four out of seven top ranking journals do not have human-assigned and publicly available keywords. This reveals the common absence of keywords in academic papers, leaving a huge quantity of documents with missing keywords in need of efficient processing. Compared to the time-consuming and laborious work of human expert annotation, machines are more rapid and robust when determining keywords, increasing the speed and quantity by exponential order. This makes the automation of such process

extremely valuable and demanding.

Journal	Impact Factor	Publisher	Keywords
Transactions of the Association for Computational Linguistics	9.194	MIT Press Direct	No
The Modern Language Journal	7.500	Wiley Online Library	Yes
Language Teaching	5.327	Cambridge University Press	No
Language Learning	5.240	Wiley Online Library	Yes
Journal of Memory and Language	4.521	ScienceDirect	Yes
Applied Linguistics	4.155	Oxford Academic	No
Research on Language and Social Interaction	3.077	Taylor and Francis	No

Table 1.2: Top journals of linguistics ranked with impact factor. Orange shading highlights the absence of human-assigned keywords for papers in the journal.

Responding to this mounting need, a host of **Automatic Keyword Extraction** (AKE) techniques spring up, which can give a succinct summary of a document in a set of representative keywords without the need for manual annotation. People marvel at such a wide variety of application scenarios these techniques can potentially make in the context of scientific literature. Facilitated by AKE, information retrieval such as categorization/clustering, indexing, searching, comparison and recommendation of large-scale scientific documents become possible and accelerated; the visualization and analysis of patterns and trends dispersed in these documents can also be eased, helping people to find emerging topics and areas of interest. This will greatly benefit a large number of scientific researchers.

However, the reality of keyword extraction techniques does not live up to our expectation. Academic discourse is known for its formality and precision, and requires the keywords to be syntagmatic linguistic units that are representative of document content and domain-specific concepts. But the quality of extracted keywords rarely meets these requirements, which are mixed with meaningless, broken or generic pieces of word sequences that replicate each other. For example, the phrases extracted by one of the most popular AKE algorithms YAKE! (Campos et al., 2020) are sometimes grammatically improper and incomplete, e.g. “Greek data involving”; and appear in

various different forms that overlap with each other, e.g. “morphological change” and “morphological”. Also, there appear some useless phrases like “forms” and “change”.

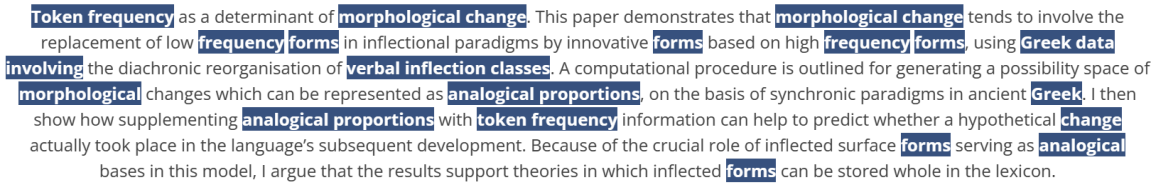

Figure 1.1 YAKE!’s keyword extraction result using the same text by Rendle et al. (2010), computed using the online tool http://yake.inesctec.pt/

Apart from YAKE!, most existing methods are not suitable for the specific context of scholarly documents. They focus on optimizing the selection of each individual keyword from an engineering perspective, ignoring the basic linguistic characteristics of keywords. To fill this gap and meet the requirement of high-quality keyword extraction, the author proposes to devise an effective method of extracting scientific keywords from the angle of looking at their innate features. Any set of keywords that show these features are considered to be able to capture the essence of a scholarly document in a satisfactory way.

1.2 Characteristics of Keywords and Problems of Keyword Extraction

Keywords, or terms in the scholarly context, can be single, containing one word only; or complex, containing multiple words. Sag et al. (2002) states that a large part of the terminology is made up of complex phraseological units or multi-word expressions that have “idiosyncratic interpretations that cross word boundaries” (p.2). Kageura & Umino (1996) points out two important conditions of terms, i.e. “unithood”, which is the degree of strength or stability of syntagmatic combinations or collocations; and “termhood”, which is the degree that a linguistic unit is related to or represents domain-specific concepts.

The thesis summarizes the four major and typical characteristics of keywords in scholarly documents as follows:

Characteristics	Explanation

Termhood	Keywords are stable, uninterrupted and holistic syntagmatic combinations that represent domain-specific concepts.
Distribution	Keyword tend to co-occur with each other and appear in important positions in the text, e.g. in the title.
Informativeness	Keywords are highly descriptive and can impart a document's central idea.
Diversity	Keywords are dissimilar to each other and variegated in topics. More diverse keywords combine to cover meaningful critical points of the text to the fullest.

Table 1.2: The four major characteristics of keywords

The above characteristics form a set of concrete criteria for high-quality scientific keyword extraction. Although most current unsupervised AKE methods are attentive to one or more of the above requirements, they fail to fulfill the other equally important ones (like the above case of YAKE!), giving rise to a series of problems:

The problem of brokenness. Brokenness occurs when the method outputs a grammatically incomplete phrase that carries no exact meaning, e.g. "methodology to track" "study use".

The reliance on frequency. This happens when a frequency-based method mistakenly identifies a common word or stop word like "forms" "this" due to its high frequency, or leaves out a keyword due to its rare presence in the text.

The problem of generality. This happens when the extracted keywords are overly broad in meaning. For example, words like "experiment" and "type" may be relevant but not specific enough to accurately reflect key information in the document.

The problem of redundancy. Redundancy occurs when the method generates duplicated or overlapping keywords that are semantically equivalent, including synonyms (e.g. "benchmark" and "baseline") and phrases nested in another such as "natural language" and "natural language processing".

1.4 Objective and Contributions

How to overcome the above problems and *properly model the four features of keywords to make a standardized keyword set?* To answer this question, the author

proposes a new unsupervised keyword extraction method called **Structural and Semantic Ranking (SSRank)** that combine both structural and semantic information from the scholarly document to fulfill these features in an integrated way.

For structural information, the SSRank algorithm constructs a sophisticated noun phrase detection framework based on POS tagging and dependency parsing to preserve the grammatical integrity and soundness of candidate phrases, preserving the feature of *termhood*. Then, it models the *distribution* of words in an undirected and edge-weighted textual graph, and iteratively computes their importance scores by utilizing word co-occurrence relationships and positional information. Finally, a nonlinear length formula inspired by Zipf's law is applied to pool the component word scores of each phrase to obtain the phrase importance.

For semantic information, SSRank uses semantic distance to quantify the *diversity* of candidate phrases, with the assumption that a set of phrases are more likely to cover more topics if they are semantically remote, and more likely to be redundant if they are semantically close. Specifically, the author proposes two methods: (1) clustering candidates using the HAC algorithm according to the modified edit distance, and then ranking the clusters according to the average inner-group semantic distance; (2) clustering candidates using the K-means algorithm based on the averaged Word2Vec embedding, and then sorting clusters according to the semantic similarity between the phrases and the document. Finally, SSRank integrates these two kinds of information through a method called keyword chaining, which constructs a chain of top candidate phrases from the leading clusters which form the optimal list of keywords.

The major contributions of the thesis are as follows:

(1) The Contribution of Structural and Semantic Parallel Architecture

The main innovation of this article is the design of the SSRank algorithm architecture. The overall design revolves around two cornerstones in the construction of language: syntax and semantics, with the first involving the form, and the latter on content. For form, SSRank preserves the tidy structure inside phrase and models its positional and distributional importance in the whole constitution of the text. For

content, SSRank groups candidates into semantic clusters to determine the best set of keywords that epitomize the essence of meaning with minimal redundancy.

This parallel architecture of form and meaning in SSRank conforms to the formation of language. Also, it simulates human behavior in selecting keywords: first identifying a group of important candidate phrases, and then removing duplicated content. The design ensures the superior performance of SSRank in keyword extraction tasks.

In concrete implementation, the paper uses undirected weighted graph and PageRank algorithm to model structural information, and designs four different distance metrics for semantic clustering. However, in fact, any algorithm based on statistics or graph theory can be used for structural modeling, and any clustering or distance measurement algorithm can be used for semantic modeling. The concept of design is universal, and its internal components can be variegated in various ways.

(2) The Meticulous Design of Algorithm Components

For details in the design of algorithmic components, in the candidate formation process, the author proposed two meticulously designed noun phrase extractors, which achieves a recall score as high as 0.919, meaning that it can generate noun phrases with over 90% accuracy. The structural scoring component is highlighted with the innovative construction of a nonlinear length formula based on Zipf's Law to achieve integration of phrase importance scores. For the semantic component, four semantic distance metrics are proposed, among which pseudo unique thermal coding (POE) cosine distance is a pure invention that that has never been proposed or used in any previous works and can be potentially used for document similarity calculation. These metrics can potentially make a wide variety of applications related to information retrieval and text analysis beyond the AKE task. Finally, the keyword chaining process is proposed to integrate structural and semantic information. These designs make SSRank flexible in the length and number of extracted keywords, and good at generating grammatically complete and semantically diverse keywords.

(3) The Construction of the JML Dataset and Evaluation Metric PM@F

In terms of experimental design, the author constructed a linguistic journal dataset JML and propose a metric PM@F based on partial matching. JML is one of the few datasets for keyword extraction that relates to a non-STEM area, which can potentially empower the research on literature searching and analysis tools in the non-tech academic community. And PM@F can address the problem when some gold keyphrase partially overlap with the predicted keyphrase, which is ignored by traditional metrics of exact match.

Also, in the experimental part, the study combines ablation studies that are typical in machine learning research with comparative studies and case studies that are typical in traditional linguistic research, to present the experimental results in a comprehensive and clear manner.

1.5 Thesis Outline

The study consists of five major parts.

Chapter 1 mainly introduces the research background and significance of keyword extraction given the data deluge in the information era, which shows the tremendous practical value such techniques can make in information retrieval of large-scale scientific documents. To meet the requirement of high-quality keywords in the academic context, the author proposes to model the four innate features of keywords by integrating structural and semantic information, and thus elicits the proposed algorithm SSRank as well as major contributions of the paper. Finally, the author gives the organizational structure of the thesis and summarizes the content of each chapter.

Chapter 2 gives a synthetic summary of existing knowledge in the field of unsupervised keyword extraction, and lays the theoretical foundations supporting the SSRank model. Section 2.1 briefly introduces the definition and history of keyword extraction technology, and highlights the flexibility but underperformance of unsupervised approaches. Section 2.2 further introduces three branches of the most representative and well-known existing methods of unsupervised keyword extraction

with an analysis of problems and defects of these methods from a macro perspective. Section 2.3 is dedicated to a clarification of relevant theoretical and technical understructures, including the PageRank algorithm and semantic representations of text. Section 2.4 mainly introduces evaluation datasets and metrics typically used in experiments. Finally, Section 2.5 summarizes this chapter.

Chapter 3 is an introduction to the novel unsupervised AKE algorithm proposed in the thesis: *Structural and Semantic Rank* (SSRank). The author first briefly introduces the concept and idea behind algorithmic design. Then, each chapter is dedicated one of the four essential steps of the SSRank system: Section 3.1 for candidate formation, Section 3.2 for structural scoring, Section 3.3 for semantic clustering, and Section 3.4 for keyword selection.

Chapter 4 centers around the description of experimental setup and results. Section 4.1 introduces the construction of the JML corpus, and then analyzed the dataset statistics and data preparation for evaluation, as well as the design of a new metric for partial matching called PM@F. To verify the effectiveness of SSRank, Section 4.2 showed results of ablation studies for SSRank, and Section 4.3 evaluated the overall effectiveness of SSRank compared with three baseline methods, including a case study. Beyond the quantitative and qualitative results, Section 4.4 further provided an application scenario SSRank can make in the scholarly context.

Chapter 5 concludes the thesis and discusses the limitations of the study.

Chapter 2 Literature Review

This chapter gives a synthetic summary of existing knowledge in the field of unsupervised keyword extraction, and lays the theoretical foundations supporting the SSRank model. Section 2.1 briefly introduces the definition and history of keyword extraction technology, and highlights the flexibility but underperformance of unsupervised approaches. Section 2.2 further introduces three branches of the most representative and well-known existing methods of unsupervised keyword extraction with an analysis of problems and defects of these methods from a macro perspective. Section 2.3 is dedicated to a clarification of relevant theoretical and technical understructures, including the PageRank algorithm and semantic representations of text. Section 2.4 mainly introduces evaluation datasets and metrics typically used in experiments. Finally, Section 2.5 summarizes this chapter.

2.1 Definition and History of Keyword Extraction

In computational linguistics and information extraction research, **keywords** are more commonly called keyphrases, as they often appear as multi-word phrases instead of a single word. Formally, keywords or keyphrases are defined as "a selection of short, significant expressions consisting of one or more words that can summarize the content of the text very compactly." (Meng et al., 2022, p.10) In the context of scholarly documents, keywords are also called key terms[1], which are the linguistic representation of concepts in a research domain (Castellví, 2003).

Automatic Keyword Extraction (AKE), or simply keyword extraction, means the process of automatically extracting a set of word spans from a given sequence of text that best represent the main content of the original text. As a building block of text mining and information retrieval, it plays an important role in the discovery of useful information from unstructured textual sources (Feldmann, 1999; Salloum et al., 2018).

[1] In this thesis, the author will use "keywords", "keyphrases" and "key terms" (in the scholarly context) interchangeably as they are regarded the same.

It is also closely related to research topics like Automatic Term Recognition (ATR), which is the extraction of linguistic representation of concepts in a research domain (Castellví, 2003); and Automatic Keyphrase Generation (AKG), which differs from AKE in its capability of generating absent keywords that never appear in the text. But since AKG methods are only recently invented and involve neural networks trained on a gigantic corpus like the famous CopyRNN (Meng et al., 2017), this study mainly focuses on AKE methods that extract keywords present in text only.

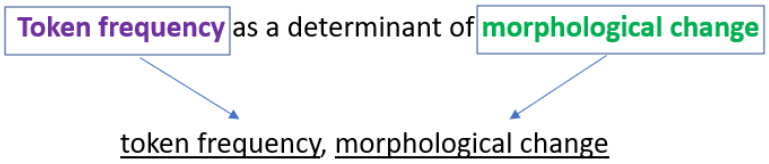

Fig 2.1: A visual demo of automatic keyword extraction

The task of keyword extraction has attracted high academic interest since the 90s. Initially, keyword extraction was employed for cataloging and indexing documents in digital libraries (Fagan, 1987), and the method is as simple as to assign thesaurus entries, for example, the controlled medical vocabularies MeSH (Mary et al., 2004) and UMLS (Bodenreider, 2004) in PubMed**.** Later since the 2000s, two broad categories of methods that do not solely rely on controlled vocabulary have been applied: the **supervised** approaches that rely on a set of features as input to train a binary classifier to determine whether a candidate is a keyphrase or not as output, e.g. KEA (Witten et al., 1999) and WINGNUS (Nguyen & Luong, 2010); and the **unsupervised** approaches which try to infer the structure of a dataset without the use of training data, e.g. TextRank (Mihalcea & Tarau, 2004) and YAKE! (Campos et al., 2018).

Compared to supervised keyword extraction methods which require manual label preparation, there are four main advantages of unsupervised methods: (1) unsupervised methods are less costly and more easily approachable as manually annotating a high-quality and large-scale corpus is a laborious task; (2) as human annotations are often inconsistent at label assignment and notorious for various kinds of biases, unsupervised methods are less vulnerable to such fluctuation in objectivity; (3) unsupervised methods

are more domain independent, flexible and widely applicable across different situations.; (4) without training procedures, unsupervised models are usually computationally cheap.

Despite the wide applicability and popularity of unsupervised approaches, their drawback is also obvious: since unsupervised models do not learn from pre-assigned answers, they usually perform not as well as supervised methods. In order to improve the accuracy of unsupervised methods, scholars have made various kinds of efforts. The following section will give an introduction of the three most representative branches.

2.2 Previous Works on Unsupervised Keyword Extraction

The field of keyword or keyphrase extraction is a vigorous testbed for scientific research as we see many different methods spouting out in recent years. By searching "keyphrase extraction" "term extraction" "keyword extraction" through Google Scholar, one would get tons of hundreds of different unsupervised algorithms previously invented. I divide them into three different kinds: *statistics-based*, *graph-based*, and *language model-based*, although it is possible that some statistical or graphical methods may integrate word embeddings or other pre-trained language models, which combine into a hybrid method.

Fig 2.2 presents a hierarchical structure of the three branches of unsupervised keyword extraction methods: (1) statistics-based methods mainly consider the statistical properties of phrases relating to the document or corpus; (2) graph-based methods utilize graphical theories to compute candidate importance; (3) language-model based methods leverage pretrained language models to compute semantic scores.

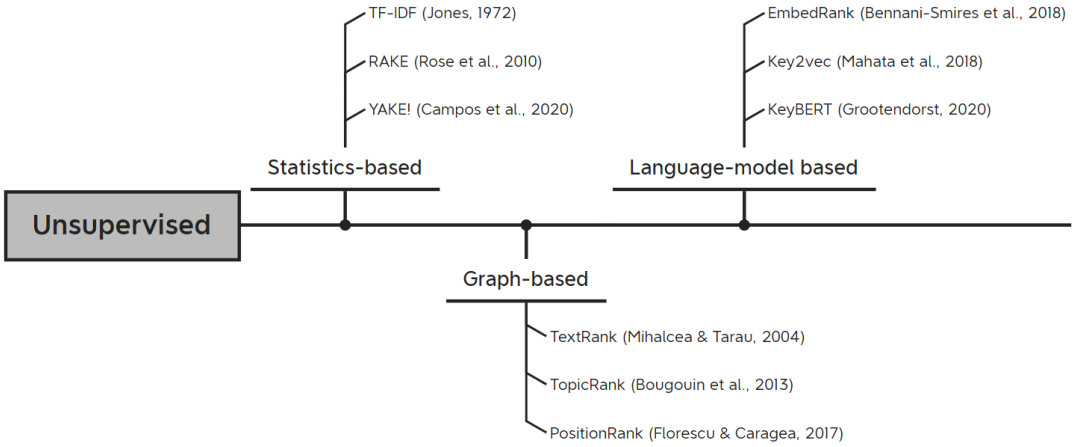

Fig 2.2: Summary of three kinds of unsupervised keyword extraction methods

These methods follow a very similar methodological paradigm that is typical in unsupervised keyword extraction: (1) first, they select candidate phrases using some text processing heuristic; (2) then they assign a weight or score to each candidate and rank them accordingly; (3) finally, they select the N highest-ranked candidates as the resulting keywords (Papagiannopoulou & Tsoumakas, 2020).

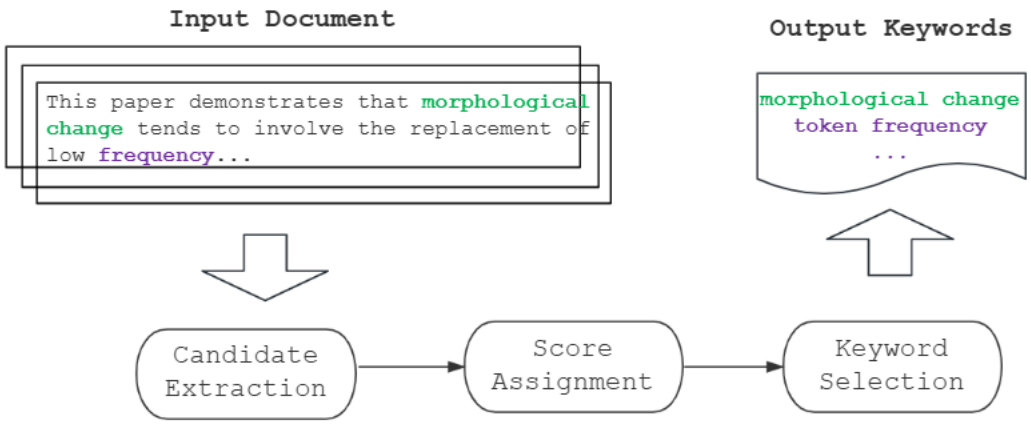

Fig 2.3: The typical methodological paradigm of unsupervised keyword extraction

The following three subsections are devoted to a brief introduction to each kind of methods, highlighting their innovative points and salient components.

2.2.1 Statistics-Based Methods

(4) TF-IDF

TF-IDF stands for Term Frequency-Inverse Document Frequency, first invented by Jones (1972). It measures how frequently a term appears in a single document as compared to the entire corpus, by weighing its frequency to collection frequency. With this very simple procedure, less frequent and more specific terms are given greater value. It is computed as the multiplication of Term Frequency, meaning the number of instances of a term in a single document only, and Inverse Document Frequency, a measure of whether a term is rare or common in a collection of documents. The specific computation goes as follows:

$$TF - IDF \text{ Score } = TF_{t,d} * IDF = TF_{t,d} * \log \frac{N}{df}$$

In this equation, $TF_{t,d}$ means the number of times a term t appears in document d, N means the total number of documents, and df means the number of documents that contain the term t. TF-IDF is a strong baseline of keyword extraction (Li et al., 2007) and query retrieval (Ramos, 2003), but it relies on external corpus statistics.

(5) RAKE

The RAKE Algorithm (Rose et al., 2010) is another domain-independent and language-independent algorithm that utilizes statistical text features, and it works on single documents. It uses a list of stop words as a set of phrase delimiters and word delimiters to partition the document into candidate keywords, which are all content words. Then, it calculates the word scores based on the computation of three indicators of in the term occurrence matrix: (1) term frequency; (2) term degree (the number of different terms that it co-occurs with), and (3) ratio of degree to frequency (see Fig 2.2). Finally, it obtains the candidate keyword scores by adding their component word scores.

	algorithms	bounds	compatibility	components	constraints	constructing	corresponding	criteria	diophantine	equations	generating	inequations	linear	minimal	natural	nonstrict	numbers	set	sets	solving	strict	supporting	system	systems	upper
deg(w)	3	2	2	1	2	1	2	2	3	3	3	4	5	8	2	2	2	6	3	1	2	3	1	4	2
freq(w)	2	1	2	1	1	1	1	2	1	1	1	2	2	3	1	1	1	3	1	1	1	1	1	4	1
deg(w) / freq(w)	1.5	2	1	1	2	1	2	1	3	3	3	2	2.5	2.7	2	2	2	2	3	1	2	3	1	1	2

Fig 2.4: Scores calculated from the word co-occurrence matrix (Rose et al., 2010)

(6) YAKE!

YAKE! (Campos et al., 2020) is a recent new approach which employs a statistical analysis that pays particular attention to structure (casing and term position), term frequencies, term relatedness to context, and co-occurrence. The quantification of each of these features involves various different equations that encode detailed information like uppercase, acronym, frequency, the number of co-occurring words, etc., which makes the algorithm meticulous and complicated. The final equation for the term score is presented below:

$$S(t) = \frac{T_{Rel} * T_{\text{Position}}}{T_{\text{Case}} + \frac{\text{TF}_{\text{Norm}}}{T_{\text{Rel}}} + \frac{T_{\text{Sentence}}}{T_{\text{Rel}}}}$$

2.2.2 Graph-Based Methods

(1) TextRank

Among various graph-based algorithms, the TextRank model proposed by Rada Mihalcea and Paul Tarau (2004) is the most representative and typical one. TextRank is one of the first works that introduce the idea of PageRank algorithm into the field of keyword extraction, and determines whether lexical units with varying granularity extracted from the source text can be counted as a keyword by situating them in the structure of a text graph, with nodes corresponding to these lexical units and edges corresponding to their association pattern. It ranks the importance of vertices within the graph by taking into account global information computed recursively through the entire graph, and selects the most important vertices as keywords.

Originally, the paper uses co-occurrence relation to define the edges, but the types of relation to draw connections can be any “lexical or semantic relations, contextual overlap, etc.” (p. 2) Similarly, the lexical units can be various sizes and characteristics, like words, collocations, etc. The formula for graph-based ranking proposed by TextRank is as follows:

$$WS(V_i) = (1-d) + d * \sum_{V_j \in \text{In}(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$

(2) PositionRank

PositionRank (Florescu & Caragea, 2017) is an improved extension of TextRank by modeling positional information of word occurrences using a biased PageRank. The idea behind PositionRank is to "assign larger weights (or probabilities) to words that are found early in a document and are frequent." (p. 1108) Specifically, it sets a biased probability vector $\tilde{p}$ which represents different probabilities that a random walk would jump to different node in the graph. And nodes that have higher probability are preferred in the jump and thus receive higher scores in the ranking.

The biased probability vector $\tilde{p}$ is obtained by aggregating all inverse positions for each given word, and then normalizing the weights:

$$\tilde{p} = \left[\frac{p_1}{p_1 + p_2 + \cdots + p_{|V|}}, \frac{p_2}{p_1 + p_2 + \cdots + p_{|V|}}, \ldots, \frac{p_{|V|}}{p_1 + p_2 + \cdots + p_{|V|}} \right]$$

The PageRank score of each node is then retrieved through recursive computation using the following equation:

$$S(v_i) = (1-\alpha) \cdot \tilde{p}_i + \alpha \cdot \sum_{v_j \in \text{Adj}(v_i)} \frac{w_{ji}}{O(v_j)} S(v_j)$$

(3) TopicRank

While the above graphical methods achieve already good results, the keyphrase candidates are often redundant in regard to the topics they represent. TopicRank (Bougouin et al., 2013) makes another kind of enhancing effort on TextRank by extracting keyphrases from the most important topics of a document, where topics are defined as clusters of similar candidates.

The specific extraction process goes as follows: first, the document is preprocessed using sentence segmentation, word tokenization and Part-of-Speech tagging, and then keyphrase candidates are clustered into topics using hierarchical agglomerative clustering. The topics form an interconnected complete graph with edges weighted according to the strength of the semantic relations between topics, i.e. the reciprocal

distances between the offset positions of the candidate keyphrases:

$$w_{i,j} = \sum_{c_i \in t_i} \sum_{c_j \in t_j} \text{dist}\left(c_i, c_j\right)$$

$$\text{dist}\left(c_i, c_j\right) = \sum_{p_i \in \text{pos}\,(c_i)} \sum_{p_j \in \text{pos}\,(c_j)} \frac{1}{|p_i - p_j|}$$

Then, PageRank model is used to assign a significance score to each topic. Finally, keyphrases are extracted by selecting one keyphrase candidate for each of the most important topics according to first appearance, frequency or the centroid.

2.2.3 Language Model-based Methods

(1) EmbedRank

EmbedRank (Bennani-Smires et al., 2018) is a typical language model-based method that leverages language models called sentence embeddings. The method extracts candidate phrases that follow the adjectives plus nouns POS pattern, and then embeds both the candidates and the document in the same high-dimensional vector space, and finally ranks candidates according to the semantic relatedness between the candidate and document as a proxy for informativeness, and semantic distance between candidates as an indicator of diversity. For the final selection of keyphrases, EmbedRank uses a modified MMR (embedding-based maximal marginal relevance) to balance the relevance and diversity with the factor lambda:

$$\text{MMR:} = \underset{C_i \in C \setminus K}{\arg max} \left[\lambda \cdot \widetilde{\cos_{\text{sim}}}(C_i, \text{doc}) - (1-\lambda) \underset{C_j \in K}{max} \widetilde{\cos_{sim}}\left(C_i, C_j\right)\right]$$

(2) Key2Vec

Another typical language model-based method is Key2Vec (Mahata et al., 2018) which makes use of domain-specific phrase embeddings for keyphrase extraction from scientific articles. But it is also a hybrid algorithm that employs a graph-based ranking approach.

First, it splits a text document into sentences, tokenizes a sentence into unigram tokens, and identifies noun phrases and named entities from it. Then, it directly trains

multiword phrase embeddings using *Fasttext*, with the training vocabulary consisting of both unigram as well as multi-word phrases. The main aim of the underlying embedding model is to capture semantic and syntactic similarities between textual units. Next, it calculates the cosine distance between the theme vector, which is the thematic representation or the main theme of the article, and vector for each candidate keyphrase, which are constructed using the same embeddings, and assign a score to each candidate, with 1 indicating a complete similarity with the theme vector and 0 indicating a complete dissimilarity. Finally, similar to TextRank, it ranks the candidate phrases using the theme-weighted PageRank algorithm:

$$R\left(c_j^{d_i}\right) = (1-d)w_{c_j}^{d_i} + d \times \sum_{c_k^{d_i} \in \varepsilon\left(c_j^{d_i}\right)} \left(\frac{sr\left(c_j^{d_i}, c_k^{d_i}\right)}{\left|\text{out}\left(c_k^{d_i}\right)\right|} \right) R\left(c_k^{d_i}\right)$$

(3) KeyBERT

KeyBERT (Grootendorst, 2020) is another state-of-the-art language-model based method that focuses on semantic similarity. It employs the CountVectorizer from Scikit-Learn to tokenize the given input document into n-grams which are regarded as candidates. Then, it leverages pretrained BERT embeddings to transform the entire document and the candidate keywords/keyphrases into vectors. Finally, it ranks these vector representations of candidates according to cosine similarity to the embedding of the original document.

2.2.4 Problems of Existing Approaches

Although the above approaches have provided various kinds of effective solutions to the task of keyword extraction, they are subject to a series of problems during each stage of the methodological paradigm.

(1) The Extraction of Candidate Phrases

In the candidate extraction stage, most existing algorithms use simple n-gram or regex matching approaches. The most common strategies are to extract:

- ✧ all possible n-grams with n ranging from 1 to 6;

✧ all sequences that match the regex pattern of Adj*Noun+.

The thesis argues that the n-gram method has three shortcomings. First and most importantly, it results in keyphrases that are meaningless or broken, e.g. “methodology to track” “track the changing”. This severely violates the principle of termhood, which states that keywords should be uninterrupted and holistic syntagmatic combinations that make complete sense. Second, it constrains the candidate to specific range of length, with keywords shorter or longer than the fixed number discarded. This behavior leaves out a small number of qualified terms and curtails the final accuracy. Third, the method can generate an exceedingly huge quantity of n-grams. In fact, a sequence of m words can produce a number of n-grams as many as $\frac{n(2m-n+1)}{2}$, which include many cases of redundancy and overlap. Such a big number brings challenges for subsequent filtering and selection.

The regex matching method can overcome the problems of brokenness, but the regex pattern is rather naïve. It ignores phrases containing punctuations (e.g. “retrieval-induced forgetting” “BIA+”), prepositions (e.g. “judgments of learning” “exposure to non-native speech”), as well as cased verbs like gerund (e.g. “task switching” “reading aloud” “sentence processing” “semantic priming”) and past participles before nouns (e.g. “linguistically mediated visual search”).

Thus, to extract eligible and proper phrases that perfectly satisfy the rule of termhood, it may be helpful to make better use of part of speech tags and dependency parsing when segmenting the text sequences into candidate phrases.

(2) The Assignment of Scores

During the score assignment stage, the frequency of candidate phrases plays an important role for statistics-based methods like TF-IDF, rendering them vulnerable to the problem of low frequency. Although the frequency that keywords appear in text is presumably high, there are many cases where a term only appear once or twice in the text. Statistical methods that are dependent upon the frequency threshold make less frequent terms less likely to be detected, and some false positive candidates with high frequency are easily counted in.

Graph-based methods can overcome the frequency problem by situating the words in an interrelated network, but they are often subject to the length problem. Since they typically use simple addition of component words when computing keyphrase scores, as in the case of TextRank, longer terms like “fuzzy modeling approach” usually receive higher importance scores than shorter terms like “fuzzy modeling” and thus rank higher.

To tackle the frequency problem and length problem in candidate score assignment, more work remains to be done in modifying the model structures.

(3) The Final Selection of Keywords

In the final stage of keyword selection, the problem of lack of diversity arises due to the neglection of correlation among keyphrases, resulting in duplicated information and poor coverage. The generated keyphrases are often semantically similar and focusing on the same topic, instead of balancing a variety of meaningful and critical points conveyed in the source document.

For example, the above approaches behave not well in removing duplicates, like treating “feedback control” and “feedback stochastic controls” as different words. When both words rank high in the resulting list, they are likely to be selected as separate keyphrases, although they express nearly the same meaning. There are other cases where previously generated phrases appear twice in the final output, including morphological variants of the same words, like “schema diagrams” and “schema diagram”; phrases synonymous to each other, like “information age” and “information era”; and phrases nested in another, meaning to appear within another longer phrase, like “support vector” in “support vector machine”.

Since more diverse keywords can provide readers with more information and cover more important segments of the original text, the diversity issue should never be negligible.

2.3 Theoretical and Technical Foundations

As the proposed SSRank model utilizes the PageRank algorithm when calculating keyword importance scores, as well as word embeddings when computing semantic

similarity between candidate phrases. This section mainly introduces the basic theories and technologies involved in these two parts.

2.3.1 The PageRank Algorithm

Graph-based models are largely adopted from the well-known Google PageRank algorithm (Brin and Page, 1998). This subsection introduces the generalities of the original PageRank to lay the theoretical foundation, which serves as a basis for subsequent application and improvement of it in the context of keyword extraction.

PageRank algorithm is originally designed to rank the matching results of searching query based on web link analysis. It is based on the traditional thought of citation analysis: if one page is linked to the other, then the other page is considered to gain the contribution score from the first page. In other words, the scores of pages are inner dependent according to their connections. As many links point to each other, score calculation becomes an iterative process, and finally the pages are ranked by the scores.

Specifically, the idea is to build a directed graph $G = (V, E)$ with a set of vertices V and a set of edges E, where E is a subset of $V \times V$. The PageRank value of a web page or vertex V_i is obtained by the follow formula:

$$S(V_i) = (1 - d) + d * \sum_{j \in \mathrm{In}(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

where $S(V_i)$ is the weight of webpage V_i, $In(V_i)$ is a set of inbound links or predecessors of V_i, $Out(V_j)$ is a set of out-going links or successors of V_j, and $|Out(V_j)|$ is the number of outgoing links. It should note that d is a human-assigned damping factor to ensure that the random walk does not get stuck into cycles of the graph and allow the "teleport" operation to another node in the graph, and $1 - d$ represents the equal probability of a web surfer randomly landing on a web page if without outgoing links.

2.3.2 Semantic Representation and Similarity Computation

As John Firth's famous quotation goes, "You shall know a word by the company

it keeps" (Hilpert & Saavedra, 2017, p.2). The distributional hypothesis indicates that words that occur in similar contexts tend to have similar meanings (Rubenstein & Goodenough, 1965; Pantel, 2005). It is interesting to see this abstract hypothesis of meaning put into concrete algorithms that deal with mathematical concepts like vectors, matrices, and higher-order tensors (Turney & Pantel, 2010). By anchoring the context as the set of words that appear within a fixed-size window, we are able to build a dense vector as a semantic representation of a word by the many contexts of it, called "word vectors" or "word embeddings".

Facilitated by these word vectors, one can compute the "physical" distances between words in a vector space in order to measure their similarity: the closer they are located in the vector space, the more common the contexts they share in the corpus and the more similar they are. This significantly transforms traditional ways of encoding semantic information, like using the WordNet thesaurus (Miller, 1995) to get lists of synonym and hypernyms. Using simple vector arithmetic and just cosine similarity or the Euclidean distance, the semantic relationship of two entities can be acquired.

Word2vec (Mikolov et al. 2013) is a famous framework for learning such distributed representation of words. It trains word vectors using two simple probabilistic methods called continuous bag-of-words (CBOW) and skip-gram, shown in Fig 2.6. CBOW aims to predict the current word as the output from the surrounding context as the input, for example to treat "The cat _ over the puddle" as a context and from these words to predict or generate the center word "jumped". Skip-gram does the opposite, and predicts the distribution (probability) of context words as the output from a center word as the input. And there are two kinds of training objectives: negative sampling and hierarchical softmax. Negative sampling defines an objective by sampling several negative examples for every training step instead of looping over the entire vocabulary, while hierarchical softmax defines an objective using an efficient tree structure to compute probabilities for all the vocabulary. The hidden layer that the input is fed into and trained is obtained as the final embedding vector.

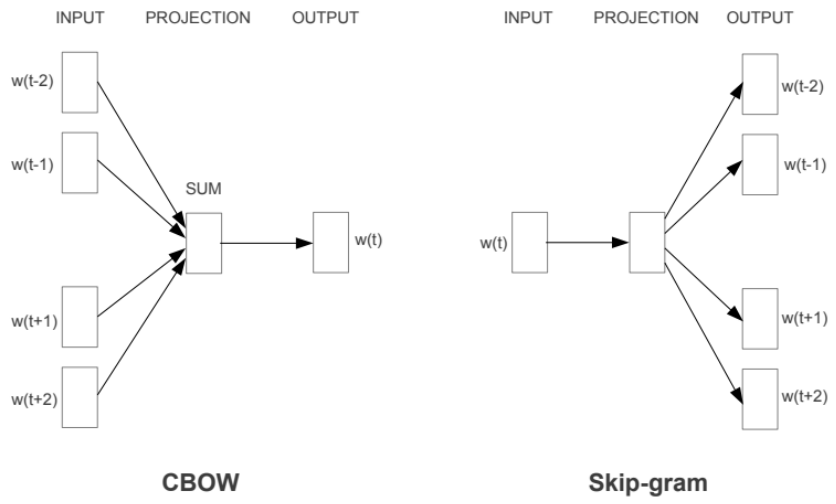

Figure 2.6: Model architecture of CBOW and Skip-gram (Mikolov et al. 2013)

GloVe (Pennington et al., 2014) is another famous method developed by Stanford NLP group for training distributed representations. While Word2vec is a "predictive" model that learns vectors by optimizing a feed-forward neural network using SGD to minimize loss, the Glove model leverages matrix factorization techniques to yield a lower-dimensional vector representation from the high-dimensional word-context co-occurrence counts matrix of the entire corpus. Although their resulting word vectors have subtly different properties, both models give similar results for many tasks in practice.

FastText (Joulin et al., 2016) is another word embedding method that extends word2vec by representing each word as an n-gram of characters. While Word2Vec and GLOVE treat each word as the smallest unit to train on, FastText uses n-gram characters as the smallest unit. For example, the word vector of "apple" could be broken down into separate word vector units as "ap" "app" "ple". The biggest benefit of using FastText is that it generates better word embeddings for rare words, or even words not seen during training because the n-gram character vectors are shared with other words. This is something that Word2Vec and GLOVE cannot achieve.

Doc2vec (Le and Mikolov, 2014) is a document version of Word2Vec that seeks to convert documents into vectors. In the training process, a set of tagged documents with labels being name of the document are feed into the neural network to create a

vector representation of a group of words taken collectively as a single unit. It doesn't only give the simple average of the words in the sentence. Doc2vec uses "distributed memory" (dm) and "distributed bag of words" (DBOW) options, with the former being the default in Doc2Vec.

While Word2vec is a popular technique that trains double-layered shallow neural networks to model a word and its linguistic contexts in the vector space, the semantic representation remains fixed irrespective of the context within which the word appears. **BERT** (Devlin et al., 2018) is a different state-of-the-art pre-trained model of contextualized embeddings that overcomes this problem by constructing a deep network of Transformer blocks (Vaswani et al., 2017). It generates dynamic word representations that are informed by the contextual words in their proximity. This is achieved by its stacked self-attention mechanisms to capture relationships across different positions in a sequence.

2.3 Evaluation Logistics

In this section, we describe the evaluation logistics, namely the datasets and evaluation metrics.

2.4.1 Relevant Datasets

(1) SemEval2017

SemEval2017 is the 11th International Workshop on Semantic Evaluation, organized as a shared task evaluation for a variety of natural language processing and computational linguistics tasks, including sentiment analysis, word sense disambiguation, and stance detection. The SemEval2017 Task 10 dataset (Augenstein et al., 2017) is tailored for keyword and relation extraction from scientific documents. It consists of 500 paragraphs selected from journal articles, with the domains ranging from Computer Science, Material Sciences to Physics. The keywords for each text are double-assigned by one undergraduate student volunteer studying relevant major and an expert annotator, with the annotation of the latter prioritized when encountering

disagreement.

(2) INSPEC

INSPEC is a bibliographic database that covers all specialized fields including physics, electrical engineering, mathematics and computer science, widely used as a reference resource in academic and research communities. The database includes rich metadata, such as author names, publication dates, and keywords assigned by experts. The INSPEC dataset (Hulth, 2003) is composed of 2000 scientific abstracts with corresponding title and keywords collected from the Inspec database during the period 1998 and 2002 pertaining to CS related disciplines. Each document is manually annotated with two types of keywords: the controlled keywords that are included in the INSPEC thesaurus, and the uncontrolled keywords that are assigned by the editors. Both may or may not appear in the document.

(3) KDD

The KDD dataset refers to the articles published on the leading ACM Conference on Knowledge Discovery and Data Mining (KDD) between the years 2004-2014. The KDD conference is widely regarded as one of the most important events in the data mining and data science community. The articles cover an extensive variety of topics including big data analytics, machine learning, data visualization, database and data management, social network analysis, and more. There are in total 755 documents and each is annotated with author-specified gold keywords.

(4) WWW

The WWW dataset (Gollapalli & Caragea, 2014) is anther large-scale resource consisting of abstracts of papers collected from the WWW conference published in the years 2004-2014. The World Wide Web Conference (WWW) is an annual academic conference focused on research and development related to a wide range of topics, including web search and data mining, web science, web architecture and standards, social media, and web applications and services. The dataset contains a collection of 1330 documents labeled with author-input keywords.

2.4.2 Evaluation Metrics

Although a host of statistical and neural methods haven been proposed, the evaluation metrics for keyword extraction are relatively simple and direct. The three most commonly used metrics are *precision*, *recall*, and *F-score* (Goutte & Gaussier, 2005). They are computed by means of comparing the extracted keywords with the gold reference or ground truth: precision is the number of correctly-predicted keyphrases divided over the number of all predicted keyphrases; recall is computed as the ratio of the number of correctly predicted keyphrases to the total number of ground truth keyphrases; and F-score is the combination of precision and recall by harmonic mean.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$F\text{-score} = \frac{2\ \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Since the number of predicted keyphrases can vary, some improvements are made to the original metrics. Following Meng et al. (2017), people use Precision@K, Recall@K, F1@K to choose only the top k predictions for evaluation, and the commonly used k is 5 or 10. Other frequently used metrics include F1@M and F1@O, proposed by Yuan et al. (2020), in which M denotes the number of predicted keyphrases and O means the number of ground truth keyphrases.

While useful, these accuracy-like metrics are not sufficient for evaluating keyphrase prediction models. By summarizing the performance as a single aggregate statistic, it becomes difficult to figure out where the model is failing, and how to fix it (Wu et al., 2019). Merely comparing different models through numbers is straightforward, but does not tell us much about the pros and cons of each model. As Meng et al. (2022) pointed out, one of the future challenges of keyphrase prediction lies in the lack of clearly-defined metrics that measure some specific properties of keyphrases.

2.5 Chapter Summary

The first section is an overview of definition and history of keyword extraction. Automatic Keyword Extraction (AKE) serves as a building block of text mining and information retrieval technologies. Its development traces back to the 90s, when thesaurus entry assignment was used for cataloging and indexing documents in digital libraries (Fagan, 1987). Later since the 2000s, two categories namely supervised and unsupervised methods have come into use. Compared to supervised approaches which require manual label preparation, unsupervised AKE methods are less costly and thus more widely applicable, but usually perform not as well as supervised methods.

Then in the second section, the author gives a summary of three major branches of unsupervised keyphrase extraction techniques: *statistics-based*, *graph-based*, and *language model-based*. They follow a similar methodological paradigm. The thesis points out several common problems during each stage: (1) the candidate extraction procedure violates the principle of termhood as it produces a large number of redundant and broken text pieces or phrases with invariably naïve syntactic pattern; (2) the score assignment stage faces the frequency problem and the length problem due to the reliance on frequency computation or simple addition of component scores; (3) the final step of keyword selection is subject to the lack of diversity and informativeness as the generated keyphrases are often similar or generic in meaning.

After the review of previous methods, the third section is dedicated to a clarification of relevant theoretical and technical foundations supporting the SSRank model. The author first introduces the original Google PageRank algorithm, from which derives a large number of graph-based keyword extraction techniques. Then, the author describes a variety of text representation techniques which embed text in a semantic vector space and can be used to calculate semantic distance.

The fourth section introduces four evaluation datasets that consists of scientific abstracts and annotated keywords, namely SemEval2017, INSPEC, WWW and KDD; and three commonly used metrics and their variations, namely Precision, Recall and F-score. These provide the logistics for subsequent experiment.

In summary, this chapter builds the theoretical foundations of unsupervised AKE, which paves the way for the introduction of specific designs of SSRank in the next chapter.

Chapter 3 The SSRank Model

This chapter will introduce the novel unsupervised AKE algorithm proposed in the thesis: *Structural and Semantic Rank* (SSRank). The algorithm is meticulously designed for keyword extraction in scholarly context, catering to the rigid requirement of grammatical completeness and information compactness of terms.

As shown in Fig. 3.1, the SSRank algorithm encompasses four essential steps: (1) Candidate formation. The extraction of candidate noun phrases from the original text. (2) Structural Scoring. The construction of undirected graph with nodes representing tokens and edges representing co-occurrence, and the computation of structural score for each candidate phrase by pooling the PageRank score of its component tokens. (3) Semantic clustering. Grouping phrases according to semantic distance**.** (4) Keyword selection. Ranking of clusters based on average within-group distance or alignment with the theme of document, and the formation of a keyword chain composed of the highest scored candidates from the top ranking clusters. The complete list or the first *K* keywords are selected from this chain to form the final keyword set.

The first four sections of this chapter are dedicated to an elaboration on each of these four steps, and the final section summarizes the chapter.

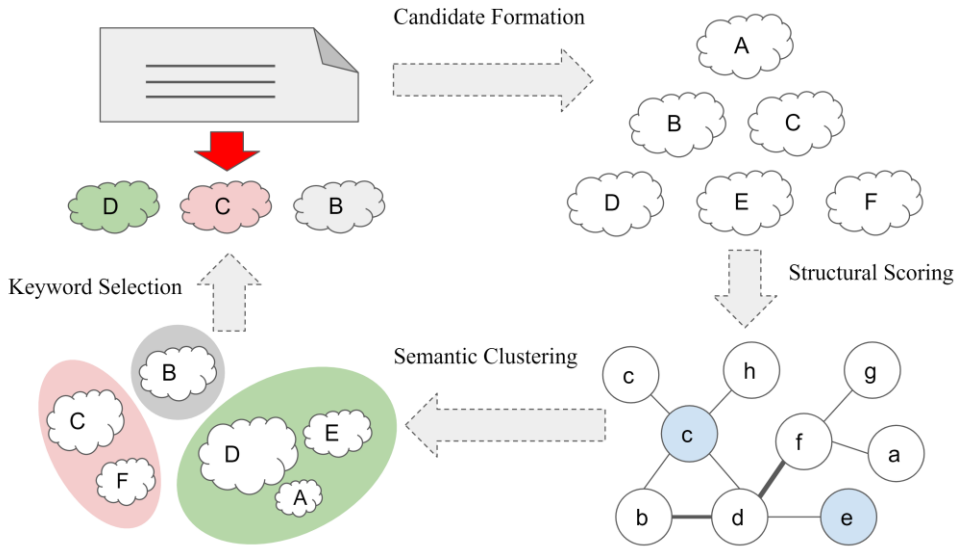

Fig 3.1: Visual demonstration of the four steps in SSRank keyword extraction

For illustration, the thesis chooses the same sample abstract as in Mihalcea and Tarau (2004) due to its small size and convenience for comparison with the original TextRank paper:

Compatibility of systems of linear constraints over the set of natural numbers. Criteria of compatibility of a system of linear Diophantine equations, strict inequations, and nonstrict inequations are considered. Upper bounds for components of a minimal set of solutions and algorithms of construction of minimal generating sets of solutions for all types of systems are given. These criteria and the corresponding algorithms for constructing a minimal supporting set of solutions can be used in solving all the considered types of systems and systems of mixed types.

3.1 Candidate Formation

Candidate formation is a critical procedure in the SSRank algorithm, which involves identifying all possible words or phrases that could potentially serve as keywords. If candidate formation is too restrictive or too broad, it may result in the exclusion of important keywords or the inclusion of irrelevant ones. Thus, the process should be appropriately calibrated.

SSRank designs a linguistic approach of candidate formation that uses heuristics based on the syntactic properties of words. The entire procedure consists of three steps: text preprocessing, syntactic parsing, and noun phrase detection. It should be noted that most of these steps are implemented with Python, NLTK[2] and the spaCy[3] toolkits, but it can be replaced with other free, open-source software libraries for advanced natural language processing like Stanford's CoreNLP. Also, though the implementation is specially designed for English only, similar steps can be applied to any other language supported by the spaCy[4] pipeline.

[2] https://www.nltk.org/
[3] https://spacy.io/
[4] SpaCy supports more than 20 languages, see https://spacy.io/usage/models

3.1.1 Text Preprocessing

As a crucial part of any natural language processing system, text preprocessing serves to clean and transform the raw text into a machine-readable format tailored for later stages of algorithmic strategies. In the preprocessing stage of SSRank, an extensive number of pre-processing steps are applied, including tokenization, noise removal, stemming, and special annotation. These steps are beneficial in reducing the size of vocabulary indexing, extracting useful features, and removing noisy and erroneous data to increase efficiency and accuracy.

(4) Tokenization

The input document is first divided into individual sentences by identifying sentence boundaries like different types of punctuation, abbreviations and contractions. This step is known as sentence splitting or sentence segmentation.

Then, the sentences are further broken into a list of smaller units of words called tokens. For example, tokenizing a sentence like “Compatibility of systems of linear constraints over the set of natural numbers” would generate the following tokens: [“compatibility”, “of”, “systems”, “of”, “linear”, “constraints”, “over”, “the”, “set”, “of”, “natural”, “numbers”].

(5) Noise Removal

First, tokens that are not valid candidates are removed. For example, left and right brackets and content within it are removed, as they usually serve as supplementary information, indexing of reference, or acronyms of terms that are considered trivial. Page numbers like “p.7” and digits appearing alone are also removed, but digits occurring like “N400” “L2” with letters are preserved.

Second, named entities are removed. While upper cased words are often counted as keywords, some of them are references to real-world objects called entities like organizations, places, and people’s names. Thus, the latter parts should be removed from the list of candidates.

(6) Stemming

Stemming is used to reduce the inflectional forms and derivationally related forms of the tokens to their root forms. For example, morphological variants like “investigation” “investigations” “investigating” and “investigated” that carry essentially the same meaning are transformed into the same stem “investigat”. The specific tool used is PorterStemmer from the NLTK package.

It should be noted that stemming and lemmatization are both popular techniques to produce the root form of inflected words, and the thesis uses stemming in extraction and evaluation as it cuts down the form of a word into a most simplified stem that may not be recognized as an actual word, whereas lemmatization is used for final presentation of keywords as it transforms a word into a form that is a valid word in the language called lemma and is thus more readable for humans.

(7) Special Annotation

Tokens that have special and salient properties are annotated. The thesis mainly undertakes two kinds of annotations: position annotation and uppercase annotation. Specifically, as keywords are more likely to appear in the title and in uppercase, candidates are marked as in or out of title, and uppercase or lowercase.

3.1.2 Syntactic parsing

SSRank uses syntactic parsing to annotate the syntactic label of each token, including POS and dependency tags, which prepares for subsequent noun phrase detection.

(1) POS tagging

The first step is Part of Speech (POS) tagging, which assigns each word in the sentence with a grammatical category. In traditional syntax, words are often categorized into nouns, verbs, adjectives, adverbs, which are call lexical parts of speech; as well as determiners, prepositions, conjunctions, etc., which are call functional parts of speech. In computational linguistics, these POS categories can be automatically identified by a

POS tagger. The POS tags can be *universal* which are language independent or *fine-grained* which are language specific. For example in English, we can specify the universal POS tag "NOUN" as plural common nouns (NNS) or singular common nouns (NN). For a full list of fine-grained and universal POS tags[5], see Appendix A.

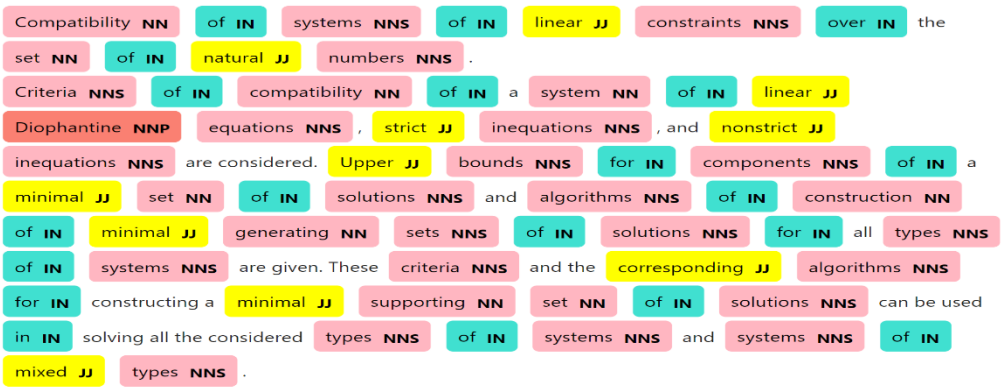

Fig 3.2: Fine-grained POS tagging results by spaCy with specific labels NN, NNS, NNP, IN and JJ colored differently

SSRank labels each token with a fine-grained POS category. This can be implemented with the easy-to-use spaCy pipeline using the pretrained model [6] "en_core_web_sm", with an accuracy of 0.97.

(2) Dependency parsing

The second step is dependency parsing, which identifies grammatical relationships between words. Dependency grammar describes the syntactic structure of a sentence in terms of directed binary grammatical relations directed from head to dependent, e.g. from the verb to its arguments. While each relation has a head and dependent, the sentence has a root node – often the verb – that marks the head of the entire sentence structure. As shown in Fig 3.3, each pair of relation is marked with a directed arc with a label naming the specific relation. For example, the relation between the plural noun (NNS) "constraint" and the adjective (JJ) "linear" is that "linear" is the adjectival modifier ("amod" in short) of "constraint".

[5] https://universaldependencies.org/u/pos/
[6] The information about model accuracy can be found in https://spacy.io/models/en.

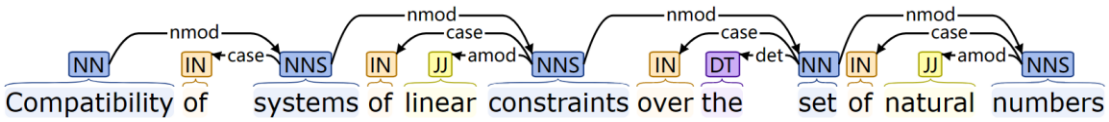

Fig 3.3: Dependency parsing of the sample text implemented and visualized using https://corenlp.run/

For convenience, dependency parsing is again realized by the SpaCy pipeline using the "en_core_web_sm" model[7], with an accuracy of 0.90. For a full list of dependency labels for English, see Appendix B.

3.1.3 Noun Phrase Detection

Given the POS tagging and Dependency Parsing result from the syntactic parsing step, it is possible to build a candidate phrase detector that filters through some prespecified syntactic patterns. As shown in Fig 3.4, though keywords tend to scatter over a wide range of different POS patterns, we can observe that most keywords in the scholarly context are nouns. Thus, the aim of candidate formation is to identify noun phrases. These phrases constitute a set of candidates that are ready for future scoring and ranking.

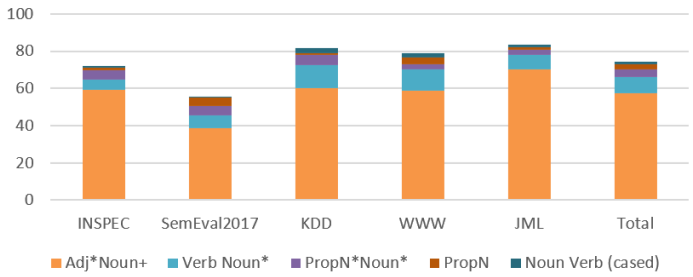

Fig 3.4: Distribution of most common keyword POS pattern per dataset (%)

After careful consideration, the author combines the following regex matching strategies to detect a noun phrase:

✧ Select the longest sequence of nouns and adjectives, including abbreviations and acronyms.

✧ Select phrases which begin with a determiner and end with a noun.

[7] The information about model accuracy can be found in https://spacy.io/models/en.

✧ Select noun phrases containing prepositional “of” “to”, or the hyphen “-”, or verbs as modifier of nouns.

✧ Select phrases which end with gerunds and modified by nouns or adjectives.

(1) Regex Matching Based on Fine-grained POS Tags

NLTK provides a convenience tool called RegexpParser[8] which is a grammar-based regular expression parser for chunking, which can only parse noun phrases based on POS tags. Each time, it finds the first possible match in the sentence and then continue looking for matches afterwards until to the end of sequence. The author realized the above noun phrase matching strategies through the following patterns:

$\#\ the\ longest\ sequence\ of\ adjectives, verbs\ and\ nouns$
$NP1: \{<JJ>*<VBG\|VBN>?<NN.*>+\}$
$NP2: \{<DT><RB>?<JJ>?<VBG\|VBN><NN.*>+\}$
$\#\ \ the\ longest\ sequence\ that\ starts\ with\ DT\ and\ ends\ with\ NN$
$NP3: <DT><JJ\|HYPH\|NN.*\ \|VBG\|VBN>+<NN.*>$
$\#\ nouns\ that\ contain\ \ prepositions$
$NP4: \{<JJ>+<NN.*>+<IN\|TO><NN.*>+\}$
$\#\ nouns\ with\ hyphen$
$NP5: <JJ\|NN.*><HYPH><JJ>*<VBG\|VBN>?<NN.*\ \|VBG>+$

Table 3.1: Regex matching patterns based on POS tags using NLTK RegexpParser

Using the above patterns, the NP detector can extract grammatically sound noun phrases that do not overlap with each other. Since the regex matcher goes in an progressive way that always extracts the largest spans of matching text, one pattern may be shrouded by a stronger pattern, and purely using POS tags for noun phrase detection is likely to bring noise and errors due to the complex structure of some sentences. The noun phrases extracted from the sample text are presented as follows:

upper bound, criteria, compatibility, component, construction, solution, linear constraint, system of linear constraint, linear Diophantine equation, minimal set,

[8] https://tedboy.github.io/nlps/generated/generated/nltk.RegexpParser.html

minimal generate set, minimal support set, set, type of system, system, nonstrict inequation, strict inequation, corresponding algorithm, algorithm, mixed type, type, set of natural number, natural number

(2) Hybrid Regex Matching of Text, Dependency and Universal POS Tags

Since NLTK RegexpParser cannot match full information, the author uses the more flexible spaCy's dependency matcher[9] to construct a hybrid noun phrase detector that can match anything, including POS and dependency tags as well as the exact text. It is a matcher engine that gives people access to not only the tokens within the document but also their relationships.

Before the matching process begins, the author first merges word sequences linked with hyphens into one word like "off-ground" to "offground". And then the author uses the two patterns in Table 3.2 to detect a noun phrase. The first pattern extracts any normal NP with its head being a noun or gerund that takes the role of subject, object, complement, conjunction etc. in the sentence, and then extract its subtree elements, which combines to form a complete noun phrase. The second pattern extracts a noun phrase containing "of".

	TAG	DEP	TEXT	OP
Pattern_1: *Normal NP*	*JJ, NN.*, HYPH, VBG, VBN, RB*	Not in *[mark, neg]*		*
	NN., VBG*	*agent, dobj, pobj, nsubj, nsubjpass, csubj, csubjpass, conj, attr, ccomp, pcomp, xcomp, appos, dative, dep*		
Pattern_2: *NP with "of"*	*JJ, NN.*, HYPH, VBG, VBN, RB*	Not in *[mark, neg]*		+
	IN		of, to	
	JJ, NN., HYPH, VBG, VBN, RB*	Not in *[mark, neg]*		*
	NN., VBG*	*agent, dobj, pobj, nsubj, nsubjpass, csubj, csubjpass, conj, attr, ccomp, pcomp, xcomp, appos, dative, dep*		

[9] https://spacy.io/usage/rule-based-matching#dependencymatcher

Table 3.2: Regex matching patterns based on dependency tags and universal POS tags using spaCy dependency matcher

Assisted by the rich syntactic information, it is inspiring to see that this NP detector can extract noun phrases from any kind of document with perfect grammaticality and a variety of lengths! Taking the sample text as an example, the full set of extracted noun phrases are:

criteria, system of linear Diophantine equations, types, nonstrict inequations, systems, types of systems, mixed types, supporting set, solutions, compatibility of systems, construction of minimal generating sets, components, inequations, sets, sets of solutions, construction, equations, constraints, Diophantine equations, considered types, system, generating sets, upper bounds, minimal generating sets of solutions, constructing, bounds, minimal set of solutions, solving, systems of linear constraints, systems of mixed types, considered types of systems, corresponding algorithms, set of solutions, criteria of compatibility, minimal supporting set, minimal generating sets, supporting set of solutions, minimal set, linear Diophantine equations, algorithms, generating sets of solutions, numbers, compatibility, set of natural numbers, natural numbers, algorithms of construction, linear constraints, minimal supporting set of solutions, strict inequations, set

(3) Stopword Removal

The NP detector is not perfect in producing candidate phrases, as it is very likely to contain noises like “this study” “all results” that are multi-word expressions that bear little information. These noises are further removed by filtering through a stopword list, which contains function words that carry no practical meaning as well as content words that are unlikely to appear in a keyphrase. A partial list of stopwords is presented in Table 3.3. The complete list can be found in Appendix.

Stopwords
et, task, such, bad, all, other, many, very, data, than, further, paper, that, only, four, work, experiment, good, one, result, well, few, paper a, no, our, any, now, results, analysis, too, then, this, system, some, five, research, -, model, once, own, both, method, more, problem, each, the, these, those, same, approach, its, even, so, three, study, two, again, most...

Table 3.3: A partial list of stopwords

3.2 Structural Scoring

The second procedure is the assignment of structural scores to the set of tokens

contained in the candidate phrases. To this aim, SSRank constructs a textual graph that encodes the organization of the words and their relationships, embodied respectively as the nodes and edges in the graph. In this way, the text is abstracted into a mathematical object that allows the computation of node importance using relevant graph theory.

The procedure can be broken down into three steps: (1) Graph formation. SSRank builds an undirected textual graph using words contained in the set of candidate phrases as vertices, and their co-occurring relations as edges. (2) Single word score assignment. (3) Phrase pooling. After getting the importance score for each single word, we can obtain the importance of candidate phrases by pooling their component word scores, and sort them in descending order. These steps are elaborated below.

3.2.1 Graph Formation

SSRank builds a textual graph $G = (V, E)$ for document D to represent the internal structure of the document, where V and E represent a set of vertices and edges, and $E \subseteq V \times V$.

(1) Vertices

According to previous studies, the selected vertices can be any text fragments of any length, including words, phrases and even topics. The thesis uses each unique and stemmed word contained in the set of candidate phrases as vertices, as words are the basic units of language (Carstairs-McCarthy, 2017, p.4). For example, if the candidates are “natural language processing” and “keyword extraction”, then the vertices would include [“natur”, “languag”, “process”, “keyword” and “extract”].

(2) Edges and Weights

Similar to the original TextRank, SSRank defines the graph edges and weights using the number of pairwise co-occurrences within a sliding window of size w, normally set between 2 and 10.

Specifically, two vertices v_i and v_j are connected by an edge $(v_i, v_j) \in E$ if the words corresponding to these nodes co-occur within a window of w contiguous

tokens in the sentences of D. For example, the word "equation" appears within a number of three words of another word "linear" in the phrase "linear Diophantine equation", so it is treated as co-occurrence of window size three.

The weight of an edge $(v_i, v_j) \in E$ is then computed by counting the total number of co-occurrences of the two vertices within document *D*, which is an indicator of node proximity. For example, when the window size is set to 3, "compatibility" and "system" occur in the sample text twice, so the edge weight between the two vertices is 2.

(3) Directions

It should be noted that a graph can be constructed as directed or undirected. A directed graph can capture the successive order of a sequence, with an edge $v_i \rightarrow v_j$ meaning v_j appears behind v_i within the context window and not in front of, while an undirected graph is not attentive to this sequential order. Since Mihalcea and Tarau (2004) showed that the presence of direction in a textual graph does not significantly influence the ranking performance, this thesis uses the undirected version, which can also be regarded as a bidirectional graph.

A visualization of the graph is presented as follows with words mapped as points and their co-occurrence relationships mapped as lines or links. This is a simple homogeneous network that can be used to apprehend the connections and interactions among lexical units in the given input text, and draw out the key units in the network.

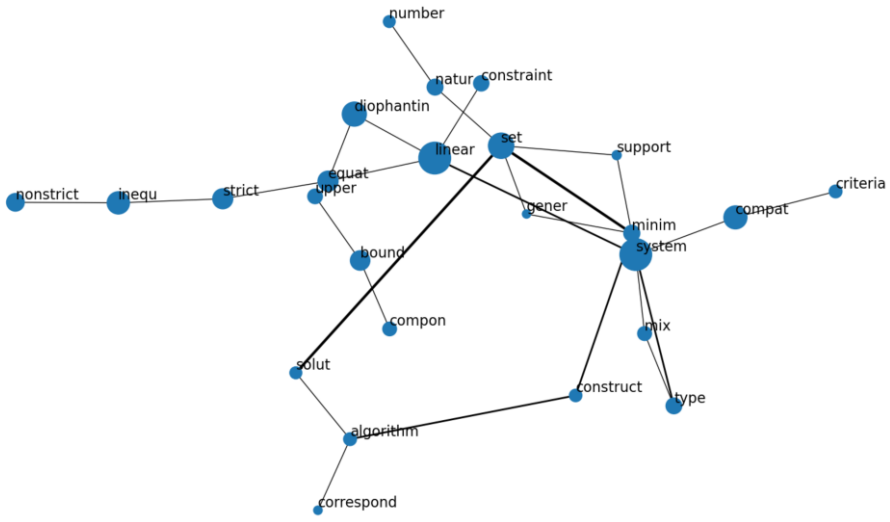

Fig 3.5: An edge-weighted graph built by SSRank with the line width indicating edge weight and node size indicating the importance score

3.2.2 Single Word Score Assignment

For score assignment, SSRank uses an edge-weighted PageRank algorithm to determine the influential nodes in the graph. It simulates a random traversal: at each step, it values the probability that the adjacent neighbors $Adj(v_i)$ terminates at the current node v_i, or moves to a random out-neighbor (Roshni & Unnikrishnan, 2021). Mathematically, for each vertex v_i, the structural score $S(v_i)$ is defined as the following equation:

$$S(v_i) = \begin{cases} \alpha * \left((1-d) + d * \sum_{v_j \in \mathrm{Adj}(v_i)} \frac{w_{ji}}{O(v_j)} S(v_j) \right), \text{if } V_i \in \text{ Title \& \& Upper} \\ (1-d) + d * \sum_{v_j \in \mathrm{Adj}(v_i)} \frac{w_{ji}}{O(v_j)} S(v_j), \text{otherwise} \end{cases} \quad (3.1)$$

where $O(v_j) = \sum_{v_k \in \mathrm{Adj}(v_j)} w_{jk}$, meaning the sum of all edge weights; w_{ji} means the weight of the edge (v_i, v_j), and d is the damping factor often set to 0.85.

Note that the structural score consists of two parts. (1) The first part is the graphic score $S(v_j)$ that constitutes the main part of the formula, which is computed iteratively and recursively from previous nodes. The graphic score depicts the connectiveness of a word within the input document, similar to the measure of degree. It indicates how words are distributed evenly in the text, repeated multiple times in different contexts. The damping factor d serves as an allocation ratio to balance the graphic part with the random uniform distribution. (2) The second part is the position score that highlights words appearing in the title or being uppercased by multiplying the score with a scaling factor α. Thus, words appear in special position or form can get higher importance.

SSRank starts by assigning an initial score to each word, which is computed as the frequency the word appears in the document. Then, the scores are iteratively updated according to the formula. The iterations continue until the number of iterations reaches 100, or the scores reaches a convergence point of $1e-5$, meaning that they stop

changing significantly from one iteration to the next. At this point, SSRank returns the score of each individual word, which can be used to compute the score of each candidate phrase.

3.2.3 Phrase Pooling

After the importance scores of single words are computed, the final step is to combine the scores of constituent words into the score of a single phrase. As aforementioned in the Literature Review chapter, the most commonly taken measure is to simply sum the scores of all component words in a phrase. But this measure is only preferable when extracting longer keywords, and thus suffers from the length problem.

To get a better understanding of the effect keyword length can have on the probability that a candidate can be chosen as keywords, the author plotted the distribution of keyword lengths in different datasets (see Appendix C for more information). For all datasets, the length frequency for scientific keywords peaks at 2, and drastically declines until reaches 8. This phenomenon can be roughly approximated with the general Zipf 's law (1949), which states that the frequency of occurrence of a word in a large corpus of text is inversely proportional to its rank.

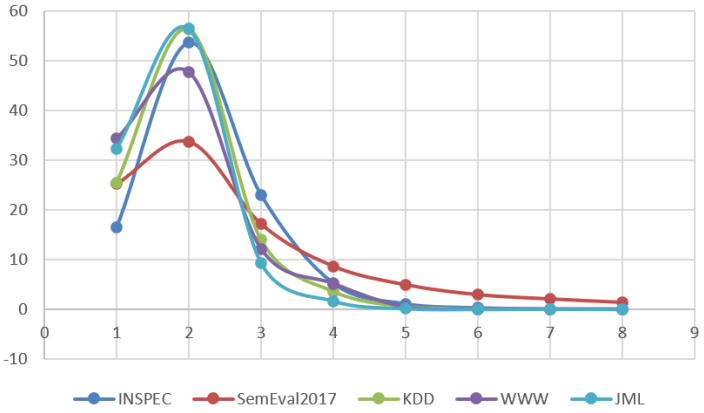

Fig 3.6: The relationship between the proportion (%) of keywords and their n-gram length in each dataset

Thus, to model the influence of length in keyword assignment, the author innovatively uses a fit of the power-law (Arampatzis & Kamps, 2008) when length is

greater than 1 to pool the component scores of each candidate phrase into an aggregated whole:

$$Score(p) = \frac{\sum_{i=1}^{n} w_i \text{ (n-1)}}{e^{n-2}} \tag{3.2}$$

$$\text{where } w_1 > w_2 > \cdots > w_n, \ and \ n > 1, \ n = len(p).$$

Using this formula of non-linear phrase pooling, bigrams and trigrams are preferred, while unigrams and sequences of length longer than four are less preferred. In this way, the length problem can be solved. The final results of structural scoring of candidates extracted from the sample text are shown in Fig 3.5.

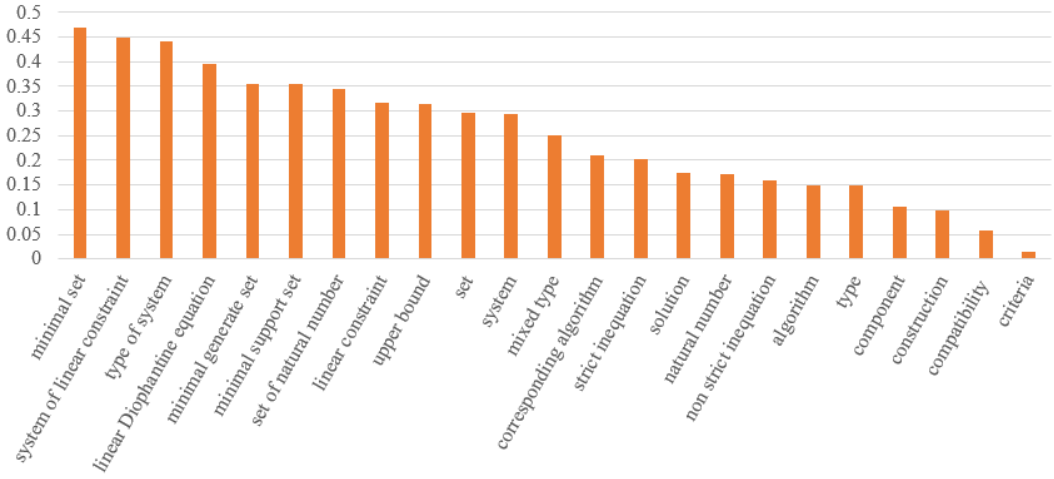

Fig 3.7: The result of structural scoring of candidates extracted from the sample text

3.3 Semantic Clustering

The third step is to group candidates according to their semantic distance so that candidates in one group is as synonymous to each other as possible. As described in the previous section, candidates are ranked according to their structural score, which represents their distributional and positional significance in the text. However, there is high probability that semantically equivalent or related phrases like “nonstrict inequation” and “strict inequation” may rank close to each other and leads to the problem of redundancy. Other cases of redundancy occur when one phrase includes another, like “system of linear constraint” and “linear constraint”; or when two phrases contain some overlapping words, like “minimal generative set” and “minimal

supporting set". To tackle this problem, the thesis utilizes semantic clustering to automatically assemble similar phrases into one group, and rank these topical groups to select each representative keyword from each high-ranking group.

The study proposes two kinds of clustering methods: (1) The first is a bottom-down approach using HAC clustering with a modified edit distance as the distance metric. (2) The second is a top-down approach using K-means clustering and word embeddings. Both clustering methods can be implemented with the Python machine learning package scikit-learn[10].

3.3.1 Distance Computation

(1) Overlap Distance

As aforementioned, HAC uses distance as a measure of cluster difference to aggregates elements that are close to each other into one group. The distance metric used in TopicRank is word overlap, which considers two keyphrase candidates as similar if with at least 25% overlapping words. However, this simple mapping of word appearance or absence is not enough to discriminate phrase pairs like "strict equation - strict inequation" and "strict equation - strict type".

To explain this more clearly, let's consider the word overlap between a phrase pair p_j and p_j:

$$overlapDist(p_i, p_j) = 1 - \frac{numOverlap}{max\left(num(p_i),\ num(p_j)\right)} \tag{3.3}$$

which is the number of words that cooccur in both phrases divided by the number of words in the longer phrase. For example, word pair "strict equation" and "type" receive a minimum similarity score of 0, while "strict equation" and "strict type" receive a relatively better score of 0.5. However, word pair "strict equation" and "strict inequation" receive an equal score of 0.5, which is unreasonable since their actual semantic gap is smaller as "inequation" is apparently closer to "equation" than "type".

[10] https://scikit-learn.org/

(2) Edit Distance

Given the problem described above, the author tried to use edit distance to measure phrase proximity. Edit distance is also called Levenshtein distance (Levenshtein, 1965), which is used to measure the difference between two sequences. The metric computes the distance as the minimum number of single-character editing operations needed to convert one phrase to another, including insertions, deletions, or substitutions of letters. The Levenshtein distance between two phrases $lev(p_i, p_j)$ can be efficiently calculated through dynamic programming. Then the edit distance score between two phrases p_i, p_j can be computed as:

$$editDist(p_i, p_j) = \frac{lev(p_i, p_j)}{max\left(len(p_i), len(p_j)\right)} \tag{3.4}$$

Taking the above phrase pairs as examples, the Levenshtein distance between "strict equation" and "strict inequation" is 2 and the longer length of the two is 17, so the distance score is $2/17 = 0.118$. Following the same calculation, the edit distance between "strict equation" and "strict type" is $7/15 = 0.467$, which means that the distance of the first pair is smaller than the second pair. This suggests the effectiveness of edit distance in quantifying phrase dissimilarities while taking into account duplications within words. However, another problem arises as the word pair "component" and "compatibility" that are very similar in form receive a counterintuitive high score of 0.615, even though they are semantically unrelated!

(3) Pseudo One-hot Encoding Cosine

To combine the merits of word overlap and edit distance and overcome their drawbacks, the author proposes another method of distance calculation which is called Pseudo One-hot Encoding (POE) Cosine, which takes morpheme overlap into consideration, but do not make it too excessive. The distance metric is based on the construction of pseudo One-hot vectors of phrases, a modified version of the famous On-hot encoding representation of data. The concrete steps for calculation are described as follows.

First, all unique words contained in the set of candidate phrases are taken to form a vocabulary of n words. Then, a matrix A of size (m, n) is constructed where m is the number of candidates. Each element $A[i][j]$ in the matrix is assigned as 1 if the candidate phrase p_i includes the word w_j in the vocabulary, and 0 if not. If the letters of w_j has a continuous sequence overlap of over 50% with the letters of a word in p_i, they are considered to be morphological variants of the same stem, like "equation" and "inequation", and $A[i][j]$ is assigned as the number of overlapping letters divided by the total number of letters in the longer word as shown in Eq. (3.3).

	strict	nonstrict	inequation	equation	linear	Diophantine	component	…
strict inequation	1	0.667	1	0.8	0	0	0	…
nonstrict inequation	0.667	1	1	0.8	0	0	0	…
linear Diophantine equation	0	0	0.8	1	1	1	0	…
component	0	0	0	0	0	0	1	…
compatibility	0	0	0	0	0	0	0	...

Table 3.4: Vector Representation of Phrases

As shown in Table 3.4, each row represents a candidate phrase and each column is the unique vocabulary. In this way, each candidate p_i is converted into a one-dimensional sparse vector $V(p_i)$ with values ranging between 0 and 1. Finally, the cosine distance between two phrase vectors is simply calculated as:

$$Cosine(p_1, p_2) = 1 - \frac{\sum_{k=1}^{n} V(p_1)_k V(p_2)_k}{\sqrt{\sum_{k=1}^{n} V(p_1)_k^2}\sqrt{\sum_{k=1}^{n} V(p_2)_k^2}} \tag{3.5}$$

For example, the distance between the phrase pair "strict inequation" and "nonstrict inequation" is 0.036, the distance between "strict inequation" and "linear Diophantine equation" is 0.523, and the distance between "component" and "compatibility" is 1. In this way, the problem of inner overlapping morphemes between component words can be better solved.

(4) Phrase Embedding Cosine

The fourth method is to compute cosine distance between phrases using averaged

word embeddings. Since word embeddings can only vectorize words instead of phrases, the study averages the semantic vectors of the constituent words of each phrase to get its vector embedding, under the theoretical support of the compositionality of meaning. This method is proved to be useful and can achieve approximate results to document embedding methods like Doc2vec and sentence-BERT which are computationally more expensive.

Specifically, the author uses the pretrained glove vectors "glove-wiki-gigaword-300" loaded with Gensim[11] to convert words into vectors and get the averaged vector for each phrase. Then, Word Embedding (WE) cosine distance is computed using exactly the same formula as Eq. (3.5).

However, unfortunately, scholarly text is often suffused with complex terminology that is rarely used in daily language, and these terms can differ in different domains. For domain-specific terms like "nonstrict inequation", we cannot get their embedding since they contain uncommon words that are absent from the vocabulary of the pretrained model, and thus cannot compute their WE cosine distance. For example in this case, both "nonstrict" and "inequation" are not included in the vocabulary. This makes WE cosine inapplicable in many scholarly circumstances.

To compare these four different distance metrics, the thesis uses some daily toy phrases as an example, and shows their results in Table 3.6. As we can see, overlap distance cannot capture some semantic connections like "happy" and "unhappy", "child" and "children", while edit distance and POE cosine are capable of finding these relations. WE cosine performs best is determining such similarity, as it can even relate the meaning between "child" and "people".

Phrase_1	Phrase_2	Overlap Distance	Edit Distance	POE Cosine	PE Cosine
happy child	happy child	0	0	0	0
	unhappy children	1	0.313	0.077	0.251
	happy	0.5	0.545	0.278	0.229
	child	0.5	0.545	0.308	0.158
	unhappy	1	0.727	0.317	0.546

[11] https://radimrehurek.com/gensim/auto_examples/index.html

people	1	0.818	1	0.534
type	1	0.909	1	0.763

Table 3.5: Comparison of different distance metrics

3.3.2 Hierarchical Agglomerative Clustering

Given the distance measure described in the previous step, SSRank then clusters these phrases accordingly using Hierarchical Agglomerative Clustering (HAC).

HAC clustering is an unsupervised machine learning algorithm that seeks to build a hierarchy of clusters from the sample data by comparing their numerical distance. The algorithm operates in a bottom-up way: it initially considers each candidate as a singleton cluster, and then recursively merges (or agglomerate) pairs of clusters that are most adjacent to each other until all clusters have been combined into one gigantic cluster that contains all candidates. As the algorithm progresses upward from each individual phrase in the bottom layer to growingly larger clusters and finally to the top node, a “hierarchy or exhaustive partition” of the whole candidate set is created (Schütze et al., 2008). By adjusting the distance threshold or number of clusters, we can get any kinds of clustering results.

The specific procedures go in the following manner. First, all candidate phrases are assigned to its own cluster, forming the root nodes in the hierarchy. Second, the two closest nodes are merged into a new node according to the average linkage, meaning average distance of the node cluster. Finally, the process of node merging is repeated until the distance is greater than a threshold of 0.75. That is, candidates are aggregated into the same group with a stem similarity of more than 25%.

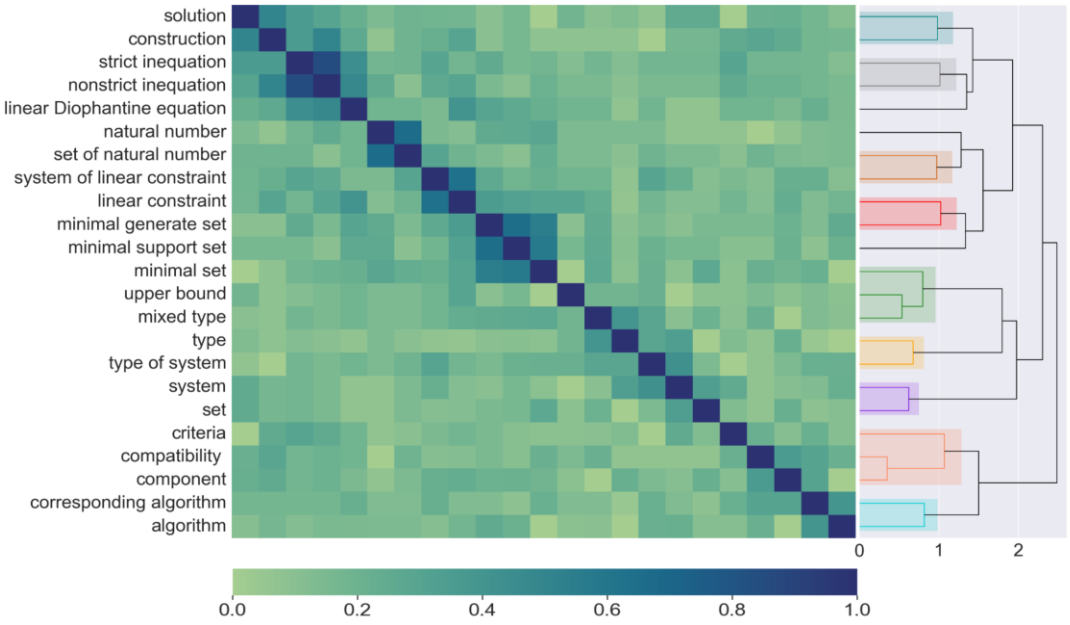

Fig 3.6: HAC result of the sample text using edit distance with the heatmap (left) showing candidate distance and the dendrogram (right) showing clusters with different colors

Fig 3.6 shows the process of HAC clustering using edit distance. The algorithm outputs a dendrogram (right) that plots the taxonomic relationship among candidates, which "allows us to reconstruct the history of merges that resulted in the depicted clustering" (Schütze et al., 2008).

Any of the above four distance metrics can be used, and the best result comes with POE cosine: (1) minimal set, minimal generate set, minimal support set, set; (2) set of natural number, natural number; (3) system of linear constraint, system, type of system; (4) linear Diophantine equation, linear constraint; (5) nonstrict inequation, strict inequation; (5) corresponding algorithm, algorithm; (6) mixed type, type; (7) component; (8) upper bound; (9) construction; (10) compatibility; (11) criteria; (12) solution. This shows that HAC clustering correctly groups phrases that are semantically close.

Overall, HAC is a flexible method for semantic clustering as it does not always require a prespecified number of clusters as input, and can use any valid measure of distance. Compared to flat clustering, it can return an informative hierarchical structured set of clusters. But it can be computationally expensive for large datasets.

3.3.3 K-means Clustering

K-means clustering is a simple partitional clustering approach that divide data into groups so that within-group points are close to each other and outside-group points are distant. Despite its computational efficiency, it only works on vectorized data, so the thesis uses averaged word embeddings for distance measurement.

The author implements K-means clustering with the following steps: First, a prespecified number of k clusters are randomly initialized. Second, each phrase is assigned to the closest cluster centroid by calculating the cosine distance between each phrase embedding and each centroid. Then, the centroid of each cluster is recalculated by taking the mean of all the phrase embeddings assigned to it. Finally, the centroids are continuingly recomputed until the centroids stop to relocate.

Fig 3.7 shows the result of K-means clustering, with different colors indicating different groups. Since high-dimensional vectors are hard to view in a plane, the author uses the dimensionality reduction technique T-SNE to project the vectors into 2 dimensions for visualization. It is satisfactory to see that semantically close phrases like "set of natural number" and "natural number" are aggregated into one group, and even the phrase "upper bound" is also included in the same group! This may be due to their common relevance to the specific domain of mathematics.

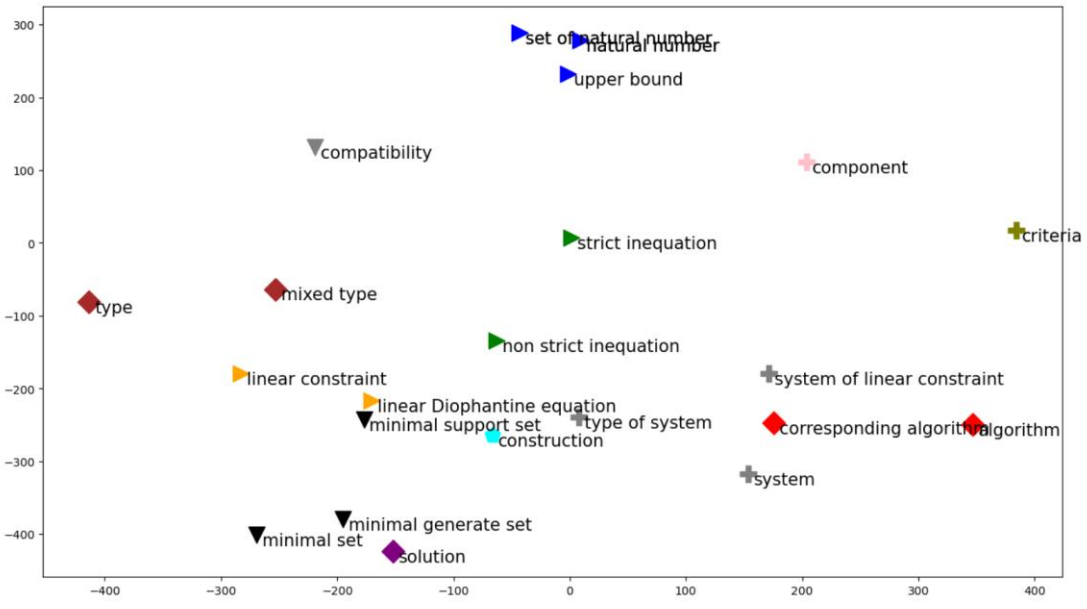

Fig 3.8: K-means result of the sample text using PE cosine for distance measurement. Candidates are anchored in the plot with different shapes and colors indicating their clusters.

Since there are overall 23 candidates and the number of clusters is set to n/2, the resulting clusters for the sample text are: (1) minimal set, minimal generate set, minimal support set; (2) set of natural number, natural number, upper bound; (3) system, system of linear constraint, type of system; (4) linear Diophantine equation, linear constraint,; (5) nonstrict inequation, strict inequation; (6) corresponding algorithm, algorithm; (7) mixed type, type; (8) component; (9) solution; (10) construction, (11) compatibility, (12) criteria.

Overall, K-means is good at distinguishing nuance semantic relatedness between candidates. But the result is not deterministic and can be influenced by different ways of initialization of centroids.

3.4 Keyword Selection

After the candidates are assigned structural scores and clustered into different semantic groups, the final step is to select key phrases from these different topical groups. Intuitively, the simple and direct solution is to choose the highest ranking keyword from each group. However, a series of problems may arise.

First, keywords are likely to appear in the same semantic group, for example “linear constraint” and “linear Diophantine equation” as shown in the sample text. It can be observed that these two candidates are not as semantically equivalent as another synonymous pair “minimal set” and “minimal generating set”. Put differently, the first two phrases are more semantically remote and thus should not be considered as “the same phrase” that convey almost the same meaning. Brutely selecting the most promising one and discarding all other phrases in the group do not seem to be the optimal choice.

Second, not all clusters manifest good properties, i.e. similarity and closeness between observations in a single cluster as well as condensed and relevant information in the topic that aligns with the theme of the document. In the illustrative example, the

cluster consisting of the two “strict inequation” and “nonstrict inequation” is apparently more compact and informative than the cluster composed of “mixed type” and “type”. To include every cluster in an equal manner is likely to bring mistakes and noises, which propels the need to rank and select the best clusters.

Thus, the thesis proposes a more flexible algorithm of output decision making called keyword chaining. The idea is to first sort clusters in proper order and then chain the best elements of the best clusters into a list of keywords which constitute the final output, or select the top *k* from this list if there exists a prespecified number *k*.

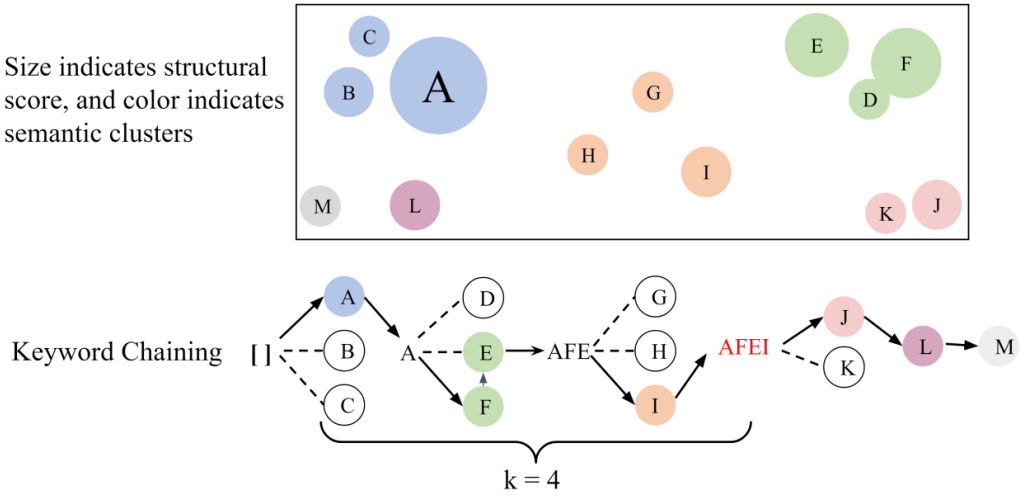

Fig 3.7: The clusters plotted in different colors (upper) and the process of keyword chaining (lower)

Specifically, it involves the following practical steps: (1) Sort clusters in ascending order according to its distance from the centroid of all clusters as well as average within-group distance:

$$clusterRank(c_i) = \lambda \cdot dist(c_i, centroid) + (1-\lambda)\underset{c_j \in C}{meandist}(c_i, c_j)$$

Normally, the λ is chosen as 0.5, which balances cluster compactness and informativeness. (2) Each time pick the highest ranking keywords from the topmost cluster that score higher than any keyword from its subsequent neighboring cluster. (3) Continue the process until the stack of clusters are exhausted or the keywords reach the predefined maximum number *k*.

The final result for the sample text is presented in Table 3.6. We can see that all keywords generated by SSRank partially or exactly match with human ground truths, and no keywords are missed by SSRank.

Keywords assigned by SSRank: minimal set; set of natural number; system of linear constraint; linear Diophantine equation; upper bound; strict inequation.
Keywords assigned by human annotators: linear constraints; linear Diophantine equations; minimal generating sets; nonstrict inequations; set of natural numbers; strict inequations; upper bounds

Table 3.6 Keywords extracted by SSRank compared with those assigned by human annotators. Blue color indicating partial matching and red color indicating extract matching.

3.5 Chapter Summary

This chapter elaborately presents the four procedures of the innovatively devised keyword extraction algorithm Structural and Semantic Rank (SSRank), namely candidate formation, structural scoring, semantic clustering and keyword selection. The workflow of SSRank is summarized in the following flowchart.

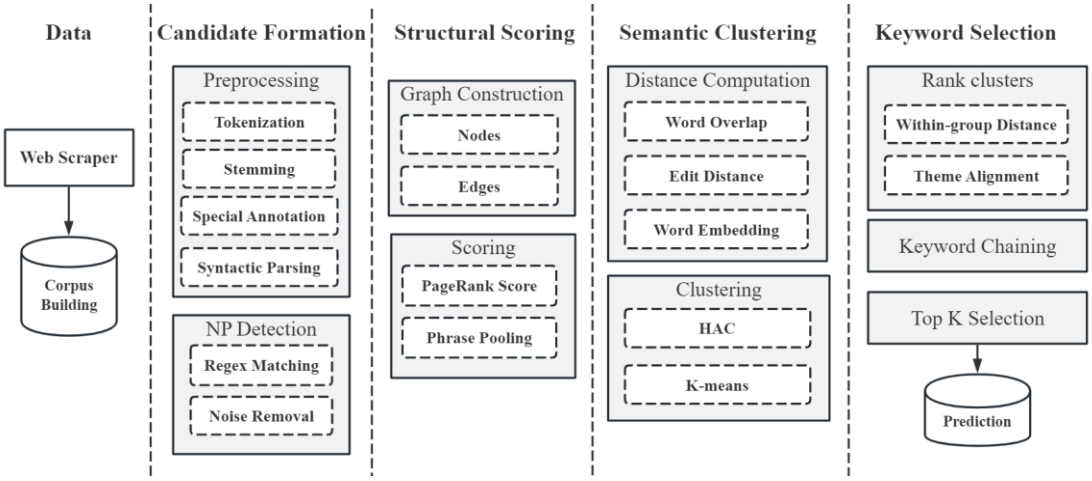

Fig 3.9: Flowchart of SSRank keyword extraction procedures

Given a document from existing dataset or any corpus built using web scrapping (the process of corpus building will be shown in Chapter 4) with noise removed and text preprocessed, SSRank first extracts candidate phrases via a carefully designed NP

detector based on regex matching of POS and dependency tags. Then, on the one hand, unique words taken from the candidate phrases form the nodes of the textual graph, and the count of co-occurrence pairs form the edges, which are combined to compute an importance score for each word, which are pooled to produce the structural score of each phrase. On the other hand, the phrases are clustered into different semantic groups using HAC clustering or K-means with four different distance metrics to enhance topical coverage and diversity. These groups are ranked according to within-group distance and degree of topical centrality, which selects the top groups that have more inner duplications and outer theme alignment with the original document. Finally, the top-ranking phrases from the top-ranking groups are chained together to represent the final set of keywords.

Chapter 4 Experiment and Result

To verify the effectiveness of SSRank, the thesis conducted two targeted experiments. The first experiment implemented ablation studies for SSRank, which aimed to understand the contribution of individual components of SSRank to its overall performance, and to determine the optimal parameter values and boost algorithmic performance in each process. The second experiment evaluated the overall effectiveness of SSRank compared with three baseline methods including statistical, graph-based and language-model based approaches over five datasets varied with different dataset sizes and domains.

4.1 Experiment Setup

4.1.1 Building the JML corpus

One contribution of the thesis is the creation of the English scientific text corpus for linguistics called JML. As reviewed in Chapter 2, existing datasets for keyword extraction have covered scientific areas like science, technology, engineering, and mathematics (STEM) and biomedicine, with subjects belonging to humanities and social sciences largely missing and inaccessible. This lack of dataset diversity motivates the building of JML to foster information extraction research and technologies in non-tech areas. The detailed procedures for building the corpus are described as follows.

(1) Journal selection. To ensure the high quality of text and keyword annotations, it is important to gather research papers from the most authoritative journals. The author first found a ranked list of linguistics journals from the platform Scimago Journal & Country Rank (SJCR), and by scanning through each journal's webpage, recorded their impact factors and labeled them as with or without publicly available keywords (the list of journals are already shown in Table 1.2). Then, the author filtered through this list and selected the *Journal of Memory and Language* due to its explicit presentation of keywords.

(2) Webpage scraping. Aided by the fast and powerful Python web crawling tool Scrapy, the author was able to write a scraper to collect all latest research papers in 2012-2021 from the websites of the above selected journals. Structured data were obtained through CSS elements or HTML metadata with tags like “title” “abstract” “citation_keywords” from the received HTTP response, and saved in JSON files.

(3) Text processing. Given the collection of scraped text, the final step is to organize it into clean and accurate datasets. This includes data cleansing procedures like removing items with missing information or errors, eliminating duplicates and irrelevances, as well as data balancing and formatting.

The final JML corpus consists of 572 linguistic papers with abstract, title and keywords, but one can obtain much more if conditions allow. It can be a useful dataset for benchmarking keyword extraction and generation techniques in non-stem domains. And the creation procedures for JML can be applied to any scientific literature in any field of study.

4.1.2 Dataset Statistics and Preparation

Apart from the JML dataset, the other four datasets used are INSPEC, SemEval2017, WWW and KDD, which are the most popular and freely available corpora comprised of scientific abstracts and corresponding keywords. Among them, the SemEval2017 dataset is specially annotated for the task keyword extraction from scientific documents, while the other four are built from the journal database with author-specified keywords, which may include more noise and bias. Thus, the experiment takes the results of SemEval2017 dataset as the primal reference.

Since information about these datasets is already given in the Literature Review chapter, this section focuses on an overview of some most revealing dataset statistics that concerns corpus composition, length and other information, as shown in Table 4.1. In the table, #kps means the average number of keyphrases per document, #docs means the number of documents in the entire dataset, #tokens means the average number of tokens per document, and domain means the scientific subject that the dataset mainly

concerns about. The other two statistical indexes are presence rate, which means the ratio of gold references that exist in the document, and extraction rate, which means the ratio of keyphrases to the total number of tokens per doc.

From the statistics we can observe that the SemEval2017 dataset possess the greatest number of keyphrases per document and 99.3% of them are directly taken from the original text, while the INSPEC dataset stands out by its hugest volume of documents. Other statistics suggest that the JML dataset has the longest document length, whereas its average number of keyphrases is only less than 5, yielding an extraction rate of 2.7% only, meaning that only 2.7% of the tokens in a scientific abstract can be chosen as keyphrases.

Datasets	#Kps	Presence Rate	#Docs	#Tokens	Extraction Rate	Domain
INSPEC	14.1	61.4%	2000	124.4	11.3%	Engineering & physics
SemEval2017	17.3	99.3%	493	168.9	10.2%	CS, MS & physics
KDD	4.1	36.4%	755	74.1	5.5%	CS
WWW	4.8	36.8%	1330	82	5.9%	CS
JML	4.9	57.2%	572	183.9	2.7%	Linguistics

Table 4.1: Statistics of available datasets for keyword extraction. Blue shading highlights the largest value in each column.

Since SSRank is completely unsupervised, the thesis randomly sampled 400[12] pieces of documents from each dataset to form the testing set, and no training or development set is demanded. The thesis also aggregated the ensemble of 100 documents randomly picked from each dataset into a single blended collection of 500 documents. This mixed collection can help us to get a cross-domain unified overview of the performance of SSRank.

[12] The number 400 is a balance between the minimum size of the five datasets and validity.

4.1.3 Two Metrics: EM@F and PM@F

For evaluation, the thesis uses Precision, Recall and F-score as the main performance metrics. Specifically for all documents in each evaluation set, the total number of algorithmic assigned terms, gold references, and true positives are calculated to get the metric performance for the whole dataset. Since some humanly assigned keywords are not present in the document, meaning they do not explicitly appear in the text (Mihalcea, 2004), these absent keywords are removed from the list of gold references to achieve a recall of 100%.

Another problem occurs when some gold keyphrase partially include the predicted keyphrase or the other way round, as in the case "predictive control" and "model predictive control". Thus, apart from the traditional metrics which requires that the predictions and the gold references should be identical, in this thesis called *exact match*, the thesis introduces another kind of evaluation method called *partial match*, which retrieves the number of true positives as a list of gold keyphrases that are partially matched with the predictions.

$$\text{Precision@PM} = \frac{\left|PM(Y_{pred}, Y_{gold})\right|}{\left|Y_{pred}\right|}$$

$$\text{Recall@PM} = \frac{\left|\text{PM}(Y_{\text{pred}}, Y_{\text{gold}})\right|}{\left|Y_{\text{gold}}\right|}$$

$$F\text{-score} = \frac{2\ \text{Precision@PM} * \text{Recall @PM}}{\text{Precision@PM} + \text{Recall@PM}}$$

Table 4.2 shows an example of score computation for exact match and partial match, whose sole difference lies in the computation of true positives. Partial match scores are always equal to or higher than exact match scores as it counts more true positives in.

Examples	Exact Match			Partial Match		
	P	R	F	P	R	F
Pred: low frequency form, morphological change, analogical basis, Greek data Gold: analogy, Greek, morphological change, morphology, token frequency	0.25	0.2	0.22	0.75	0.6	0.67
Pred: telecom, insider investment, industry insider, company stock Gold: telecom industry, insider investment, telecommunication	0.25	0.33	0.29	0.5	0.67	0.57

Table 4.2: Examples of F@EM and F@PM score computation

4.2 Results of Ablation Studies

To understand the contribution of individual components of SSRank to its overall performance, and to determine the optimal parameter values and boost algorithmic performance, the author conducts several ablation studies. Specifically, (1) for the candidate formation process, the accuracy of the two kinds of NP detector proposed by the author is tested with an approximation of Recall on AKE datasets. (2) For structural scoring, the author conducted parameter tuning for the positional PageRank, including the damping factor and window size. (3) For semantic clustering, the author investigated the effect of different distance metrics and clustering methods on the performance of SSRank, and compared the performance of SSRank with and without semantic clustering to understand its contribution to the system.

4.2.1 The Performance of the NP Detector

In the candidate formation process, the author devises two kinds of NP detector: the first one is based on regex matching of POS labels and implemented with the NLTK RegexpParser, while the second is a hybrid one based on the spaCy dependency matcher that utilizes both POS and dependency tags as well as the exact text.

In the first ablation study, the author tested their accuracy and compared with two

publicly used noun phrase extraction tools: the spaCy noun_chunk[13] and TexBlob[14]. Since it is not easy to find a dataset specifically built for NP detection, the author uses the measure of Recall to approximate the accuracy, under the assumption that all keywords should be noun phrases and thus correctly retrieved by the NP detector. The results across the five datasets are shown in Table 4.3.

	DM-detector	RP-detector	noun_chunk	TexBlob
INSPEC	0.919	0.699	0.460	0.520
SemEval2017	0.721	0.592	0.315	0.333
KDD	0.893	0.755	0.715	0.678
WWW	0.837	0.737	0.698	0.687
JML	0.900	0.778	0.671	0.515

Table 4.3: The recall of the proposed NP detector compared with two existing NP extraction tools

The results show that both the dependency match based NP detector (DM-detector) and the RegexpParser based NP detector significantly outperform the existing tools. Among all, the DM-detector can identify candidates with the highest accuracy over every dataset, obtaining a score of 0.919 on INSPEC and 0.900 on JML; RP-detector gets a lower score due to its non-overlapping characteristics, but it still gets 0.778 on JML. These findings highlight the quite satisfactory performance of the NP detector component in the SSRank system, which guarantees its superb ability to extract grammatically sound noun phrases.

4.2.2 Parameter Tuning for Structural Scoring

(1) Damping Factor

The first parameter under examination is the damping factor *d*, which is an indispensable variable in formulating the equation of structural scoring. It plays a

[13] https://spacy.io/usage/linguistic-features
[14] https://textblob.readthedocs.io/en/dev/

critical role in deciding how the nodes in the graph do teleportation and guaranteeing the final convergence of the algorithm. Specifically, it means the probability each node in the graph can be reached through other nodes directly connected to it, the value of which can thus range from 0 to 1. When *d* is 0, the graph part of the traversing process is annihilated, resulting in the trivial uniform distribution. As *d* goes to 1, the graph part becomes increasingly significant. Previous literature reveals that the generally recommended range of the damping factor is 0.8 to 0.9 (Kazemi et al., 2020), with 0.85 being the typical value (Rajput et al., 2020).

The following figure plots the influence of the damping factor with the horizontal axis represents the parameter and the vertical axis represents Precision, Recall and F-Score. As suggested in Fig 4.1, the choice of damping factor has great impact on the performance of SSRank for keyword extraction across various datasets. Although the effect of damping factor is nearly negligible, we do see an interesting upward trend in nearly every curve for all three metrics especially when d is about to reach the maximum value 0.9. This phenomenon is particularly apparent for the KDD dataset, whose precision reaches from 0.241 to 0.245 and recall rises from 0.351 to 0.363 as the factor d continues to increase. Overall, the F-score of the aggregation of all datasets also shows such metric growth from 0.367 to 0.369.

In summary, SSRank achieved the greatest results with a damping factor of 0.9 which gives the maximum to the score produced by the graph structure.

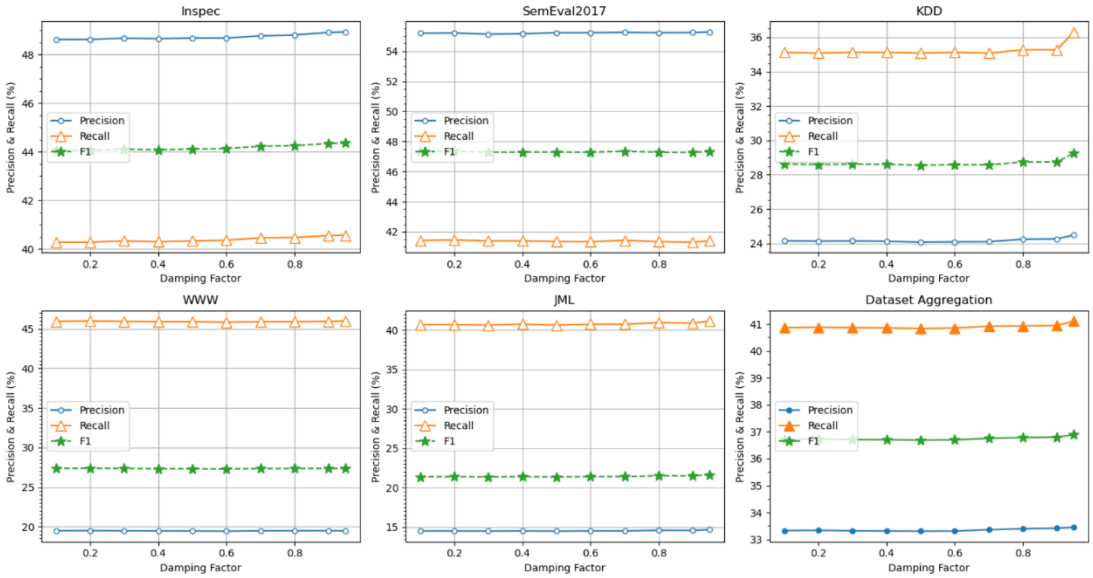

Fig 4.1: Results of Precision and Recall on changing damping factor

(2) Co-occurrence Window Size

Co-occurrence window size w is another important parameter, as SSRank constructs the graph edges and weights using the number of pairwise co-occurrences within a sliding window. Concretely, "co-occurrence" means two words co-occur in the distance less than a certain number of words, and the "window size" means the number of words that appear between these two words under examination. For example, the word "equation" appears within a number of three words of another word "linear" in the phrase "linear Diophantine equation", so it is treated as co-occurrence of window size three. It should be noted that in SSRank, only words that exist in the same sentence are considered to be adjacent pairs, which means that SSRank only captures structural relations within the sentence as an inner-connective syntactic unit. According to previous studies, the window size is normally set between 2 and 10**.**

As suggest in Fig 4.2, results show that SSRank is again very robust under changing window sizes. On the one hand, there is an evident uplift of precision scores for various datasets when the window size goes from 2 to 3, and then the precision begins to level off. For example, the precision for INSPEC changes sharply from 0.241 to the highest score 0.25. This may be because that a window size of 2 is too small to obtain a complete textual graph and enlarging the window size allows SSRank to discover more relationships between words and thus results in better modeling of the text structure. On the other hand, there is a mutual decline in the recall scores when the window size increases from 2 to 3 and some very slight fluctuations after that. The reason behind might be that the expansion of window size gives rise to the possibility for SSRank to over-score irrelevant words and omitting real keywords. The final F-score results for the assemblage of all datasets show that SSRank exhibits the best performance when setting co-occurrence window size w=3, a number that can properly locate contiguous words. This finding is in accordance with previous literature.

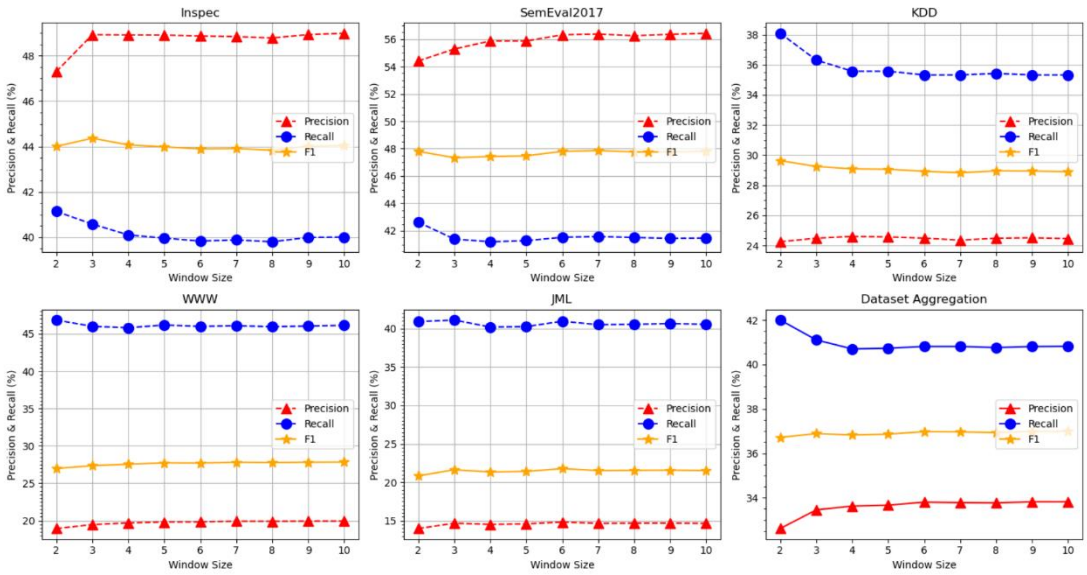

Fig 4.2: Results of Precision and Recall on changing window size

4.2.3 Parameter Tuning for Semantic Clustering

For the third ablation study, the author investigated the effect of different distance metrics and clustering methods on the performance of SSRank. Specifically, the author compared the influence of HAC clustering on different distance metrics, i.e. word overlap distance (HAC-Overlap), edit distance (HAC-ED), POE cosine (HAC-POE), and WE cosine (HAC-WE), with that of K-means clustering on WE cosine.

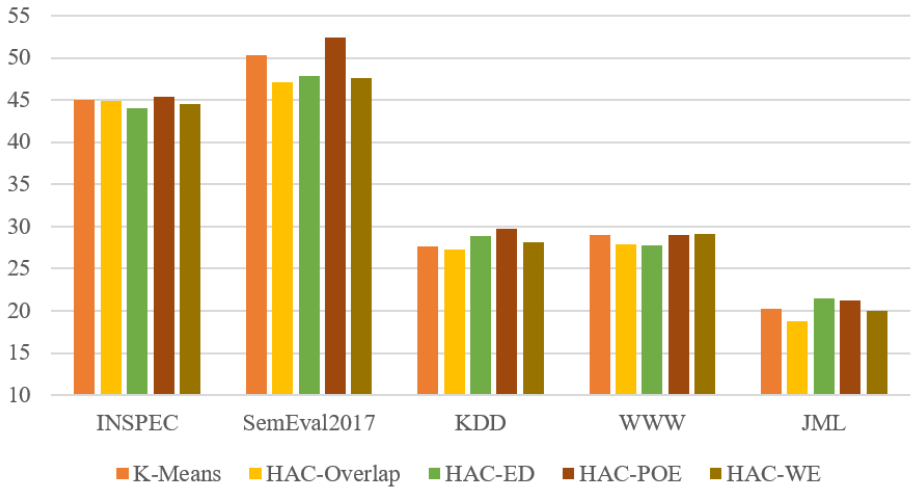

Fig 4.3: Results of Precision and Recall on changing damping factor

The results in Fig 4.3 show that the choice of distance metric and clustering method has a considerable impact on the performance of SSRank. Generally, the HAC

clustering method performs almost equally well as K-means despite its use of simple distance measures without the assistance of language models. This may be due to the encounter of many out of vocabulary words in academic terms when vectorization of text was performed, which lost some information and thus affected the result.

For individual datasets, HAC performs better on KDD, WWW and JML, while K-Means is better on INSPEC and SemEval2017. Furthermore, the study revealed that the POE cosine distance metric outperformed the other three distance metrics in most cases. The most outstanding performance of HAC-POE comes with the SemEval2017 dataset, with a F@PM of 0.5528, showing its great potential in similarity measurement. The worst performance comes with HAC-Overlap, the most naïve design, with a score as low as 0.1883 on the JML dataset. On average over all datasets, HAC-POE raises the F-score of HAC-Overlap with around 0.023 points, an increase of 7%.

To understand how the semantic clustering part contributes to its overall performance, the author also compared the performance of the complete system with that of SSRank with semantic clustering removed, as shown in Table 4.4. On average, the semantic clustering component raises the F@EM with around 0.0133 points, an increase of 10.68%; and F@EM with around 0.0303 points, an increase of 8.9%. The findings highlight the overall improvement of the performance after adding the semantic clustering component.

	S-Rank		SSRank	
	F@EM	F@PM	F@EM	F@PM
INSPEC	0.2344	0.4501	0.25	0.4893
SemEval2017	0.2212	0.503	0.2588	0.5528
KDD	0.054	0.2663	0.058	0.2849
WWW	0.051	0.2803	0.0591	0.2996
JML	0.0638	0.202	0.0652	0.2267

Table 4.4: Results of S-Rank (without semantic clustering) and SSRank

4.3 Results of Comparative Study

In order to compare the effectiveness of SSRank against that of other currently existing AKE methods, the author then carried out a comparative study with three state-

of-the-art baselines. Each of the three is a representative of one branch of unsupervised approaches, i.e. statistical (YAKE!), graph-based (TextRank) and language-model-based (KeyBERT). These baselines are also chosen for their free availability, open-source and public popularity. For SSRank, the author set the damping factor d=0.9, window size w=3 and used HAC-POE for semantic clustering as these parameters achieved the best performance in the tuning experiment.

4.3.1 Comparison of General performance

Table 4.5 lists the results of F@EM and F@PM for all four algorithms over five datasets. Although it can be observed that keyword extraction suffers from poor effectiveness as the results show generally low F-score, SSRank outperforms other unsupervised AKE methods most of the time. The best results occur with the SemEval2017 dataset with F@EM=0.2588 and F@PM=0.5528 and the INSPEC dataset with F@EM=0.25 and F@PM=0.4893. On the opposite, the results for KDD, WWW and JML are relatively low, with the worst cases being F@EM=0.058 on KDD and F@PM=0.1467 on JML. This happens uncoincidentally as they are among the datasets with the least presence rate and extraction rate, making keyword extraction extremely difficult.

	TextRank		YAKE!		KeyBERT		SSRank	
Dataset	F@EM	F@PM	F@EM	F@PM	F@EM	F@PM	F@EM	F@PM
INSPEC	0.1637	0.3475	0.112	0.4102	0.1107	0.2721	0.25	0.4893
SemEval2017	0.1806	0.434	0.1192	0.5295	0.1324	0.351	0.2588	0.5528
KDD	0.057	0.1622	0.0633	0.2253	0.0037	0.1324	0.058	0.2449
WWW	0.0574	0.1629	0.0447	0.2377	0.0506	0.1902	0.0591	0.1946
JML	0.0641	0.1212	0.0456	0.2084	0.0427	0.0874	0.0652	0.1467

Table 4.5 F@EM and F@PM performance of various AKE methods on five benchmark datasets

While SSRank defeated TextRank and KeyBERT with F@EM and F@PM on all datasets, it got the second place with F@EM on KDD and F@PM on WWW and JML. Making a close comparison of the two metrics upon these two datasets, we can find that

SSRank and YAKE! performed nearly equally well as each won three metrics out of six: SSRank got a higher F@PM=0.2449 on KDD and higher F@EM=0.0591 on WWW and F@EM=0.0652 on JML.

The precision and recall results shown in Fig 4.4 indicate that the precision of SSRank is higher than any algorithm on any dataset. But the recall was relatively lower in some cases. This may be due to SSRank's over emphasis of diversity and uniqueness of keyphrases, with some duplicated phrases in the gold references filtered out and thus downgrading its performance. Overall, these results of empirical evaluation confirm that SSRank performs better than almost all the benchmark methods across different datasets and domains.

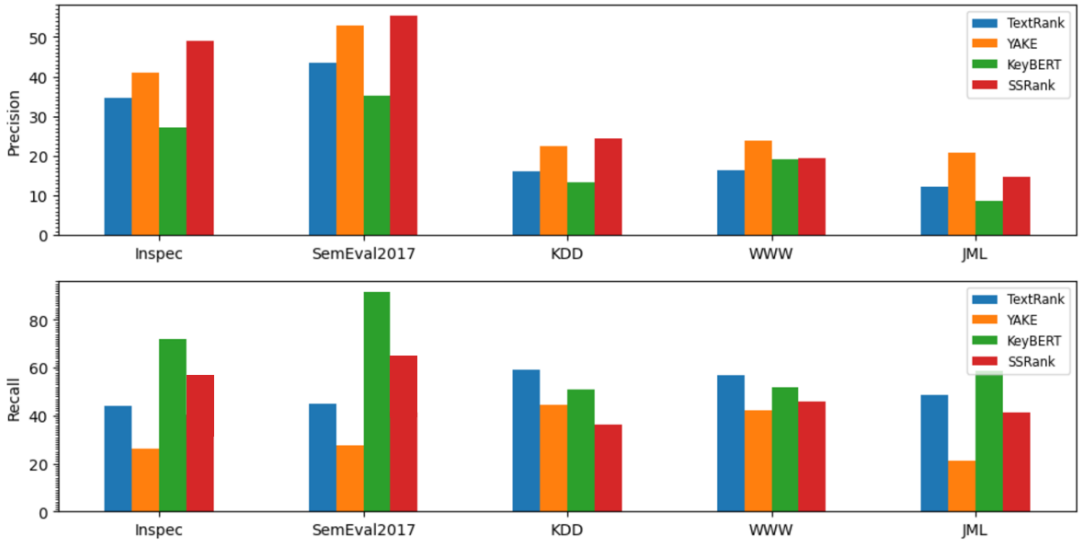

Fig 4.4: Comparison of Precision and Recall results of different benchmarks on different datasets

To conclude, SSRank successfully outshines all other unsupervised AKE baselines by performing the best on 4/5 datasets with the F@EM metric and on 3/5 datasets with the F@PM metric.

4.3.2 Case Study

After the quantitative assessment, the thesis also compares the outputs of SSRank with other methods to confirm its effectiveness in extracting keyphrases that are syntactically integrated and semantically diversified.

Robust output feedback model predictive control using off-line linear matrix inequalities. A fundamental question about model predictive control. Since we employ an off-line approach for the controller design which gives a sequence of explicit control laws, we are able to analyze the robust stabilizability of the combined control laws and estimator, and by adjusting the design parameters, guarantee robust stability of the closed-loop system in the presence of constraints. The algorithm is illustrated with two examples.
SSRank predictions: closed-loop system, robust stabilizability, design parameter, robust output feedback model predictive control, off-line linear matrix inequality
YAKE! predictions: model predictive control, feedback model predictive, output feedback model, linear matrix inequalities, model predictive, off-line linear matrix, Robust output feedback, predictive control, feedback model, output feedback, linear matrix, matrix inequalities, inequalities A fundamental, fundamental question, explicit control laws, combined control laws, control laws, off-line linear, Robust output, guarantee robust stability
KeyBERT predictions: model predictive control, predictive control, predictive control using, robust stabilizability, feedback model predictive
TextRank predictions: model predictive control, explicit control laws, Robust output feedback model predictive control, robust stability, constraints, the combined control laws, estimator, linear, the robust stabilizability, the design parameters, the controller design, A fundamental question, the closed-loop system, the presence, a sequence, two examples, The algorithm

Table 4.6: An example of predicted keyphrase by various AKE methods. Phrases in red are exact matching keyphrases, and phrases in blue are partial matching keyphrases.

Table 1 shows a piece of text selected from the INSPEC dataset, which is taken as an example to qualitatively analyze SSRank's performance. It can be seen that SSRank correctively predicts the first two keyphrases "robust output feedback model predictive control" and "off-line linear matrix inequality" that appear early in text. This is powered by SSRank's outstanding ability in extracting high-quality noun phrases and rating highly of words appearing in the first sentence or in the title. By comparison, YAKE! was able to identify phrases that match in part with these two phrases, like "model predictive control" and "linear matrix inequalities", but failed to churn out the complete and exactly matching form. This may be due to its restriction of n-gram range (2, 3) as the gold keyphrases goes beyond its predefined n-gram length. It also outputs a bundle of overlapping phrases that are semantically close like "feedback model predictive" "output feedback model", and gramatically broken like "inequalities A fundamental" "guarantee robust stability". The same problem occurs in keyphrases produced by KeyBERT, which are also semantically repetitive like "model predictive control"

“predictive control” and grammatically unsound like “predictive control using” “feedback model predictive”.

SSRank also successfully picks out the keyphrase “closed-loop system”, which was detected by TextRank as well but neglected by YAKE! and KeyBERT. This shows the capacity of graph-based methods to discover keyphrases that are statistically infrequent but structurally connected with other significant words. For example in this case, “closed-loop system” is preceded by “robust stability” and followed by “constraint”, and is thus given more importance. However, SSRank did not include the phrase “predictive control”, as it is embodied in the first keyphrase. This again exemplified SSRank’s inclination to promote diversity and eliminate redundancy.

4.4 Application Scenario

To further explore the practical value of SSRank on large-scale literature analysis, the author used SSRank to extract a total set of 25173 keywords on a collection of 3814 scientific abstracts from the Association for Computational Linguistics (ACL)[15] anthology, which is a top global conference related to natural language processing and computational linguistics. It should be noted that the original anthology is not assigned with any keywords. SSRank showed it high time efficiency as the processing finished within 25 seconds. The term co-occurrence network and keyword annual trend were displayed in the following charts, reflecting SSRank’s strong potential in wide applications in the academic field.

[15] https://aclanthology.org/

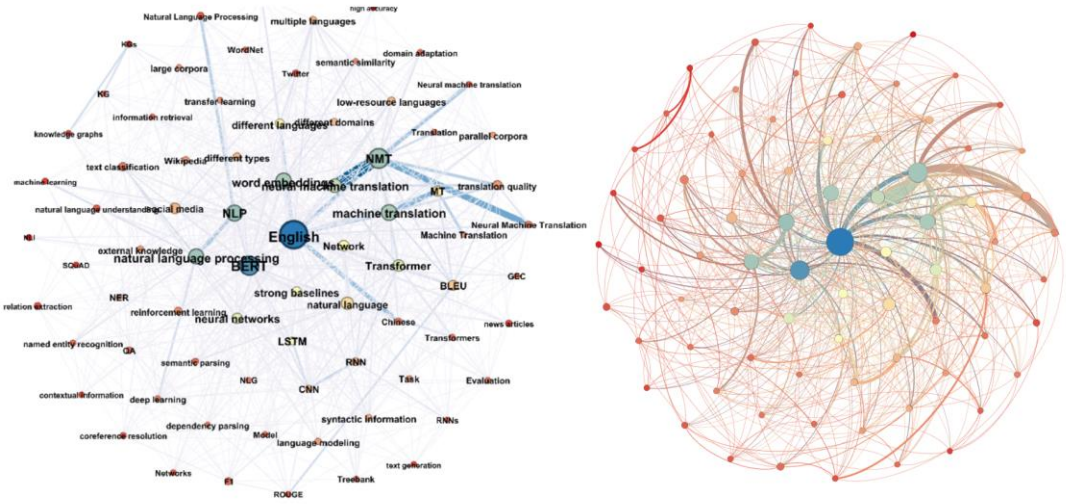

Fig 4.5: The term co-occurrence network of ACL

From the annual trend in recent three years, it is interesting to find the recent explosion of research on large-scale Pretrained Language Models (PLMs) from 2022, and the steady and common interest in Bidirectional Encoder Representations from Transformers (BERT).

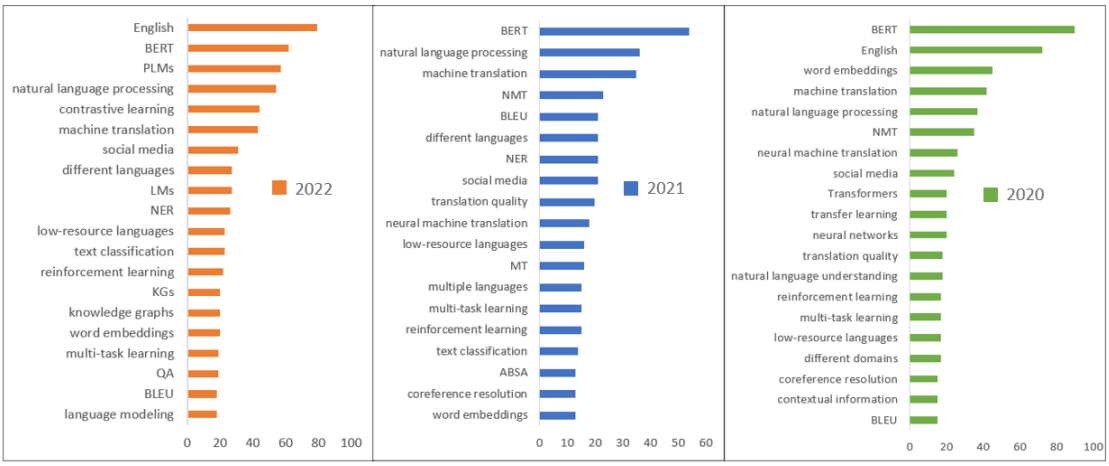

Fig 4.6: The keyword annual trend of ACL in recent three years

4.5 Chapter Summary

In this chapter, a series of extensive experiments conducted to verify the effectiveness of SSRank are introduced.

Results of the ablation studies show the outstanding performance of its NP detector,

obtaining the highest recall of 0.919 on INSPEC and significantly outperforms other publicly available noun phrase extraction tools. The studies also show the significant contribution of the semantic clustering component, increasing the F@EM value of the model by 0.0133 points, an increase of 10.68%; and F@EM with around 0.0303 points, an increase of 8.9%. Parameter tuning results show that SSRank achieves the best performance with the damping factor d=0.9, window size w=3 and using HAC-POE for semantic clustering.

The second experiment evaluated the overall effectiveness of SSRank compared with three baseline methods, showing that SSRank generally outperformed these representative statistical, graph-based and language-model based approaches. And the case study shows its ability in generating grammatically sound and semantically diverse terms.

Finally, the simple but powerful application scenario provided in the last section shows the time efficiency and practical values of SSRank in the scholarly context.

Chapter 5 Conclusions and Limitations

Automatic Keyword Extraction (AKE) is a method based on text mining and natural language processing technology, which can provide a succinct summary of lengthy text by automatically extracting the most important and representative keywords from it. It plays an indispensable role in the indexing and retrieval of large-scale scientific documents, helping researchers quickly understand the topic and content of documents, and providing references for research directions. Despite its increasing value and demand, existing methods of keyword extraction is vulnerable to a series of problems like grammatical incompleteness, reliance on frequency, content generality, and semantic duplication.

To address these issues, thesis proposes a novel unsupervised algorithm SSRank for keyword extraction that caters to the four essential characteristics of keywords, and integrates both structural and semantic information in scholarly text. Given a document from existing dataset or any corpus built using web scrapping, SSRank first extracts noun phrase candidates with noise removed via regex matching of POS and dependency tags. Then, for *structural* information, unique words taken from the candidate phrases form the nodes of the structural graph, and the count of co-occurrence pairs form the edges to compute an importance score for each word, and are pooled to be the aggregated score of each phrase. For *semantic* information, the phrases are clustered into different semantic groups using HAC clustering or K-means to enhance topical coverage and diversity. These groups are also ranked according to within-group distance, which selects groups that have closer distance and thus more inner duplications; and theme alignment with the original text, which selects groups that are proximate to the central meaning of the text. Finally, the top-ranking phrases from the top-ranking groups are chained together to represent the final set of keywords.

The thesis evaluated the effectiveness of SSRank on four representative datasets and one linguistics-related dataset constructed by the author using traditional metrics as well as a newly proposed matching method called *partial match*. The outcomes of a

series of comprehensive quantitative and qualitative experimental analyses including ablation, comparison and case studies have substantiated that the model exhibit superior performance in comparison to the existing state-of-the-art models, and proved the prominent ability of SSRank in preserving both termhood and diversity of the extracted keywords. The final application scenario reflects the effectiveness, practicality and convenience of the SSRank algorithm.

Apart from the above prominent performances, there are several extra advantages of SSRank that sets it apart from existing unsupervised methods.

First, many state-of-the-art unsupervised algorithms require the specification of keyphrase length. For example, when implementing KeyBERT the default n-gram range is (1, 2). But the number can vary significantly as some phrases can extend to as long as 6 words. SSRank differs in that it extracts noun phrases of different lengths with no n-gram range to be predefined. Instead, it constructs a nonlinear formula based on Zipf's Law to model the effect of length on the probability of a candidate being selected as keywords. Thus, SSRank is good at dealing with the length problem.

Second, the flexibility of SSRank is also reflected in its unfixed number of output keywords. In previous works, the number of keyphrases k is usually prespecified as 5 or 10, but for SSRank the number can vary according to the document length and the choice of semantic clusters. This gives high elasticity in SSRank as it can adapt the number of outputs to the given input.

Third, SSRank manifest magnificent ability in the retrieval of noun phrases containing stopwords, like "philosophy of mind"; separated with hyphen, like "E-Z Reader"; ending or embedded with a verb, like "task switching" and "working memory".

However, the study is not without its constraints and shortcomings:

(5) The Limits in Algorithmic Design

First, since SSRank is specially designed for the scholarly context, its ability to be generalized across diverse domains beyond scientific abstracts like news articles and book reviews is still under investigation. Second, SSRank is currently available for English text only, despite its potential to be adapted to any language by switching the

noun phrase detector, stopword list and other necessary components to another language. Moreover, due to the limitations of the extractive paradigm, it is impossible to generate abstractive keywords that never appear in the original document, though they constitute a large portion of the gold keyphrase set. Finally, the performance of SSRank is only tested on short documents, with the larger terrain of long documents still unexplored. A solution for keyword extraction from long text may be extracting key sentences first to form a smaller set of sentences and then applying SSRank to the short text.

(6) The Limits in Experiments

In the experiment, although the results were extensively compared exploiting different metrics and datasets, it may be better to further assess the quality of the produced keyphrases through human screening and rating, which can provide a more comprehensive and reliable judgement of the output. Moreover, although the positive effect of the SSRank semantic clustering process on the overall algorithm performance has been verified in ablation studies, there is still a lack of evaluation criteria and techniques for the semantic clustering process alone. In addition, the interaction between the two components of structural scoring and semantic clustering still remains to be discovered, and the underlying mechanism remains to be explored.

(7) Directions for Future Study

Bottomed on the overall architecture of SSRank, future research can focus on various combinations of different methods of structural importance ranking and semantic clustering to find the optimal solution to keyword extraction. More importantly, how to integrate this structural-semantic paradigm into large-scale deep neural networks to model the termhood, distribution, informativeness and diversity of keyphrases is also a valuable research topic.

Appendix A: List of Part-of-Speech Tags

Appendix A provides additional information about the full list of Part-of-Speech (POS) tags used in spaCy and initially provided by the Penn Treebank Project, including universal tags suitable for all languages and fine-grained tags for English language only.

Open class words	Closed class words	Other
ADJ	ADP	PUNCT
ADV	AUX	SYM
INTJ	CCONJ	X
NOUN	DET	
PROPN	NUM	
VERB	PART	
	PRON	
	SCONJ	

Table A.1: Alphabetical list of universal part-of-speech tags for all languages

Number	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential *there*
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	*to*
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person

		singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

Table A.2**:** Alphabetical list of fine-grained part-of-speech tags for English language

Appendix B: Dependency Labels for English

Label	Description	Explanation
ACL	Clausal modifier of noun	A finite or non-finite clausal modifier (acl) is either an infinitival modifier is an infinitive clause or phrase that modifies the head of a noun phrase, or a participial modifier is a clause or phrase whose head is a verb in a participial form (e.g., gerund, past participle) that modifies the head of a noun phrase, or a complement.
ACOMP	Adjectival complement	An adjectival complement (acomp) is an adjective phrase that modifies the head of a VP|SINV|SQ, that is usually a verb.
ADVCL	Adverbial clause modifier	An adverbial clause modifier (advcl) is a clause that acts like an adverbial modifier.
ADVMOD	Adverbial modifier	An adverbial modifier (advmod) is an adverb or an adverb phrase that modifies the meaning of another word.
AGENT	Agent	An agent (agent) is the complement of a passive verb that is the surface subject of its active form.
AMOD	Adjectival modifier	An adjectival modifier (amod) is an adjective or an adjective phrase that modifies the meaning of another word, usually a noun.
APPOS	Appositional modifier	An appositional modifier (appos) of an NML|NP is a noun phrase immediately preceded by another noun phrase, which gives additional information to its preceding noun phrase.
ATTR	Attribute	An attribute (attr) is a noun phrase that is a non-VP predicate usually following a copula verb.
AUX	Auxiliary	An auxiliary (aux) is an auxiliary or modal verb that gives further information about the main verb (e.g., tense, aspect).
AUXPASS	Auxiliary (passive)	A passive auxiliary (auxpass) is an auxiliary verb, be, become, or get, that modifies a passive verb.
CASE	Case marker	A case marker (case) is either a possessive marker, ...
CC	Coordinating conjunction	A coordinating conjunction (cc) is a dependent of the leftmost conjunct in coordination.
CCOMP	Clausal complement	A clausal complement (ccomp) is a clause with an internal subject that modifies the head of an ADJP|ADVP|NML|NP|WHNP|VP|SINV|SQ.
COMPOUND	Compound modifier	A compound (compound) is either a noun modifying the head of noun phrase, a number modifying the head of quantifier phrase, or a hyphenated word (or a preposition modifying the head of the prepositioanl phrase).

CONJ	Conjunct	A conjunct (conj) is a dependent of the leftmost conjunct in coordination.
CSUBJ	Clausal subject	A clausal subject (csubj) is a clause in the subject position of an active verb.
CSUBJPASS	Clausal subject (passive)	A clausal passive subject (csubjpass) is a clause in the subject position of a passive verb.
DATIVE	Dative	A dative (dative) is a nominal or prepositional object of dative-shifting verb.
DEP	Unclassified dependent	
DET	Determiner	A determiner (det) is a word token whose pos tag is DT|WDT|WP that modifies the head of a noun phrase.
DOBJ	Direct Object	A direct object (dobj) is a noun phrase that is the accusative object of a (di)transitive verb.
EXPL	Expletive	An expletive (expl) is an existential there in the subject position.
INTJ	Interjection	
MARK	Marker	A marker (mark) is either a subordinating conjunction (e.g., although, because, while) that introduces an adverbial clause modifier, or a subordinating conjunction, if, that, or whether, that introduces a clausal complement.
META	Meta modifier	A meta modifier (meta) is code (1), embedded (2), or meta (3) information that is randomly inserted in a phrase or clause.
NEG	Negation modifier	A negation modifier (neg) is an adverb that gives negative meaning to its head.
NOUNMOD	Modifier of nominal	A modifier of nominal (nmod) is any unclassified dependent that modifies the head of a noun phrase.
NPMOD	Noun phrase as adverbial modifier	An adverbial noun phrase modifier (npmod) is a noun phrase that acts like an adverbial modifier.
NSUBJ	Nominal subject	A nominal subject (nsubj) is a non-clausal constituent in the subject position of an active verb.
NSUBJPASS	Nominal subject (passive)	A nominal passive subject (nsubjpass) is a non-clausal constituent in the subject position of a passive verb.
NUMMOD	Number modifier	A numeric modifier (nummod) is any number or quantifier phrase that modifies the head of a noun phrase.
OPRD	Object predicate	An object predicate (oprd) is a non-VP predicate in a small clause that functions like the predicate of an object.
PARATA	Parataxis	A parenthetical modifier (parataxis) is an embedded chunk, often but not necessarily surrounded by parenthetical

XIS		notations (e.g,. brackets, quotes, commas, etc.), which gives side information to its head.
PCOMP	Complement of preposition	A complement of a preposition (pcomp) is any dependent that is not a pobj but modifies the head of a prepositional phrase.
POBJ	Object of preposition	An object of a preposition (pobj) is a noun phrase that modifies the head of a prepositional phrase, which is usually a preposition but can be a verb in a participial form such as VBG.
POSS	Possession modifier	A possession modifier (poss) is either a possessive determiner (PRP$) or a NML|NP|WHNP containing a possessive ending that modifies the head of a ADJP|NML|NP|QP|WHNP.
PRECONJ	Pre-correlative conjunction	A pre-correlative conjunction (preconj) is the first part of a correlative conjunction that becomes a dependent of the first conjunct in coordination.
PREDET	Pre-determiner	A predeterminer (predet) is a word token whose pos tag is PDT that modifies the head of a noun phrase.
PREP	Prepositional modifier	A prepositional modifier (prep) is any prepositional phrase that modifies the meaning of its head.
PRT	Particle	A particle (prt) is a preposition in a phrasal verb that forms a verb-particle construction.
PUNCT	Punctuation	Any punctuation (punct) is assigned the dependency label PUNCT.
QUANTMOD	Modifier of quantifier	
RELCL	Relative clause modifier	A relative clause modifier (relcl) is a either relative clause or a reduced relative clause that modifies the head of an NML|NP|WHNP.
ROOT	Root	A root (root) is the root of a tree that does not depend on any node in the tree but the artificial root node.
XCOMP	Open clausal complement	An open clausal complement (xcomp) is a clause without an internal subject that modifies the head of an ADJP|ADVP|VP|SINV|SQ.

Appendix C: Detailed Dataset Statistics

Table 4.3 shows the distribution of keyword lengths in different datasets with percentage sign removed. We can hardly observe any keywords that are longer than 8-grams, but SemEval2017 is the special exception that contains keywords as long as 25 words.

Dataset	Length							
	1	2	3	4	5	6	7	8
INSPEC	16.44	53.68	23.05	5.27	1.13	0.32	0.09	0.02
SemEval2017	25.23	33.74	17.19	8.69	4.98	2.99	2.13	1.44
KDD	25.48	56.32	13.97	3.56	0.55	0.1	0	0.03
WWW	34.36	47.71	12.15	5.17	0.47	0.08	0.03	0.03
JML	32.33	56.45	9.35	1.68	0.14	0	0.04	0

Table C.1: Distribution of keyword n-gram length per dataset (%). Blue shading highlights the largest value in each row.

The distribution of keyword POS patterns in Table 4.4.

POS Pattern	INSPEC	SemEval2017	KDD	WWW	JML	Total
Adj*Noun+	59.33	38.76	60.17	58.81	70.35	57.48
Verb Noun*	5.34	6.81	12.31	11.32	7.49	8.65
PropN*Noun*	5.06	5.11	5.37	2.96	2.72	4.24
PropN	1.28	4.33	1.23	3.28	1.65	2.35
Noun Verb	1.1	0.74	2.39	2.31	1.25	1.55

Table C.2: Distribution of keyword POS pattern per dataset (%). Blue shading highlights the largest value in each column.

References

Alami Merrouni, Z., Frikh, B., & Ouhbi, B. (2020). Automatic keyphrase extraction: A survey and trends. *Journal of Intelligent Information Systems*, *54*(2), 391–424. https://doi.org/10.1007/s10844-019-00558-9

Al-Zaidy, R. A., & Giles, C. L. (2017, August). Automatic knowledge base construction from scholarly documents. In *Proceedings of the 2017 ACM symposium on document engineering* (pp. 149-152).

Amin, A., Rana, T. A., Mian, N. A., Iqbal, M. W., Khalid, A., Alyas, T., & Tubishat, M. (2020). TOP-Rank: A Novel Unsupervised Approach for Topic Prediction Using Keyphrase Extraction for Urdu Documents. *IEEE Access*, *8*, 212675–212686. https://doi.org/10.1109/ACCESS.2020.3039548

Arampatzis, A., & Kamps, J. (2008, July). A study of query length. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 811-812).

Augenstein, I., Das, M., Riedel, S., Vikraman, L., & McCallum, A. (2017). Semeval 2017 task 10: Scienceie-extracting keyphrases and relations from scientific publications. *arXiv preprint arXiv:1704.02853*.

Bennani-Smires, K., Musat, C., Hossmann, A., Baeriswyl, M., & Jaggi, M. (2018). Simple unsupervised keyphrase extraction using sentence embeddings. *arXiv preprint arXiv:1801.04470*.

Bodenreider, O. (2004). The unified medical language system (UMLS): integrating biomedical terminology. *Nucleic acids research*, *32*(suppl_1), D267-D270.

Borisov, O., Aliannejadi, M., & Crestani, F. (2021). Keyword Extraction for Improved Document Retrieval in Conversational Search. *arXiv:2109.05979 [cs]*. http://arxiv.org/abs/2109.05979

Boudin, F., Gallina, Y., & Aizawa, A. (2020). Keyphrase Generation for Scientific Document Retrieval. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 1118–1126. https://doi.org/10.18653/v1/2020.acl-main.105

Bougouin, A., Boudin, F., & Daille, B. (2013). TopicRank: Graph-Based Topic Ranking for Keyphrase Extraction. *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, 543–551. https://www.aclweb.org/anthology/I13-1062

Campos, R., Mangaravite, V., Pasquali, A., Jorge, A. M., Nunes, C., & Jatowt, A. (2018, March). YAKE! collection-independent automatic keyword extractor. In *European Conference on Information Retrieval* (pp. 806-810). Springer, Cham.

Campos, R., Mangaravite, V., Pasquali, A., Jorge, A., Nunes, C., & Jatowt, A. (2020). YAKE! Keyword extraction from single documents using multiple local features. *Information Sciences*, *509*, 257–289. https://doi.org/10.1016/j.ins.2019.09.013

Çano, E., & Bojar, O. (2019). Keyphrase Generation: A Multi-Aspect Survey. *2019 25th Conference of Open Innovations Association (FRUCT)*, 85–94. https://doi.org/10.23919/FRUCT48121.2019.8981519

Çano, E., & Bojar, O. (2020). Two Huge Title and Keyword Generation Corpora of Research Articles. *Proceedings of the Twelfth Language Resources and Evaluation Conference*, 6663–6671. https://aclanthology.org/2020.lrec-1.823

Carstairs-McCarthy, A. (2017). *Introduction to English Morphology: words and their structure*. Edinburgh university press.

Castellví, M. T. C. (2003). Theories of terminology: Their description, prescription and explanation. *Terminology. International Journal of Theoretical and Applied Issues in Specialized Communication*, *9*(2), 163-199.

Chan, H. P., Chen, W., Wang, L., & King, I. (2019). Neural Keyphrase Generation via Reinforcement Learning with Adaptive Rewards. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2163–2174. https://doi.org/10.18653/v1/P19-1208

Chen, J., Zhang, X., Wu, Y., Yan, Z., & Li, Z. (2018). Keyphrase Generation with Correlation Constraints. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 4057–4066. https://doi.org/10.18653/v1/D18-1439

Chen, W., Chan, H. P., Li, P., & King, I. (2020). Exclusive Hierarchical Decoding for Deep Keyphrase Generation. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 1095–1105. https://doi.org/10.18653/v1/2020.acl-main.103

Ding, H., & Luo, X. (2021). AttentionRank: Unsupervised Keyphrase Extraction using Self and Cross Attentions. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 1919–1928. https://doi.org/10.18653/v1/2021.emnlp-main.146

Dutta, S., Assem, H., & Burgin, E. (2021). Sequence-to-Sequence Learning on Keywords for Efficient FAQ Retrieval. *arXiv:2108.10019 [cs]*. http://arxiv.org/abs/2108.10019

El-Beltagy, S. R., & Rafea, A. (2009). KP-Miner: A keyphrase extraction system for English and Arabic documents. *Information systems*, *34*(1), 132-144.

Florescu, C., & Caragea, C. (2017, July). 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)* (pp. 1105-1115).

Frantzi, K., Ananiadou, S., & Mima, H. (2000a). Automatic recognition of multi-word terms:. The C-value/NC-value method. *International Journal on Digital Libraries*, *3*(2), 115–130. https://doi.org/10.1007/s007999900023

Garg, A., Kagi, S. S., & Singh, M. (2020). SEAL: Scientific Keyphrase Extraction and Classification. *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*, 527–528. https://doi.org/10.1145/3383583.3398625

Garg, K., Chowdhury, J. R., & Caragea, C. (2022). *Keyphrase Generation Beyond the Boundaries of Title and Abstract* (arXiv:2112.06776). arXiv. http://arxiv.org/abs/2112.06776

Gollapalli, S. D., & Caragea, C. (2014, June). Extracting keyphrases from research papers using citation networks. In *Proceedings of the AAAI conference on*

artificial intelligence (Vol. 28, No. 1).

Gollapalli, S. D., Li, X., & Yang, P. (2017). Incorporating Expert Knowledge into Keyphrase Extraction. *Proceedings of the AAAI Conference on Artificial Intelligence*, *31*(1). https://doi.org/10.1609/aaai.v31i1.10986

Goutte, C., & Gaussier, E. (2005). A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. In *Advances in Information Retrieval: 27th European Conference on IR Research, ECIR 2005, Santiago de Compostela, Spain, March 21-23, 2005. Proceedings 27* (pp. 345-359). Springer Berlin Heidelberg.

Grishman, R. (2015). Information extraction. *IEEE Intelligent Systems*, *30*(5), 8-15.

Grootendorst, M. (2020). KeyBERT: Minimal keyword extraction with BERT. Zenodo. 10.5281/zenodo.4461265.

Hey, T., & Trefethen, A. (2003). The data deluge: An e-science perspective. *Grid computing: Making the global infrastructure a reality*, 809-824.

Huang, X., Xu, T., Jiao, L., Zu, Y., & Zhang, Y. (2021). Adaptive Beam Search Decoding for Discrete Keyphrase Generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(14), 13082–13089. https://doi.org/10.1609/aaai.v35i14.17546

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

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *J. Document, 28*(1) ,11-21.

Justeson, J. S., & Katz, S. M. (1995). Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural language engineering*, *1*(1), 9-27.

Kazemi, A., Pérez-Rosas, V., & Mihalcea, R. (2020). Biased TextRank: Unsupervised graph-based content extraction. *arXiv preprint arXiv:2011.01026*.

Khabsa, M., & Giles, C. L. (2014). The number of scholarly documents on the public web. *PloS one*, *9*(5), e93949.

Kim, J., Jeong, M., Choi, S., & Hwang, S. (2021). Structure-Augmented Keyphrase Generation. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2657–2667. https://doi.org/10.18653/v1/2021.emnlp-main.209

Knittel, J., Koch, S., & Ertl, T. (2021). ELSKE: Efficient Large-Scale Keyphrase Extraction. *Proceedings of the 21st ACM Symposium on Document Engineering*, 1–4. https://doi.org/10.1145/3469096.3474930

Koloski, B., Pollak, S., Škrlj, B., & Martinc, M. (2021). Extending Neural Keyword Extraction with TF-IDF tagset matching. *arXiv:2102.00472 [cs]*. http://arxiv.org/abs/2102.00472

Kulkarni, M., Mahata, D., Arora, R., & Bhowmik, R. (2022). Learning Rich Representation of Keyphrases from Text. *Findings of the Association for Computational Linguistics: NAACL 2022*, 891–906. https://doi.org/10.18653/v1/2022.findings-naacl.67

Lai, T. M., Bui, T., Kim, D. S., & Tran, Q. H. (2020). A Joint Learning Approach based on Self-Distillation for Keyphrase Extraction from Scientific Documents. *arXiv:2010.11980 [cs]*. http://arxiv.org/abs/2010.11980

Langville, A. N., & Meyer, C. D. (2004). Deeper inside pagerank. *Internet Mathematics*, *1*(3), 335-380.

Levenshtein, V. (1965). Binary codes capable of correcting spurious insertions and deletions of ones. *Russian Problemy Peredachi Informatsii*, *1*, 12-25.

Liang, X., Wu, S., Li, M., & Li, Z. (2021). Unsupervised Keyphrase Extraction by Jointly Modeling Local and Global Context. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 155–164. https://doi.org/10.18653/v1/2021.emnlp-main.14

Mahata, D., Kuriakose, J., Shah, R. R., & Zimmermann, R. (2018). Key2Vec: Automatic Ranked Keyphrase Extraction from Scientific Articles using Phrase Embeddings. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 634–639. https://doi.org/10.18653/v1/N18-2100

Mary, V., Marquet, G., & Le Beux, P. (2004). MeSH and specialized terminologies: coverage in the field of molecular biology. In *MEDINFO 2004* (pp. 530-534). IOS Press.

Meng, R., Zhao, S., Han, S., He, D., Brusilovsky, P., & Chi, Y. (2017). Deep Keyphrase Generation. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 582–592. https://doi.org/10.18653/v1/P17-1054

Meng, Rui, Debanjan Mahata, and Florian Boudin. "From Fundamentals to Recent Advances: A Tutorial on Keyphrasification." *Advances in Information Retrieval: 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10–14, 2022, Proceedings, Part II*. Cham: Springer International Publishing, 2022.

Miah, M. B. A., Awang, S., Azad, Md. S., & Rahman, M. M. (2022). Keyphrases Concentrated Area Identification from Academic Articles as Feature of Keyphrase Extraction: A New Unsupervised Approach. *International Journal of Advanced Computer Science and Applications*, *13*(1). https://doi.org/10.14569/IJACSA.2022.0130192

Mihalcea, R., & Tarau, P. (2004). TextRank: Bringing Order into Text. *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 404–411. https://www.aclweb.org/anthology/W04-3252

Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM*, *38*(11), 39-41.

Nakagawa, H., & Mori, T. (2002). A simple but powerful automatic term extraction method. In *COLING-02: COMPUTERM 2002: Second International Workshop on Computational Terminology*.

Nokkaew, K., & Kongkachandra, R. (2018). Keyphrase Extraction as Topic Identification Using Term Frequency and Synonymous Term Grouping. *2018 International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)*, 1–6. https://doi.org/10.1109/iSAI-NLP.2018.8693001

Pantel, P. (2005). Inducing ontological co-occurrence vectors. In Proceedings of the 43rd Conference of the Association for Computational Linguistics, ACL'05 (pp. 125–132). Morristown, NJ, USA: Association for Computational Linguistics.

Papagiannopoulou, E., & Tsoumakas, G. (2020). A review of keyphrase extraction. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *10*(2), e1339.

Pikies, M., Riyono, A., & Ali, J. (2020). Novel Keyword Extraction and Language Detection Approaches. *arXiv:2009.11832 [cs]*. http://arxiv.org/abs/2009.11832

Rabby, G., Azad, S., Mahmud, M., Zamli, K. Z., & Rahman, M. M. (2020). TeKET: A Tree-Based Unsupervised Keyphrase Extraction Technique. *Cognitive Computation*, *12*(4), 811–833. https://doi.org/10.1007/s12559-019-09706-3

Rajput, S., Gahoi, A., Reddy, M., & Sharma, D. M. (2020, December). N-Grams TextRank A Novel Domain Keyword Extraction Technique. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON): TermTraction 2020 Shared Task* (pp. 9-12).

Ramos, J. (2003, December). Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning* (Vol. 242, No. 1, pp. 29-48).

Rose, S., Engel, D., Cramer, N., & Cowley, W. (2010). Automatic Keyword Extraction from Individual Documents. In M. W. Berry & J. Kogan (Eds.), *Text Mining* (pp. 1–20). John Wiley & Sons, Ltd. https://doi.org/10.1002/9780470689646.ch1

Rubenstein, H., & Goodenough, J. (1965). Contextual correlates of synonymy. Communications of the ACM, 8 (10), 627–633.

Rubin, R. E. (2017). *Foundations of library and information science*. American Library Association.

Sag, I. A., Baldwin, T., Bond, F., Copestake, A., & Flickinger, D. (2002). Multiword expressions: A pain in the neck for NLP. In *International conference on intelligent text processing and computational linguistics*. Springer, Berlin, Heidelberg.

Sahrawat, D., Mahata, D., Zhang, H., Kulkarni, M., Sharma, A., Gosangi, R., & Zimmermann, R. (2020). Keyphrase extraction as sequence labeling using contextualized embeddings. In *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II 42* (pp. 328-335). Springer International Publishing.

Salloum, S. A., Al-Emran, M., Monem, A. A., & Shaalan, K. (2018). Using text mining techniques for extracting information from research articles. *Intelligent natural language processing: Trends and Applications*, 373-397.

Saxena, A., Mangal, M., & Jain, G. (2020). KeyGames: A Game Theoretic Approach to Automatic Keyphrase Extraction. *Proceedings of the 28th International Conference on Computational Linguistics*, 2037–2048. https://doi.org/10.18653/v1/2020.coling-main.184

Schütze, H., Manning, C. D., & Raghavan, P. (2008). *Introduction to information retrieval* (Vol. 39, pp. 234-265). Cambridge: Cambridge University Press.

Shen, S., & Lee, H. (2016). *Neural Attention Models for Sequence Classification: Analysis and Application to Key Term Extraction and Dialogue Act Detection*

(arXiv:1604.00077). arXiv. http://arxiv.org/abs/1604.00077

Shen, X., Wang, Y., Meng, R., & Shang, J. (2021). *Unsupervised Deep Keyphrase Generation* (arXiv:2104.08729). arXiv. http://arxiv.org/abs/2104.08729

Subramanian, S., Wang, T., Yuan, X., Zhang, S., Trischler, A., & Bengio, Y. (2018). Neural Models for Key Phrase Extraction and Question Generation. *Proceedings of the Workshop on Machine Reading for Question Answering*, 78–88. https://doi.org/10.18653/v1/W18-2609

Sun, C., Hu, L., Li, S., Li, T., Li, H., & Chi, L. (2020). A Review of Unsupervised Keyphrase Extraction Methods Using Within-Collection Resources. *Symmetry*, *12*(11), 1864. https://doi.org/10.3390/sym12111864

Sun, S., Liu, Z., Xiong, C., Liu, Z., & Bao, J. (2021). *Capturing Global Informativeness in Open Domain Keyphrase Extraction* (arXiv:2004.13639). arXiv. http://arxiv.org/abs/2004.13639

Sun, Z., Tang, J., Du, P., Deng, Z.-H., & Nie, J.-Y. (2019). DivGraphPointer: A Graph Pointer Network for Extracting Diverse Keyphrases. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 755–764. https://doi.org/10.1145/3331184.3331219

Swaminathan, A., Zhang, H., Mahata, D., Gosangi, R., Shah, R. R., & Stent, A. (2020). A Preliminary Exploration of GANs for Keyphrase Generation. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 8021–8030. https://doi.org/10.18653/v1/2020.emnlp-main.645

Tixier, A., Malliaros, F., & Vazirgiannis, M. (2016). A Graph Degeneracy-based Approach to Keyword Extraction. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 1860–1870. https://doi.org/10.18653/v1/D16-1191

Torres-Cruz, F., Flores, E., Arcaya, W. E., Chagua, I. L., & Ingaluque, M. I. (2022). Unsupervised Learning Algorithms for Keyword Extraction in an Undergraduate Thesis. *International Journal for Research in Applied Science and Engineering Technology*, *10*(3), 785–790. https://doi.org/10.22214/ijraset.2022.40758

Ushio, A., Liberatore, F., & Camacho-Collados, J. (2021). Back to the Basics: A Quantitative Analysis of Statistical and Graph-Based Term Weighting Schemes for Keyword Extraction. *arXiv:2104.08028 [cs]*. http://arxiv.org/abs/2104.08028

Viswanathan, V., Neubig, G., & Liu, P. (2021). CitationIE: Leveraging the Citation Graph for Scientific Information Extraction. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 719–731. https://doi.org/10.18653/v1/2021.acl-long.59

Wang, Y., Fan, Z., & Rose, C. (2020). Incorporating Multimodal Information in Open-Domain Web Keyphrase Extraction. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1790–1800. https://doi.org/10.18653/v1/2020.emnlp-main.140

Wang, Y., Li, J., Chan, H. P., King, I., Lyu, M. R., & Shi, S. (2019). Topic-Aware Neural Keyphrase Generation for Social Media Language. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2516–2526.

https://doi.org/10.18653/v1/P19-1240

Willis, A., Davis, G., Ruan, S., Manoharan, L., Landay, J., & Brunskill, E. (2019). Key Phrase Extraction for Generating Educational Question-Answer Pairs. *Proceedings of the Sixth (2019) ACM Conference on Learning @ Scale*, 1–10. https://doi.org/10.1145/3330430.3333636

Wu, D., Ahmad, W. U., Dev, S., & Chang, K.-W. (2022). *Representation Learning for Resource-Constrained Keyphrase Generation* (arXiv:2203.08118). arXiv. http://arxiv.org/abs/2203.08118

Wu, H., Liu, W., Li, L., Nie, D., Chen, T., Zhang, F., & Wang, D. (2021). UniKeyphrase: A Unified Extraction and Generation Framework for Keyphrase Prediction. *arXiv:2106.04847 [cs]*. http://arxiv.org/abs/2106.04847

Wu, H., Ma, B., Liu, W., Chen, T., & Nie, D. (2022). Fast and Constrained Absent Keyphrase Generation by Prompt-Based Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *36*(10), 11495–11503. https://doi.org/10.1609/aaai.v36i10.21402

Ye, H., & Wang, L. (2018). Semi-Supervised Learning for Neural Keyphrase Generation. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 4142–4153. https://doi.org/10.18653/v1/D18-1447

Ye, H., & Wang, L. (2019). *Semi-Supervised Learning for Neural Keyphrase Generation* (arXiv:1808.06773). arXiv. http://arxiv.org/abs/1808.06773

Ye, J., Cai, R., Gui, T., & Zhang, Q. (2021). Heterogeneous Graph Neural Networks for Keyphrase Generation. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2705–2715. https://doi.org/10.18653/v1/2021.emnlp-main.213

Ye, J., Gui, T., Luo, Y., Xu, Y., & Zhang, Q. (2021). One2Set: Generating Diverse Keyphrases as a Set. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 4598–4608. https://doi.org/10.18653/v1/2021.acl-long.354

Yuan, X., Wang, T., Meng, R., Thaker, K., Brusilovsky, P., He, D., & Trischler, A. (2020). *One Size Does Not Fit All: Generating and Evaluating Variable Number of Keyphrases* (arXiv:1810.05241). arXiv. http://arxiv.org/abs/1810.05241

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

Zehtab-Salmasi, A., Feizi-Derakhshi, M.-R., & Balafar, M.-A. (2021). FRAKE: Fusional Real-time Automatic Keyword Extraction. *arXiv:2104.04830 [cs]*. http://arxiv.org/abs/2104.04830

Zesch, T., & Gurevych, I. (2009). Approximate Matching for Evaluating Keyphrase Extraction. *Proceedings of the International Conference RANLP-2009*, 484–489. https://aclanthology.org/R09-1086

Zhang, C., Zhao, L., Zhao, M., & Zhang, Y. (2022). Enhancing keyphrase extraction from academic articles with their reference information. *Scientometrics*, *127*(2), 703-731.

Zhang, M., Qin, B., Zheng, M., Hirst, G., & Liu, T. (2015). Encoding Distributional

Semantics into Triple-Based Knowledge Ranking for Document Enrichment. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 524–533. https://doi.org/10.3115/v1/P15-1051

Zhang, Y., Zhang, C., & Li, J. (2020). Joint Modeling of Characters, Words, and Conversation Contexts for Microblog Keyphrase Extraction. *Journal of the Association for Information Science and Technology*, *71*(5), 553–567. https://doi.org/10.1002/asi.24279

Zhao, J., & Zhang, Y. (2019). Incorporating Linguistic Constraints into Keyphrase Generation. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 5224–5233. https://doi.org/10.18653/v1/P19-1515

Zheng, Y., Lu, X., & Ren, W. (2020). Tracking the Evolution of Chinese Learners' Multilingual Motivation Through a Longitudinal Q Methodology. *The Modern Language Journal*, *104*(4), 781-803.

Zhu, X., Lyu, C., & Ji, D. (2020). Keyphrase Generation With CopyNet and Semantic Web. *IEEE Access*, *8*, 44202–44210. https://doi.org/10.1109/ACCESS.2020.2977508