

**AUTOMATIC EXTRACTION OF COMPUTER SCIENCE
CONCEPT PHRASES USING A HYBRID MACHINE
LEARNING PARADIGM**

by

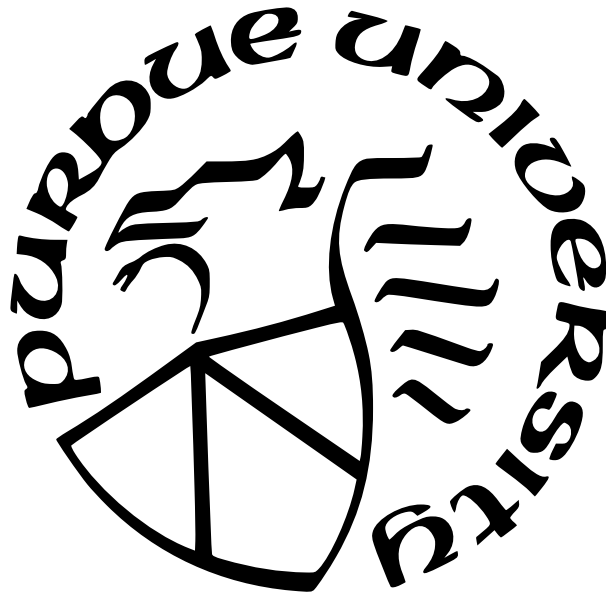
S M Abrar Jahin

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science



Department of Computer and Information Science

Indianapolis, Indiana

December 2022

**THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL**

Dr. Mohammad Al Hasan, Chair

Computer & Information Science

Dr. Shiaofen Fang

Computer & Information Science

Dr. Snehasis Mukhopadhyay

Computer & Information Science

Approved by:

Dr. Shiaofen Fang

To my family

ACKNOWLEDGMENTS

Professor Dr. Mohammad Al Hasan, chair of my committee and my research supervisor, and Professor Dr. Md. Nour Hossain, another supervisor for this research, have provided me with tremendous patience and input for which I am indebted beyond measure. I also could not have embarked on this road without the information and assistance of my defense committee. In addition, this work would not have been possible without the generosity of the National Science Foundation, which funded my research under Grant IIS-1909916.

I am also appreciative to my lab mates, particularly my friend Md Ahsanul Kabir Rana, for his considerable assistance in discussing various issues, late-night feedback sessions, and moral support. Thanks should also be extended to my team members and classmates, in particular Thushara Manjari Naduvilakandy for her outstanding contribution to the project.

It would be improper of me not to mention my family, especially my parents and my spouse. Their confidence in me has maintained my spirits and motivation strong throughout this endeavor.

PREFACE

In the modern world, the growth of human knowledge is exponential. Also, computer science is one of the most promising and quickly growing fields of study in general. If someone is new to computer science and wants to study it, they may feel like they are in an ocean of new information. There are numerous learning resources and tutorials for computer science available online. Some of them are good, while others fall short of expectations. But the main goal of each tutorial is to set up a learning path that fits a task that the student is interested in. To be clearer, since we are mostly focused on the academic side, these works do not meet our standards.

It would be amazing if there were a way that could help people learn the things they desire. A knowledge graph[1] can be of tremendous use in this situation. As we are primarily interested in computer science learning for beginners, computer science knowledge graphs[1] can be used to create an excellent computer science learning tool.

As we were interested in developing a learning tool and a knowledge graph[1] appears to be a good approach, we are growing interested in the issue and would like to know if it can be utilized to meet our requirements. When attempting to construct a computer science knowledge graph[1], we looked for a database or dictionary from which to construct the graph. A few computer science dictionaries are available from various organizations. However, they are either obsolete or more field-specific. Since we are interested in all areas of computer science and the subject is expanding exponentially, it is difficult to create a database for computer science because they become obsolete so quickly. So, we planned to make an automatic system that would add the phrases to the database on its own. If the database can automatically add new terms, there is less chance that it will become out of date.

In the third chapter of this research article, we explore the reason for our study, including why we chose to work on this subject and how we arrived at our current research objective. After that, we addressed the terms we need to know in order to proceed with this investigation. In the next chapter, we'll talk about past works that have something to do with our own. After we finished that chapter, we looked at a simulation example in which

we simulated the example's inputs and outputs. Following that, we described the data set used in our experiment. Then, we discussed our proposed algorithm and the results of our investigation. At the conclusion of the essay, we covered the planned future work for this study project.

TABLE OF CONTENTS

LIST OF TABLES	9
LIST OF FIGURES	10
ABSTRACT	11
1 INTRODUCTION	13
2 BACKGROUND	17
2.1 Knowledge Graph	17
2.2 Term Extraction	19
2.3 Different types of machine learning models	21
2.4 LSTM	23
2.5 BiLSTM	23
2.6 Transformers	24
2.7 BERT	25
3 PREVIOUS WORKS	26
3.1 Yake	26
3.2 FRAKE	26
3.3 Key-BERT	27
3.4 Spacy	27
3.5 Gensim	28
3.6 Improved TextRank	28
3.7 Rake	29
4 RESEARCH TASK	30
4.1 Problem Formulation	30
4.2 Proposed Solution	31
5 DATASET	32

5.1	SemEval-2 Dataset	32
5.2	Extraction Results on SemEval-2	33
5.3	Wikipedia Dataset	33
6	PROPOSED ALGORITHM	35
6.1	Process of the Algorithm	35
6.1.1	Utilize an effective Existing Generic Text Extraction Method	35
6.1.2	Apply ad-hoc post-processing to filter, add or change phrases	35
	Add hyphen-based phrases	36
	Remove Single words	37
	Rule-based phrase removal	37
6.1.3	Classify candidate phrases using a machine-learning model	38
	BERT Data Embedding	39
	Bi-LSTM	41
	Attention Layer	41
	Output Layer	43
6.2	Result and Output	44
6.2.1	Performance Comparison	45
7	CONCLUSIONS AND RECOMMENDATIONS	47
	REFERENCES	49

LIST OF TABLES

5.1	Extraction method comparison	33
6.1	Apply RAKE state result	35
6.2	Acronym collection state result	36
6.3	Addition of Sentences with Hyphens	37
6.4	Result after rule based phrase removal applied	37
6.5	Extraction method comparison	46

LIST OF FIGURES

6.1	Work of AI-based classifier in brief	38
6.2	tSNE 3D plot of CS vs NonCS phrases	40
6.3	tSNE 2D plot of CS vs NonCS phrases	40
6.4	tSNE 2D plot of CS vs NonCS phrases	41
6.5	Classifier model Train Graph	44

ABSTRACT

With the proliferation of computer science in recent years in modern society, the number of computer science-related employment is expanding quickly. Software engineer has been chosen as the best job for 2023 based on pay, stress level, opportunity for professional growth, and balance between work and personal life. This was decided by a rankings of different news, journals, and publications. Computer science occupations are anticipated to be in high demand not just in 2023, but also for the foreseeable future. It's not surprising that the number of computer science students at universities is growing and will continue to grow. The enormous increase in student enrolment in many subdisciplines of computers has presented some distinct issues. If computer science is to be incorporated into the K-12 curriculum, it is vital that K-12 educators are competent. But one of the biggest problems with this plan is that there aren't enough trained computer science professors. Numerous new fields and applications, for instance, are being introduced to computer science. In addition, it is difficult for schools to recruit skilled computer science instructors for a variety of reasons including low salary issue. Utilizing the K-12 teachers who are already in the schools, have a love for teaching, and consider teaching as a vocation is therefore the most effective strategy to improve or fix this issue. So, if we want teachers to quickly grasp computer science topics, we need to give them an easy way to learn about computer science. To simplify and expedite the study of computer science, we must acquaint school-teachers with the terminology associated with computer science concepts so they can know which things they need to learn according to their profile.

If we want to make it easier for schoolteachers to comprehend computer science concepts, it would be ideal if we could provide them with a tree of words and phrases from which they could determine where the phrases originated and which phrases are connected to them so that they can be effectively learned. To find a good concept word or phrase, we must first identify concepts and then establish their connections or linkages. As computer science is a fast developing field, its nomenclature is also expanding at a frenetic rate. Therefore, adding all concepts and terms to the knowledge graph would be a challenging endeavor. Creating a system that automatically adds all computer science domain terms to the knowledge graph

would be a straightforward solution to the issue. We have identified knowledge graph use cases for the school-teacher training program, which motivates the development of a knowledge graph. We have analyzed the knowledge graph's use case and the knowledge graph's ideal characteristics. We have designed a web-based system for adding, editing, and removing words from a knowledge graph. In addition, a term or phrase can be represented with its children list, parent list, and synonym list for enhanced comprehension. We've developed an automated system for extracting words and phrases that can extract computer science idea phrases from any supplied text, therefore enriching the knowledge graph. Therefore, we have designed the knowledge graph for use in teacher education so that school-teachers can educate K-12 students computer science topics effectively.

1. INTRODUCTION

With the proliferation of computer and information technology in today’s life, jobs relating to the computing discipline are growing exponentially. Based on multiple rankings conducted in 2023 by USNews [2], Bloomberg [3], and CNBC [4], software engineer or software developer has been selected as the best job considering salary, stress level, career growth opportunity, and work-life balance [5]. US Bureau of Labor Statistics (BLS) anticipates a 22% increase in employment vacancies, or over 400,000 additional positions in software development, until 2030 [6]. The strong career opportunity in the CS discipline has not gone unnoticed by aspiring students; in fact, US universities are experiencing unprecedented growth in the number of students who want to major in CS [7], [8]. Such growth motivated Universities to invest additional resources in the CS department, often by hiring more professors, and often by offering a variety of degrees closely related to CS. For instance, many universities now offer bachelor’s or advanced degrees in Software Engineering, Data Science, Machine Learning, CyberSecurity, or Artificial Intelligence—each, a sub-discipline of CS [9].

The enormous growth of student enrollment in various computing sub-disciplines has brought some unique challenges. In past, students who are genuinely interested in CS, and are well-prepared in terms of their background knowledge chose CS as their intended major. At present, with the large number of enrollments, CS readiness of students varies substantially; among the students, there are many, who are not well-prepared to be successful in their CS educational endeavour [10]. A remedy to this challenge, as identified by the experts of the National Academy of Science, is to increase the exposure of computing concepts, computational principles, information security, and data analytics to students throughout the K-12 pipeline [11][12].

If we want to incorporate computer science into the K-12 curriculum, it is imperative that we have qualified K-12 educators. However, the scarcity of qualified computer science instructors in the K-12 level is one of the greatest obstacles to this strategy [11], [13]. A reason for this is the given lucrative job opportunities in industries, recent CS graduates are not very keen on taking high school teaching jobs. Besides, existing teachers who are on this job for a long period of time are not well trained to teach computer science [14]. This

is partly due to the fact that computer science is a rapidly evolving field with continuous updates of its contents [15]. Numerous new domains and applications are being added to computer science; more languages and algorithms are added than ever before. Also, there is a significant amount of research in computer science. All these make it more difficult even for a good teacher to keep up with the trends in computer science. Besides, difficulty in establishing a consistent computer science curriculum for students in grades K-12 is another crucial issue [16].

The most viable solution to the above issues is to utilize the current K-12 teachers who are already in the schools, have a passion for teaching, and view teaching as a career. However, they should be given an opportunity to self-train computer science concepts at their own pace, and convenience. The platform from where they learn CS concepts should be free, updated, and organized. The platform should provide a repository which describes various computer science concepts in an easy-to-understand manner.

To make learning computer science simple and quick, we must familiarize school instructors with computer science concept terminology, and phrases [17] so they know which are which. Now comes the topic of what computer science concept phrases are and why we require them. We are looking at words and phrases that are used in computer science but not in other fields. If a word or phrase is used in both computer science and other fields, we mark it as non-computer science and leave it out of the learning materials. So, "computer science concept terms" or "computer science phrases" are terms or phrases that explain computer science ideas. If school-teachers are familiar with computer science terminology, they will find it easier to grasp these subjects. and familiarity with such issues would facilitate their learning.

Because the field of computer science changes so quickly, it is hard to find a complete list of computer science terms and phrases. Therefore, we must isolate the concept terms. We primarily extracted words and concepts from the provided text using three processes. As a result of identifying the ideas in the first stage, we know that the words we're interested in are inside our area of focus. The second step is to establish the link between the concept phrases. Last, in order to make a useful database, we need to collect both old and new concept phrases. To gather computer science phrases, we've created a comprehensive, deep

learning-based extraction technique that can extract computer scientific terms from a given text. The supplied text might be a sentence, paragraph, essay, or article. In addition, we have created the early preview of knowledge graph, a web-based collaborative program for storing terms with their associations.

If we want to simplify the process for school teachers to understand computer science concepts, it would be ideal if we could provide them with a tree of words and phrases from which they can identify where the phrases originated and which phrases are connected to the phrases so they can be effectively learned. As computer science is a rapidly expanding discipline, its associated terminology is also expanding at a furious speed. So, adding all ideas and phrases to the knowledge graph would be a difficult task. If we could make a system that automatically adds all computer science domain words to the knowledge graph, that would be a simple way to solve the problem. If we are able to do this, then it will be simpler for public school teachers to learn from them in a variety of ways. If school teachers select a phrase or word from the knowledge graph, they are able to select any term and view which terms or phrases are the phrase's parents. From the parent list, it's easy to see where the word came from, which helps the reader understand what the phrase means and how it works. Then, users can look at the list of the phrase's children to figure out what branches it has and what is directly connected to it. Then, they can examine the phrase's synonym list to see which words are interchangeable. If school teachers know the synonyms, they can quickly spot the words in their different forms when they appear in a document. Again, if teachers could see the image of the linkages through a graphical representation, the material would be more interesting and easier to comprehend. Moreover, if the knowledge graph were a "living" knowledge graph, i.e., if new computer science-related concepts were added automatically, that would be excellent for school teachers' learning systems.

We have identified knowledge graph use cases for the school-teacher training scheme, which drives us to develop a knowledge graph. We have replicated nearly all of the aspects of our vision's knowledge graph. We have developed a web-based knowledge graph system where terms may be added, edited, and removed. If the word has child and parent terms and phrases, the system allows for the creation, modification, and deletion of the child and parent terms and phrases. In addition, there is an appealing live search mechanism that

allows users to easily find the desired term. In the live search, any user may search for a term or phrase using a variety of filtering and sorting options that make it easy to locate any desired term. If they pick a term or phrase linked to computer science and that term or phrase has a synonym, the synonyms will be readily visible when they mouse over the term or phrase. In addition, we focused on automating the collection of terms. We have developed a tool based on deep learning that extracts computer science-related terms from a given text. The extracted phrases may be put into the knowledge graph if we can automatically generate the edges of the phrases, which is one of our research project's next key tasks.

My contributions to this master's thesis include the following:

1. Preparatory work for constructing a knowledge graph
2. Collection and preparation of data
3. A sophisticated method for the extraction of computer science idea phrases
4. Developing a web-based software solution for the creation, visualization, and modification of a preliminary knowledge graph

2. BACKGROUND

In this chapter I will discuss some background material that is related to the contribution of this thesis.

2.1 Knowledge Graph

An ontology is a formal and structural way of representing the concepts and relations in a domain. Ontologies are widely used for natural language understanding, question answering, machine translation, and intelligent personal assistance services, and decision support systems in life sciences, to name a few. Formally, ontologies are represented by formal language based tools, including DARPA Agent Markup Language (DAML) or Web Ontology Language (OWL). For an ontology, these languages provide options for storing concepts, the type of concepts, relationships among concepts, and the type of relationships. One can also add axioms that determine the validity and define constraints in ontologies. However, ontologies can also be defined informally as a directed graph in which the concepts are vertices and the relation between the concepts are the edges. Vertex and edge labels encode the type of concepts and the type of relationships. In fact, such informal versions of ontology are also named as knowledge graph (KG), a term whose definition is not yet properly established [18].

In recent years, there has been tremendous interest in building knowledge graphs (KGs), in which nodes are entities and a pair of nodes are connected by an edge with one or more relational labels. The semi-structured representation of a knowledge graph provides semantically structured information that is important for building human-like intelligent machines. Such machines can drive various knowledge-based applications, including question answering systems, intelligent personal assistance services, and decision support systems in life sciences, to name a few.

There are many advantages to using a graph-based abstraction of knowledge instead of a relational model or NoSQL. Graphs are a compact and understandable method for representing a wide variety of topics. Edges show the different and often complicated ways that things in a domains are connected to each other. With graphs, maintainers don't have to choose a schema right away, giving the data more time to change. Standard relational

operators (like joins, unions, projections, etc.) and navigational operators can be used in graph query languages to find items linked by paths of any length. Ontologies and rules can be used to define the terms in a graph and figure out what they mean. Scalable frameworks for graph analytics can be used to calculate things like centrality, clustering, summarization, etc., to learn more about the area in question. There are currently emerging ways of applying machine learning to graphs.

A large number of industrial-scale knowledge graphs [19] have already been created, noteworthy among them are YAGO [20], DBpedia [21], NELL [22], Freebase [23], and Knowledge Vault [24]. Among these, Yago and DBpedia use semi-structured data, such as Wikipedia infoboxes, as their source, along with substantial human intervention, which enables them to maintain highly accurate facts. On the other hand, NELL and Knowledge Vault use an automated unstructured approach, where facts are extracted from the natural language text of Web pages. The KGs from the latter group have high coverage and they have the ability to grow organically. From a computational point of view as well, the latter group has an advantage, as they can scale almost linearly with the size of their source data, which makes them an attractive choice considering the proliferation of information and the impressive growth of online content.

Although scalable, the fidelity of the facts in automatically constructed KGs remains a big concern. While the information in YAGO and DBpedia can be highly accurate, exceeding 95% accuracy, the accuracy of NELL, an organically generated knowledge graph using never ending learning was reported to be around 74% [22]. Besides, the accuracy of any automatically generated knowledge graph decreases over time as the incorrect facts are compounded through the inference process using an existing incorrect fact. The relatively low accuracy of automatically constructed KGs, in most cases, is due to name entity ambiguity, or relation-type ambiguity that can be mitigated by using a rich context for an entity.

In this thesis, we build a knowledge graph prototype for the computer science discipline. This knowledge graph is different from the existing real-life knowledge graph in different ways. First, the entities in our knowledge graph have only a few types, which include sub-discipline, model name, algorithm name, abstract CS objects, or CS concept terms, whereas in real-life KG, the entities can be very diverse, such as person, organization, object, product,

location, etc. Second, the relation in our knowledge graph are mostly semantic relations, such as is-a (hypernym-hyponym), hyponym-sibling, cause-effect, part-of (meronym-holonym), synonym, sequential (succession in space and time), argumental (process-agent, process-state), etc.

2.2 Term Extraction

To build a knowledge graph for the CS discipline, we first need to identify a collection of entities representing CS concept. When such entities are related to a certain language or a field of study, we may refer them as terms, keywords, or phrases; they can be composed of words or multi-word phrases, which are used to describe an object or communicate a notion in that language or in that field of study. Typically, one use “term” to refer a single-word or multi-word entity, on the other hand, a “phrase” must always consist of more than one word. For the CS knowledge graph that we are building, we have considered two to five word phrases as candidate terms. Also note, we evaluate terms and phrases based on their close association exclusively with computer science. For instance, if the term or phrase is utilized in multiple contexts and makes meaning in other fields of study, we consider it to be a non-computer science expression. For instance, "hard problem" is regarded as a non-CS word because it is applicable in a variety of contexts. On the contrary, if the term is "NP-hard problem," then it is a term from computer science [25].

In the Information Retrieval or Digital Library research discipline, the process of extracting terms from text corpora is known as term extraction, which generates a list of core vocabulary of a specific subject. A terminologist extracts terms manually by compiling a list of prospective term candidates and then consulting with a domain expert to generate a final list of verified terms. Manual process for indexing, and describing a domain’s core vocabulary is a labor-intensive task, specifically, for a domain like CS, where the vocabulary changes quickly and is always growing. To overcome this challenge, we can adopt automated term extraction, which uses a computer program to extract terms from a text corpus relevant to the given domain. Manual and automated term extraction has their relative strengths and weaknesses. Manual term extraction produces high quality vocabulary, thanks to the

knowledge of domain experts who validate those terms, but the process is slow and labor intensive [26]. On the other hand, automated process is fast, and it can extract a large number of terms if appropriate corpus is used. However, it is less accurate as extracted terms may not represent a core entity in the given domain.

In information retrieval literature, automatic Term Extraction is also called keyword extraction, terminology mining, term recognition, glossary extraction, term identification, and term acquisition. The process is based on automated text corpus analysis. It uses a machine-learning-based agent which bases its decisions objectively on corpus evidence. Because the terminological status of a phrase is frequently a question of degree and susceptible to individual variation, automatic term extraction can overcome the subjectivity of an expert that may possibly impact the term extraction-process. Besides, automatic term extraction saves the expert from having to look through the whole text by hand. The idea of using automatic term extraction is that we can use it as a preliminary filter to gather a large number of potential term candidates, which this process does well. Despite these benefits of automatic term extraction, it is important to highlight that words are fundamentally semantically defined, since they relate to a domain-specific notion, and that fully automated modelling of semantics is currently beyond the capabilities of computers. The final confirmation of a term's status must still be done by hand by experts in the field [27].

For performing automatic term (or keyword) extraction, there are several algorithms, which differs based on the methodology that they adopt. For example, term extraction can be formulated using statistical [28][29][30], graphical [31], learning-based [32][33][34], or hybrid methodologies [35][36][37]. It is noticed that the Graphical Keyword Extraction Methodologies (GKET) are more scalable and computationally less expensive than the other Automatic Keyword Extraction (AKE) techniques. There are different ways of text extraction, but all major types of text extractions are directly or indirectly related with Graph of Word [27][31].

2.3 Different types of machine learning models

Machine learning models accept a collection of data and produce meaningful knowledge from that. There are many different kinds of machine learning models. Among which the followings are prominent:

- **Supervised Learning:** Given training data consisting of a collection of data instances, each of which is a pair: a data vector and a label (a number or a category value), a supervised learning algorithm uses the training data to train a model which approximates a function to map a data vector to the corresponding label. Since each data instance is associated with a label, the model is called “supervised”, as the labels guide the learning algorithm to learn the unknown function which maps the data instance to the label.
- **Unsupervised learning:** Unsupervised learning, commonly referred to as unsupervised machine learning, utilizes machine learning algorithms to evaluate and cluster unlabeled information. Without the need for human interaction, these algorithms uncover hidden patterns or data groupings. It is great for exploratory data analysis, cross-selling techniques, consumer segmentation, and picture identification because of its capacity to identify similarities and contrasts in data [38][39].
- **Semi-supervised learning:** Semi-supervised learning is a blend of strategies for supervised and unsupervised learning. It combines a little quantity of manually labeled data (a supervised learning element) with a vast amount of unlabeled data. Through data clustering, this approach makes it feasible to train a machine learning (ML) algorithm on data annotation (e.g., the labeling or categorization of data) without manually labeling all of the training data, thereby enhancing efficiency without compromising quality or accuracy [40]. For instance, if you have a large data set containing images of dogs and cats, a semi-supervised approach would enable you to manually label a small portion of the data (identifying a few images as "dogs" and a few others as "cats"), and the machine learning algorithm would then be equipped to define the remaining data. This combines the advantages of supervised and unsupervised learn-

ing by encouraging the algorithm to make autonomously sound judgments with less initial human intervention.

- **Reinforcement learning:** In this learning paradigm, an algorithm learns a policy on how to act based on observations of the real world. Every action has an effect on the surrounding environment, and the environment offers input to the learning process.
- **Dimensionality Reduction:** Often dimensionality reduction can be categorized as unsupervised learning as for this task data instances are not associated with a label. It is used when a dataset has an excessive number of characteristics or dimensions, which negatively impact machine algorithms (overfitting). It decreases the quantity of data inputs to a tolerable level while retaining the integrity of the dataset to the greatest extent feasible. Dimensionality reduction is often employed in the data preparation phase, and there are a number of available approaches, such as Principal component analysis (PCA), Singular value decomposition (SVD), Autoencoders.
- **Representation learning** Representation learning is a type of algorithm, which learns a meaningful representation vector (also called embedding vector) for complex data objects, like words, phrases, sentences, and images. When used such representation, the performance of downstream tasks improves substantially. For example, when words or sentences are embedded by using dense real-valued vectors obtained from representation learning, semantically similar words are embedded in nearby vector, which contributes to improved performance for tasks like document classification, and ranking. In general, embedding improve generalization and performance for virtually every machine learning task, especially when training data is scarce. In recent years, deep neural networks are used for learning representation of complex data objects. This may be one of the most important discoveries that lead to the success of deep learning models in many tasks in natural language processing and computer vision.

2.4 LSTM

LSTM is an acronym for Long Short-Term Memory Networks, which are utilized in the field of Deep Learning. It is a subset of recurrent neural networks (RNNs), which can learn long-term dependencies, especially in tasks that involve predicting what will happen next. Because it has feedback links, LSTM can process the whole sequence of data, not just single data points like pictures. This is useful for speech recognition, machine translation, etc. LSTM is a subtype of RNN that performs exceptionally well on a wide range of issues. LSTM works well in areas where RNN are used. For example, it does a great job of recognizing sequences.

2.5 BiLSTM

BiLSTMs, or bidirectional LSTMs [41], are basically an enhancement over LSTMs. In bidirectional LSTMs, each training sequence is shown both forward and backward to separate recurrent nets. Both sequences are linked to the same layer of output. Bidirectional LSTMs have a lot of information about each point in a sequence [42], including all the points that come before and after it. However, the challenge is how to rely on information that has not yet occurred. The human brain employs its senses to gather information from words, sounds, and entire phrases that may make no sense at first but will make sense in a later context. Conventional recurrent neural networks can only obtain information from the prior context. In contrast, information is gained in bidirectional LSTMs by processing data in both directions within two hidden layers that are pushed toward the same output layer. This facilitates access to long-range context in both directions for bidirectional LSTMs. If a sentence is given as an example, the LSTM model will work from beginning to end or end to beginning, depending on the content and target domain. But with BiLSTM, the model will start to work in both directions, and the output of one layer will be used as the input for the next layer. Using bidirectional LSTMs, we are able to feed the learning algorithm with the original data from beginning to end and back again. Despite its domain- and task-specificity, this method often learns quicker than one-directional LSTM.

2.6 Transformers

Transformer networks turn a given series of components, such as a sequence of words in a phrase, into another sequence. Seq2Seq models are great at translation, which is the process of changing the order of words in one language into the order of words in another language. Long-Short-Term-Memory (LSTM)-based models are a common option for this task. With sequence-dependent data, LSTM modules give meaning to a sequence while remembering (or forgetting) the significant bits (or unimportant). Any Seq2Seq model consists of an Encoder and a Decoder. The Encoder converts the input sequence into a space with a higher dimension (n-dimensional vector). The Decoder receives this abstract vector and converts it into an output sequence. The output sequence might be in a different language, symbols, a duplicate of the input, etc. The Seq2Seq model works great for small sentence or text, but does not work well for longer text. In those cases, we require a different solution, similar to how the human brain works.

People have to deal with many different topics and ideas, but not all of them are important. Because of this, the human brain remembers the most important tasks and bases decisions on them. Similarly, the Transformer model does not recall or forget anything dependent on scenarios. Instead, Transformers recall the significant bits of the input. The main characteristics of transformers are non-sequential (sentences are digested as a whole rather than individually), self-attention (this attribute is utilized in NLP operations to compute similarity scores between words in a sentence), positional embeddings (This is yet another innovation made in Transformer to replace repetition. The concept is to employ fixed or learnt weights that encode information about a token's position in a sentence).

RNNs and LSTMs do great most of the time when the input text is small, but they have trouble when there are a lot of dependencies. The primary reason why transformers do not have long-term reliance difficulties is the first point. The original transformers do not rely on previous concealed states to determine word dependencies. They instead process sentences holistically. Therefore, there is no possibility of losing (or "forgetting") historical information. Both multi-head attention and positional embeddings give information about

how words are related to each other. So, for longer text, transformers generally yields better results.

2.7 BERT

BERT [43][44], which stands for "Bidirectional Encoder Representations from Transformers," is based on Transformers, a deep learning model in which every output element is connected to every input element and the weightings between them are made dynamically based on how they are related(In NLP, this is referred to as attention). BERT is an open-source embedding framework for machine learning and natural language processing (NLP). BERT is aimed to assisting computers in understanding the meaning of ambiguous words in the text by establishing context from the surrounding text. The BERT framework was already trained on Wikipedia text, and the embedding model can be fine-tuned with question-and-answer datasets if demanded.

In the past, language models could only read text from left to right or right to left, but not both at the same time. BERT is unique in that it is meant to be read in both directions simultaneously. This capacity, made possible by the development of transformers, is referred to as "bi-directionality." BERT is already trained on two NLP tasks that are different but related: masked language modeling and predicting the next sentence. The goal of Masked Language Model (MLM) training is to hide a word in a phrase and have the algorithm guess (mask) the hidden word based on the context. The goal of Next Sentence Prediction training is to teach the computer to predict whether two given phrases go together in a logical, sequential way or just by chance.

As we already know, BERT is pre-trained on a massive database, making it more generalized for many viewpoints, and we can quickly download and use the BERT model for our purposes. But occasionally, embedding requires more expertise, to be more specific domain knowledge. BERT also offers this type of adaptability. If necessary, we may adjust the BERT model using our own data set to make it more accurate for our target data set [45][46].

3. PREVIOUS WORKS

Keyphrase extraction is concerned with automatically extracting a set of representative phrases from a document that concisely summarizes its content [47]. There are not any prior works dealing with the extraction of computer science phrases from texts, but there are a number of prior methods for extracting keywords from texts. Among these, we have identified the followings which works well. They are: YAKE [48], FRAKE [49], Key-BERT [50], Spacy [51][52][53], Gensim [54][55], Improved TextRank [56][57], and RAKE [58].

Below we provide a brief discussion of each of the above methods.

3.1 Yake

In order to find probable candidate phrases, the YAKE [48] technique first pre-processes the content into a machine-readable format. A list of distinct phrases is used as the input for the second phase, which then represents the terms using a set of statistical attributes. In the third stage, these characteristics are heuristically merged to provide a single score that is likely to capture the term’s significance. The candidate keywords are then generated in the fourth stage (using an n-gram creation process) and given scores depending on their significance. Finally, we use a deduplication distance similarity metric to compare terms that are probably related to each other. Following that, the final keyword list is arranged according to relevance ratings.

3.2 FRAKE

The keyword extraction technique known as FRAKE [49] combines textual and graph-based approaches. Pre-processing, feature extraction, scores computation, key-phrase creation, and ranking are the 5 processes that make up this procedure. Stop words are removed from the input data during the preprocessing stage, which separates the data into words and sentences. Graph feature extraction and textural feature extraction are the next steps in the feature extraction process. For the purpose of extracting graph features, an unweighted, undirected graph is formed, with the words serving as the nodes and the 3-gram (trigram)

of words serving as the vertices. Depending on how central each node is, it receives one of eight scores: degree centrality (DE), closeness centrality (CL), betweenness centrality (BE), eigenvector centrality (EV), structural holes (SH), page rank (PR), clustering coefficient (CC), and eccentricity (EC). These scores are distilled down to a single score for each node using PCA. The extraction of textural aspects is then carried out with the aid of features such capital letters, word location, word frequency in document sentences, term frequency of the word, and part of speech (POS) tag of the word. In order to arrive at a single score per word, the estimated scores of words based on graph characteristics and textural aspects are combined. The process of key-phrase creation uses HUPM, and the phrase’s score is determined by adding the computed scores for each keyword in the phrase. The top K terms are chosen as the document’s key phrases after the words and phrases are sorted in order of decreasing score.

3.3 Key-BERT

Key-BERT [50] searches a document for the sub-phrases most similar to the primary phrase using BERT-embeddings and apparent cosine similarity. First, a document-level representation is obtained by extracting document embeddings using BERT. Word embeddings are then taken out for N-gram words and phrases. Finally, we utilize cosine similarity to identify the phrases or words most closely resembling the document. The words that best describe the full document might then be found by comparing terms that are similar to one another.

3.4 Spacy

The spaCy module for Python is a free, open-source tool for advanced Natural Language Processing (NLP). spaCy is built primarily for production usage and facilitates the development of programs that "understand" and analyze enormous amounts of text. It may be used to construct information extraction and natural language comprehension systems, as well as to prepare text for deep learning. When working with a large amount of text, spacy may be a useful tool, and it is utilized for a variety of research tasks [51][52][53]. There

is a module in spaCy for extracting the identified entities from the document in which our interest lies [59]. If an entity recognizer has been applied, this module produces a tuple of named entity Span objects.

3.5 Gensim

The Gensim keywords summary [54] is mostly a clustering-based strategy to summarizing news articles. This article reveals that sentence-based models outperform graph and word-based models [55]. At the initial stage, a news article is selected from the dataset and subjected to a number of preprocessing steps. After preprocessing is complete, the model will execute feature extraction to assign a score to each sentence of the text. The model generated a vector representation of the text using Gensim Word2Vec. The model then distributed the vectorized text into k clusters using the K-Means clustering procedure, where k is the number of clusters. This model determined the optimal value of k using the Elbow approach. The model will then construct a summary by selecting key sentences from these clusters depending on the score assigned to each sentence by our processing technique. One-third of the specified text will comprise the generated summary.

3.6 Improved TextRank

This technique, which is an enhanced variant of the well-known TextRank algorithm [60], is sometimes producing better results [56][57]. The preprocessing of the data, word segmentation, and stop word elimination are the initial steps in the upgraded TextRank algorithm's keyword extraction process. The weights for the TF-IDF, word location and part of speech should then be determined after sorting the preprocessed data into a text set. Give each weight the correct settings. To create the score of multi-feature fusion calculation words, the weight value and parameter value computed are added. The first N words are chosen as the resource's keywords after the words are sorted according to the determined scores.

3.7 Rake

RAKE [58], an automatic keyword extraction technique for extracting keywords from individual documents, begins keyword extraction by parsing a document's text into a collection of candidate keywords by dividing the given text into an array of words using the defined word delimiters. This array is then divided into sequences of contiguous words based on the position of phrase delimiters and stop words. Words inside a sequence are allocated the exact text location and assessed as potential keywords. Additionally, a graph of co-occurrence is constructed from the input text. Additionally, RAKE generates new candidate keywords by combining them with their internal stop words. Then, a score is produced for each potential term by adding the scores of its constituent words. RAKE assessed word frequency, word degree, and the ratio of the degree to frequency for generating word scores based on the degree and frequency of word vertices in the graph. The score for the new keyword is the sum of the scores of its constituent keywords. After candidate keywords are assessed, the highest T-scoring candidates are chosen as document keywords.

For domain-specific phrase extraction, some other exciting algorithms like Spacy [61] and Tf-IDF[62] are excellent at text extraction for particular purposes. Spacy, for instance, performs admirably when used to extract noun keywords. However, if we use Spacy to extract phrases, it might yield better results. In the age of text extraction, the TF-IDF algorithm, a statistical method that determines how relevant a word is to a document within a collection of documents, is also fantastic. To accomplish this, the frequency of a word within a document and its inverse document frequency across a collection of documents are multiplied. However, using TF-IDF for our particular case, we need to get better results when extracting noun phrases for a specific domain.

4. RESEARCH TASK

4.1 Problem Formulation

In this work, we are interested to extract phrases relating to concepts in the Computer Science (CS) discipline from text gleaned from CS technical documents, including, but not limited to, research papers, theses, and Wikipedia articles. Given a text document, which consists of a collection of sentences, we feed each sentence independently to our phrase extraction pipeline, which emits all the CS concept phrases that may exist in that sentence. We are particularly interested in multi-word phrases denoting some concepts in the CS discipline. For instance, given the following sentence: “*Artificial intelligence, cognitive modelling, and neural networks are information processing paradigms inspired by how biological neural systems process data.*”, we want to extract the following phrases: *artificial intelligence*, *cognitive modelling*, *neural networks*, and *information processing paradigm*, as each of these phrases denotes an specific CS concept. Note that, one may argue that *biological neural system* is also a valid phrase; however in this work we extract phrases which are exclusively related to CS discipline denoting some CS cocnept, and exclude phrases which may pertain to other discipline; so we omit *biological neural system* as one of the phrases. Another prospective phrase in this sentence can be *process data*, which is too generic and does not convey a CS concept, so we do not want to extract this phrase as well.

Input Our algorithm or model will accept any text, such as a sentence, paragraph, essay, article, blog post, research paper, or any other text.

Output A list of computer science phrases extracted from the text by the model or algorithm will be available as output. The algorithm can’t come up with any phrases about computer science that aren’t already in the text.

Example

For instance, suppose the text is as follows:

A digital certificate is an electronic document issued by a Certificate Authority (CA). It contains the public key for a digital signature and specifies the identity associated with the key, such as the name of an organization.

The result may then be a list of the following phrases: digital certificate, electronic document, Certificate Authority, public key, digital signature

4.2 Proposed Solution

We follow a hybrid approach to solve this research task. In this approach, we first use an unsupervised extraction techniques to obtain a collection of candidate phrases. Then a supervised approach is used to validate whether a candidate phrase is a CS phrase or not. Our unsupervised extraction method uses existing phrase extraction methodologies; but these methods are not customized to extract CS phrases, so they obtain many candidate phrases which may or may not be a CS phrase; so a supervised approach is needed which confirms whether a candidate phrase is a CS phrase or not.

For unsupervised candidate phrase extraction, we have evaluated several existing phrase extraction techniques. They are Yake, Rake, Frake, KeyBert, Improved Textrank, Spacy, and Gensim. They were discussed in [chapter 3](#). For the purpose of evaluation, we have used existing benchmark datasets and produce the performance of each of the above methods on those datasets. The best performing model is then used for generating CS candidate phrases from Wikipedia articles associated with CS related topics. To obtain such articles, a collection of seed CS keywords are provided by human experts.

For building the supervised model which takes a candidate phrase and classify them into CS phrase and no CS phrase, we need to first train the model with supervised data. Unfortunately, no such labeled data exist. So, we utilize 3 experts who manually labeled a collection of candidate CS phrases, which were obtained from Wikipedia articles by the unsupervised phrase extraction method.

In the next chapter, we discussed the datasets in details.

5. DATASET

As noted in earlier chapter, we have two types of models, an unsupervised term extraction model, and a supervised classifier model. The term extraction model is chosen from a collection of existing phrase extraction methodologies. For making this choice, we first evaluate their performance on a benchmark dataset, called SemEval-2 [63]. Below we provide more description of this dataset.

5.1 SemEval-2 Dataset

SemEval (Semantic Evaluation) is a continuing series of assessments of computational semantic analysis systems; it originated from the Senseval series of word sense evaluations. The objective of the assessments is to investigate the nature of linguistic meaning. Humans possess an innate sense of meaning, but it has proven difficult to convert this sense to computational analyses.

This sequence of assessments provides a method for defining in further detail precisely what is required to calculate meaning. Consequently, assessments provide an emergent method for identifying challenges and solutions for computations with meaning. These activities have evolved in order to elucidate a broader range of linguistic features. Initially, they attempted to detect word senses computationally in a manner that appeared straightforward. They have evolved to examine the interrelationships among sentence parts (e.g., semantic role labeling), the interrelationships between sentences (e.g., coreference), and the substance of what we are saying (semantic relations and sentiment analysis). The objectives of SemEval and Senseval are to assess semantic analysis systems. "Semantic Examination" refers to a formal analysis of meaning, whereas "computational" refers to methodologies that in principle facilitate effective implementation. For our experiment, we have used 2010 published SemEval-2 data-set's task 05 collection which is mentioned below:

task05-TRAIN (STEM)

This data collection consists of 144 research publications and their corresponding keywords. In this data collection, the cited texts are of the essay variety, with many sentences per text, and the cited keywords contain multiple phrases that are relevant to our area of interest. This data set was utilized to compare and select the optimal extraction algorithm for our purposes. In this thesis when we mention SemEval-2 dataset, we refer to task-5 collection.

5.2 Extraction Results on SemEval-2

In Table 5.1, we provide a comparison among different existing term extraction methods. Note that, for comparison, we only use recall metric, as the ground truth is not complete, i.e., there may be more valid terms in the text which are not listed in the ground truth. So, precision metric will not be meaningful. As we can see, Rake has the best performance with a recall value of 0.3751, as it was able to extract 577 terms out of 1538 terms in the ground truth of SemEval-2 dataset. So, for candidate phrase extraction we have used Rake as our chosen phrase extraction method.

Table 5.1. Extraction method comparison

Algorithm name	Found Phrase Count	Recall
Yake	183	0.1189
Frake	24	0.0156
KeyBERT	24	0.0156
SpaCy	10	0.0065
Gensim	92	0.0598
RAKE	577	0.3751

5.3 Wikipedia Dataset

While the SemEval-2 dataset is a good benchmark dataset, which has been substantially used by information retrieval community, a problem with this dataset is that it does not contain CS phrases. So, we also built another dataset from Wikipedia which have CS

phrases. This dataset contains 2,587 Wikipedia articles, which are about various CS concepts covering a diverse sub-fields of CS, such as, algorithms, data science, software engineering, networking, database, and security. The title of these pages were searched through keywords manually collected from various textbooks, research papers, and academic curricula with a rigorous process. From these articles, we have taken 7,975 sentences containing the article title in the sentence. These sentences were then fed into Rake. Output candidate phrases were classified through human annotation to obtain 3199 CS and 12027 non-CS phrases. For human annotation, we have used a team of four CS experts having different research expertises.

6. PROPOSED ALGORITHM

6.1 Process of the Algorithm

Our suggested method consists primarily of three steps:

1. Utilize an effective existing generic text extraction method to extract candidate terms.
2. Apply ad-hoc post-processing to filter, add or change phrases
3. Classify candidate phrases using a machine-learning model

Below we provide a brief overview of each of these steps:

6.1.1 Utilize an effective Existing Generic Text Extraction Method

In the first step, an unsupervised machine-learning algorithm must extract an initial candidate list. As the unsupervised text extraction algorithm, RAKE has been utilized, as mentioned in earlier chapter. If there is a new text extraction method that works better than RAKE, it can be added to this step of our proposed process. After applying RAKE, we get a list of potential keywords and phrases we use in our next steps.

If we apply RAKE to the sentences in **our produced dataset** with CS and non-CS phrases labeled, we obtained an output provided in table 6.1.

Table 6.1. Apply RAKE state result	
CS Phrase Count	5,232
Non-CS Phrase Count	12,266
CS Phrase Count (unique)	2,429
Non-CS Phrase Count (unique)	10,113
Precision	0.2990

6.1.2 Apply ad-hoc post-processing to filter, add or change phrases

During this step, we perform additional post-processing to improve the quality of the results which Rake produces. Rake has some problems when it comes to extracting phrases related to computer science. The following post-processings are applied:

- Add Acronym
- Add hyphen-based phrases
- Remove Single words
- Rule-based phrase removal

Add Acronym When the first letters of a lengthier name or phrase are combined, a new word or name is created, known as an acronym. NATO (North Atlantic Treaty Organization) is an example of an acronym made from the first letters. In our case, we are interested mainly in the long forms of acronyms as they are good phrases. We are also collecting the short forms of acronyms by removing them as single sentences. But RAKE does not extract all of them. So, we need to extract them by ourselves. RAKE does not extract all acronyms from the given text or phrase. Thus we must develop our ad hoc way to collect acronyms for our phrase extractor. We are adding a new step where we collect acronyms to do that. Applying our ad-hoc RAKE post-processing to gather abbreviated acronym forms in the provided text yielded results that look like table 6.2.

Table 6.2. Acronym collection state result

CS Phrase Count	5,446
Non-CS Phrase Count	12,707
CS Phrase Count (unique)	2,566
Non-CS Phrase Count (unique)	10,467
Precision	0.3000

Add hyphen-based phrases

A phrase is considered hyphenated or hyphen-based if there is at least one hyphen between the words. Hyphenated terms are frequently compound terms, which means that a hyphen links two or more words. In our case, hyphens occasionally provide significant phrases, particularly noun phrases. However, RAKE cannot gather all sentences with hyphens between them. So that the performance may be enhanced, we must design an ad-hoc-based approach to gather those terms. Results that resemble table Y were obtained using our ad

hoc hyphenated phrase collection post-processing to extract additional CS phrases from the given text. Results that resemble table 6.3 were obtained using our ad hoc hyphenated phrase collection post-processing to extract additional CS phrases from the given text.

Table 6.3. Addition of Sentences with Hyphens

CS Phrase Count	6,168
Non-CS Phrase Count	14,235
CS Phrase Count (unique)	2,624
Non-CS Phrase Count (unique)	10,482
Precision	0.3023

Remove Single words

Our goal is to only collect phrases so that we don't have to worry about single words or keywords. Therefore, all single-word keywords or phrases have been eliminated.

Rule-based phrase removal

We are interested in noun phrases, although RAKE supplies many other phrases. Some of these sentences, for example, are good noun phrases, while others are boring verb phrases. So, we used the NLTK Parts of Speech tagger to figure out that these phrases didn't belong on our list of target phrases and got rid of them. The improved results are represented in table 6.4 after using the rule-based classifier. If rule-based phrase removal is used at this stage, 3,299 categorized phrases are removed, of which 380 are CS phrases (false negative), and 2,919 are non-cs words (true negative). As the recall is strong in this circumstance, we are implementing this step and eliminating 380 possible useful CS phrases.

Table 6.4. Result after rule based phrase removal applied

CS Phrase Count	5,788
Non-CS Phrase Count	11,316
CS Phrase Count (unique)	2,293
Non-CS Phrase Count (unique)	7,703
Precision	0.3384

6.1.3 Classify candidate phrases using a machine-learning model

The phrases are classified into computer and non-computer science phrases using a supervised deep-learning classifier. Fifteen thousand two hundred twenty-six data points that have been manually labeled and are listed in the dataset section are used to train this model. The model is described in brief on figure 6.1 at page 38, and it has the following layers:

1. Extract different words from the phrase
2. BERT Data Embedding
3. Bi-LSTM
4. Attention Layer
5. Repetition Layer
6. Aggregation and Output Layer

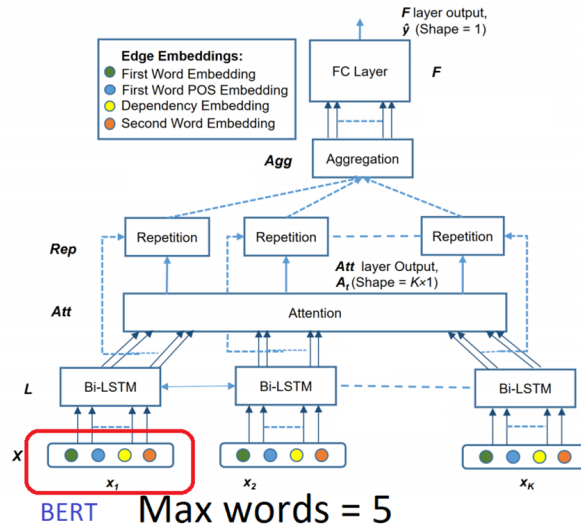


Figure 6.1. Work of AI-based classifier in brief

To understand why we decided to utilize these specific layers for our model, let's take a brief glance at the model's layers.

BERT Data Embedding

For our machine learning model to work with the text and phrases, it needs to understand what the text means. To do this, we have employed the Sentence-BERT [64] model that has been pre-trained. This model does exceptionally well with single words, but not so well with sentences and phrases. As we need to work with phrases, we have used this to identify word embedding, and then we have built phrase embedding from the embedding of individual words, which can be utilized in the next step.

All the words of a phrase are not equally important, so we have to identify which words are important and which are not important. If we like to do that, we need to process all the words of the phrase separately, so we have **extracted all the words of the phrase**. As we have different words of the phrase, we are using a pre-trained **BERT Data Embedding**, to be more specific, we have used "paraphrase-MiniLM-L6-v2" of "SBERT" which is a pre-trained BERT model with descent performance and speed.

It is necessary to distinguish between the words that are significant and those that are not significant in a phrase since not all words in a phrase are not equally significant. We have **isolated all of the phrase's words** since, in order to achieve that, we would need to process each word independently. Because the phrase has distinct terms, we are utilizing a pre-trained **BERT Data Embedding** which is represented as X in the figure 6.1 at page 38. To be more precise, we chose "paraphrase-MiniLM-L6-v2" of "SBERT," a pre-trained BERT model with respectable performance and speed. After this layer, we are having a list of embedding of the words with post-padding and masking.

Below we provide a visualization of Sentence-Bert embedding. If we plot the CS and non-CS phrases embedding in 3D, then we see the figure 6.2 at page 40.

It does not look so good as we cannot segregate the data easily from seeing the plot. If we plot the CS and non-CS phrases embedding in 2D, then we see the output like the figure 6.3 at page 40.

In addition, we compared the cosine similarity of CS-CS, CS-NonCS, and NonCS-NonCS phrases (mentioned in figure 6.3 on page no 40). In this scenario, we can see distinct separa-

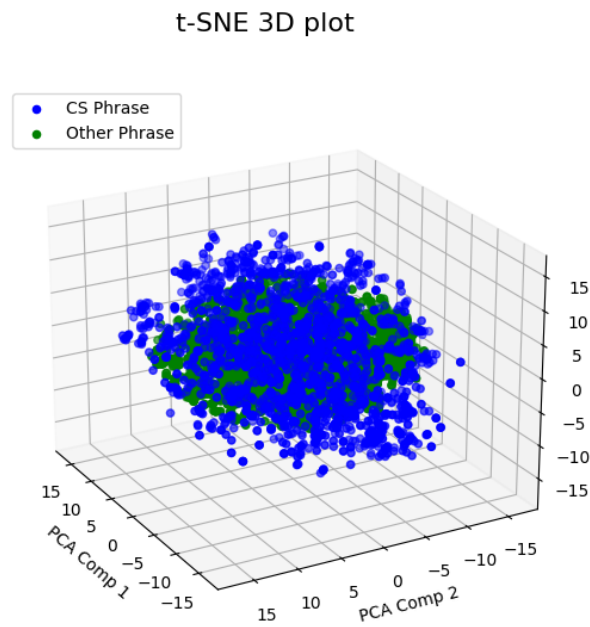


Figure 6.2. tSNE 3D plot of CS vs NonCS phrases

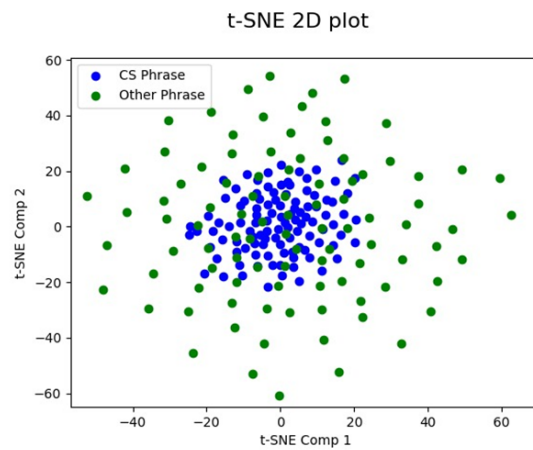


Figure 6.3. tSNE 2D plot of CS vs NonCS phrases

tions, which allows us to validate that the embedding retains some attribute that allows us to categorize phrases based on their kind.

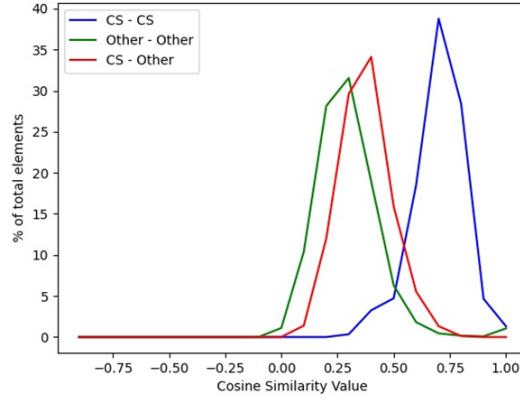


Figure 6.4. tSNE 2D plot of CS vs NonCS phrases

Bi-LSTM

Since phrases are sequential data, **Bi-LSTM** should be effective in our situation because it is ideally suited for classification problems involving sequential data. Also, we’re trying to figure out which part of the word is more important or contributes more to the relationship in this layer. This is where Bi-LSTM comes in. As our binary classifier, Bahdanau et al. [65] enhanced Hochreiter and Schmidhuber’s [66] work with an attention layer [67]. Our sentence representation’s sequential structure may be used to advantage by the Bi-LSTM model as we are also working with sequential data in this case.

Attention Layer

Additionally, using supervised learning, the attention layer [67] of the model will be taught to get the most important word among all the words. So, by looking at the attention layer, we might be able to figure out the most important parts of a sentence that can then be put together to make the syntactic pattern. The Bi-LSTM model generates a prediction of

the binary label for a phrase based on a vector-sequence representation of the phrase which is represented on the bottom part (L) in the figure 6.1 at page 38.

The vector sequence representation of a phrase, P, is the input to the Bi-LSTM, as seen in the bottom layer of Fig. X. This representation, indicated as X, includes K word embeddings in a sequence, each with dimension D, where K can be as high as 5 in our instance because we are considering phrases with a maximum of 5 words (coming from BERT embedding of words). A series of embeddings that individually represent a phrase as a matrix make up the vector representation of a phrase.

The vector sequence representation of a phrase, P, is the input to the Bi-LSTM, as seen in the bottom layer of Fig. 6.1 at page 38. Since we are examining a phrase with a maximum of 5 words, this representation, indicated as X, contains K (K can be max. 5) word embeddings in a sequence, each with dimension D. (coming from BERT embedding of words, for our case which is 384). The vector representation of a phrase is a set of embeddings, each of which represents a phrase as a matrix.

The vector sequence representation of a phrase, P, is the input to the Bi-LSTM, as seen in the bottom layer of Fig. 6.1 at page 38. Since we are examining a phrase with a maximum of 5 words, this representation, indicated as X, contains K (K can be max. 5) word embeddings in a sequence, each with dimension D. (coming from BERT embedding of words, for our case which is 384). The vector representation of a phrase is a set of embeddings, each of which represents a phrase as a matrix.

L takes $x_i \in X$ as input and provides 2 hidden state vectors as output.

1. Forward State Output, \vec{h}_i
2. Backward State Output, \tilde{h}_i

H is the summation of each h_i output coming from L for a single phrase representation X. So,

$$\begin{aligned}\vec{h}_i &= L(h_{i-1}, x_i) \\ \tilde{h}_i &= L(h_{i+1}, x_i) \\ h_i &= [\vec{h}_i + \tilde{h}_i]\end{aligned}$$

$$H = [h_1 + h_2 + h_3 + h_4 + h_5]$$

The single phrase representation, X is $K \times D$, where K is the total no of words in the phrase and D is the dimension of each word representation. So, when the given phrase is represented with X , the BiLSTM layer, L processes a concatenated output H of shape $K \times 2 \times n$ where n is the size of a single hidden vector.

The output \mathbf{H} is utilized as the input to the attention layer, \mathbf{Att} , which comes after the Bi-LSTM layer, \mathbf{L} . The output of attention layer A_t is a vector of size $K \times 1$ where each value, $a_i \in A_t$ is a value from range, $a_i \in [0, 1]$.

Each attention value, a_i will represent the importance or weight of a word in the meaning of the phrase as phrase embedding x_i which is significant in making binary classification decisions. A_t is compounded as bellow-

$$A_t = Softmax(Tanh(H * W_1)) * W_2$$

where W_1 is a trainable matrix shape $2 \times n \times 2 \times n$ and W_2 is another trainable matrix of shape $2 \times n$. So, we have applied a **Softmax** activation function to receive the value of A_t .

Output Layer

For the next layer, we are using both A_t and H as input for the representation vector, which has a shape of $K \times n \times n$ and output, R is the simple scalar multiplication of each hidden layer input $h_i \in H$ with corresponding attention value where $a_i \in A_t$.

Therefore, in the next layer R is used as input for the aggregation layer which computes the column-wise R in order to yield $2 \times n$ shape output, A_g where A_g outputs the weighted sum or attention given sum of H .

$$A_g = Summation(R)$$

With the input, A_g , coming from the previous layer, we use a fully linked dense layer in the final layer to forecast the output.

$$\hat{y} = ReLU(A_g)$$

6.2 Result and Output

This model's training accuracy is satisfactory. For instance, if we examine 1,828 samples, we obtain an AUC of 88%, which is satisfactory.

This process allows us to assess whether or not a phrase is a computer science phrase, and if it is not a computer science phrase, we remove it using this classifier filter. The only thing left is a list of computer science terms.

The categorized data is used to train our classification model. Figure 6.5 depicts the outcome of training, which reveals that our model ceased training after six epochs. In this instance, we are employing early stopping because the model would become overtrained and overfitted if we continue training.

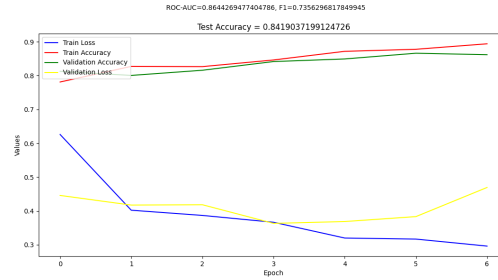


Figure 6.5. Classifier model Train Graph

Several evaluation metrics are used in key-phrase extraction [68][69]. Precision, recall, and F1 score are the three measures that researchers often choose due to their validity and usability. The same measures will be employed to assess our suggested technique.

Precision

[70][68] We utilize precision to quantify the algorithm's accuracy. When compared to the total number of extracted phrases, the precision value is represented as the percentage of accurately extracted phrases. Using the following formula, we can compute the precision [71][69]:

$$\begin{aligned}
 \text{Precision} &= \frac{\text{CorrectClassifiedCSPhrasesCount}}{\text{TotalExtractedPhrasesCount}} \\
 &= \frac{TP_s}{TP_s + FP_s}
 \end{aligned}$$

We are using 'Total Extracted Phrases Count' at the denominator because we are assuming all extracted phrases are CS phrases.

Recall

[70][68] We utilize the recall measure, which represents the percentage of successfully extracted words that are computer science terms, to assess how comprehensive the key-phrase extraction is. The yield is determined using the following formula [71][69]:

$$\begin{aligned} Recall &= \frac{CorrectCSPhrasesCount}{TotalCSPhraseInDataset} \\ &= \frac{TP_s}{TP_s + FN_s} \end{aligned}$$

F-1 Score

[68] Precision and recall frequently interact in the opposite way. The recall is poor when accuracy is high. The F1-score is utilized in this situation to combine recall and accuracy so that we may have a comprehensive understanding [71][69]. The yield is determined using the following formula-

$$F-1Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

6.2.1 Performance Comparison

As stated before, no prior work extracts Computer Science phrases from a specific document or text. Thus we compared our method to other keyword or phrase extractor techniques. Although the comparison is not fair since the methods are not tuned for computer science phrase extraction, we are utilizing the techniques to make the comparison. In table 6.5, we present the results of a comparison in which the text (algorithm input) consisted of phrases extracted from Wikipedia and utilized for data categorization.

Analyzing the outcome reveals that our proposed method performs much better than every other algorithm shown in the apparent table, table 6.5.

We are not comparing Recall since, in order to compare Recall, we require false-negative data, which is also unavailable. If we want this information, word extractors should extract

Table 6.5. Extraction method comparison

Algorithm name	Correct	Wrong	Precision
Yake	8,792	33,680	0.2070
Frake	385	3,973	0.0883
KeyBERT	0	0	undefined
TextRank	234	2552	0.0840
RAKE	15,872	46,029	0.2564
ACSPE	3,515	854	0.8044

every feasible phrase, and we need to label all possible phrases, which is a huge data-set to label with less contribution to the actual work, so we decided not to go for Recall. Therefore, Recall should be equal to Precision (if we count $FN = 0$), which adds no value. The Recall is not present in the result, so we are not calculating the F-1 score. If Precision and Recall are equal, the F-1 score should equal Precision, which adds no value.

7. CONCLUSIONS AND RECOMMENDATIONS

Our goal for this research work is to develop a deep learning-based semi-supervised model for computer science domain-specific terms and phrases extraction. To build the machine learning model, we have made some contributions as well. I tried to find a proper existing dataset that can be used in training my developed model. Before this work, there was no available machine-learning model which is extracting CS phrases from a given text. So, we were unable to find an existing dataset that was used before in computer science phrase extraction research. To solve this issue, I have developed our own dataset for the research with help of domain experts which is one of the critical tasks for this research. Our aim is to extract texts from a given text and we do not know what would be the total number of computer science phrases in a given text because the different text will have different computer science phrase that need to be extracted which lead us to choose an unsupervised model from various unsupervised text extraction models. Finding RAKE as the best term/phrase extraction method for computer science-related phrases from various experiments. As RAKE was unable to process some cases, we have added some layers over RAKE phrase extraction so that we can get better results from RAKE which is another contribution to this research work. In the phrase classification stage, we extracted different words of the phrase from where we made the embedding to get a better embedding that represents the underlying meaning of the phrase. We also applied an attention-based model to find the important words in a given phrase from which we can determine the nature of the phrase (CS or Non-CS). And at the last stage, we are able to determine the type of the extracted phrase provided by RAKE with help of the machine learning model.

The goal of this research is to extract computer science phrases from given text which can have the following points of recommendation and further investigation:

1. **Knowledge Graph Integration:** Since a data-set that worked with domain-specific terms is not available and we have made a dataset and our collected data-set is not very large, therefore if we can gather additional data, we can extract more attributes from the data and a connection from the data-set. If we can collect a large amount of data, we can create a knowledge graph from there with confidence.

2. **Collect phrases from online with Intelligent Model:** We can extend this model to collect more computer science phrases from internet by scrapping internet contents and running this model in those extracted contents.
3. **Auto knowledge graph edge creation and Verification:** In the knowledge graph, the extracted phrases from this model can be used as vertices. However, the knowledge graph will need edges to represent a proper knowledge graph. The edges can also be created and verified by machine learning models. If we can build this model, then we can automatically build a computer science knowledge graph from online content.
4. **Collaborate with different Knowledge Groups:** After developing a trustworthy knowledge graph on a modest scale, we prefer to collaborate with other research organizations to increase the number of participants. If we have additional researchers in the research group, we will be able to construct and validate the knowledge graph so that we can establish a widely acceptable computer science knowledge graph.
5. **Publish for Public Access and Public Collaboration:** Our main goal is to make data-sets available to the public so that they can be used in many different scientific studies. If we are able to do so, other researchers with similar interests will be able to benefit from our study, and more researchers will be interested in contributing to it, which is a fantastic chance to improve the computer science knowledge graph.

REFERENCES

- [1] *What is a Knowledge Graph?* [Online]. Available: https://web.stanford.edu/~vinayc/kg/notes/What_is_a_Knowledge_Graph.html.
- [2] *U.S. News & World Report Announces the 2023 Best Jobs*, publisher: UsNews. [Online]. Available: <https://www.usnews.com/info/blogs/press-room/articles/2023-01-10/u-s-news-world-report-announces-the-2023-best-jobs>.
- [3] *The Top Careers for 2023: These 20 Jobs Are In High Demand*, publisher: Bloomberg. [Online]. Available: <https://www.bloomberg.com/news/articles/2023-01-10/20-best-jobs-in-2023-fast-growing-high-paying-recession-proof-careers>.
- [4] M. Smith, *The 10 best U.S. jobs of 2023, according to new research many pay over \$100,000*, en. [Online]. Available: <https://www.cnbc.com/2023/01/10/the-10-best-us-jobs-of-2023-according-to-new-research.html>.
- [5] *2023's 100 best jobs in america : Us news*. [Online]. Available: <https://money.usnews.com/careers/best-jobs/rankings/the-100-best-jobs>.
- [6] *Software Developers, Quality Assurance Analysts, and Testers : Occupational Outlook Handbook: : U.S. Bureau of Labor Statistics*, en-us. [Online]. Available: <https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm>.
- [7] *NAS Report Investigates the Growth of Computer Science Undergraduate Enrollments*, en-US, Nov. 2017. [Online]. Available: <https://cra.org/crn/2017/11/nas-report-investigates-growth-computer-science-undergraduate-enrollments/>.
- [8] T. Camp, W. R. Adrion, B. Bizot, *et al.*, “Generation CS: The growth of computer science,” *ACM Inroads*, vol. 8, no. 2, pp. 44–50, May 2017, ISSN: 2153-2184. DOI: 10.1145/3084362. [Online]. Available: <https://doi.org/10.1145/3084362>.
- [9] *Why more colleges are offering data science programs : Us news*. [Online]. Available: <https://www.usnews.com/education/best-colleges/articles/why-more-colleges-are-offering-data-science-programs>.
- [10] J. Abdul-Alim, “Students not prepared for careers in computer science,” *Diverse Issues in Higher Education*, 2015.

- [11] D. Weintrop, “iSchools as Venues for Expanding the K-12 Computer Science Teacher Pipeline,” in *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*, ser. SIGCSE 2022, New York, NY, USA: Association for Computing Machinery, Feb. 2022, pp. 397–403, ISBN: 9781450390705. DOI: [10.1145/3478431.3499302](https://doi.org/10.1145/3478431.3499302). [Online]. Available: <https://doi.org/10.1145/3478431.3499302>.
- [12] N. A. o. Sciences and C. S. a. T. Board, *Assessing and Responding to the Growth of Computer Science Undergraduate Enrollments*, en. National Academies Press, Apr. 2018, Google-Books-ID: u4FUDwAAQBAJ, ISBN: 9780309467025.
- [13] J. Goode, “Increasing diversity in k-12 computer science: Strategies from the field,” in *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, ser. SIGCSE ’08, New York, NY, USA: Association for Computing Machinery, Mar. 2008, pp. 362–366, ISBN: 9781595937995. DOI: [10.1145/1352135.1352259](https://doi.org/10.1145/1352135.1352259). [Online]. Available: <https://doi.org/10.1145/1352135.1352259>.
- [14] A. Yadav, S. Gretter, S. Hambruch, and P. Sands, “Expanding computer science education in schools: Understanding teacher experiences and challenges,” *Computer Science Education*, vol. 26, no. 4, pp. 235–254, Dec. 2016, ISSN: 0899-3408. DOI: [10.1080/08993408.2016.1257418](https://doi.org/10.1080/08993408.2016.1257418). [Online]. Available: <https://doi.org/10.1080/08993408.2016.1257418>.
- [15] A. Munawar, M. Wahib, M. Munetomo, and K. Akama, “A Survey: Genetic Algorithms and the Fast Evolving World of Parallel Computing,” in *2008 10th IEEE International Conference on High Performance Computing and Communications*, Sep. 2008, pp. 897–902. DOI: [10.1109/HPCC.2008.77](https://doi.org/10.1109/HPCC.2008.77).
- [16] P. S. Buffum, E. V. Lobene, M. H. Frankosky, K. E. Boyer, E. N. Wiebe, and J. C. Lester, “A Practical Guide to Developing and Validating Computer Science Knowledge Assessments with Application to Middle School,” in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE ’15, New York, NY, USA: Association for Computing Machinery, Feb. 2015, pp. 622–627, ISBN: 9781450329668. DOI: [10.1145/2676723.2677295](https://doi.org/10.1145/2676723.2677295). [Online]. Available: <https://doi.org/10.1145/2676723.2677295>.
- [17] U. Fuller, C. G. Johnson, T. Ahoniemi, *et al.*, “Developing a computer science-specific learning taxonomy,” *ACM SIGCSE Bulletin*, vol. 39, no. 4, pp. 152–170, Dec. 2007, ISSN: 0097-8418. DOI: [10.1145/1345375.1345438](https://doi.org/10.1145/1345375.1345438). [Online]. Available: <https://doi.org/10.1145/1345375.1345438>.
- [18] L. Ehrlinger and W. WöSS, “Towards a definition of knowledge graphs,” in *SEMAN-TiCS*, 2016.

- [19] J. Bai, L. Cao, S. Mosbach, J. Akroyd, A. A. Lapkin, and M. Kraft, “From Platform to Knowledge Graph: Evolution of Laboratory Automation,” en, *JACS Au*, vol. 2, no. 2, pp. 292–309, Feb. 2022, ISSN: 2691-3704, 2691-3704. DOI: [10.1021/jacsau.1c00438](https://doi.org/10.1021/jacsau.1c00438). [Online]. Available: <https://pubs.acs.org/doi/10.1021/jacsau.1c00438>.
- [20] G. Kasneci, M. Ramanath, F. Suchanek, and G. Weikum, “The YAGO-NAGA approach to knowledge discovery,” *ACM SIGMOD Record*, vol. 37, no. 4, pp. 41–47, Mar. 2009, ISSN: 0163-5808. DOI: [10.1145/1519103.1519110](https://doi.org/10.1145/1519103.1519110). [Online]. Available: <https://doi.org/10.1145/1519103.1519110>.
- [21] *Knowledge Graphs*, en-GB. [Online]. Available: <https://www.dbpedia.org/resources/knowledge-graphs/>.
- [22] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell, “Toward an architecture for never-ending language learning.,” in *AAAI*, vol. 5, 2010, p. 3.
- [23] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, “Freebase: A collaboratively created graph database for structuring human knowledge,” in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, ser. SIGMOD ’08, New York, NY, USA: Association for Computing Machinery, Jun. 2008, pp. 1247–1250, ISBN: 9781605581026. DOI: [10.1145/1376616.1376746](https://doi.org/10.1145/1376616.1376746). [Online]. Available: <https://doi.org/10.1145/1376616.1376746>.
- [24] X. Dong, E. Gabrilovich, G. Heitz, *et al.*, “Knowledge vault: A web-scale approach to probabilistic knowledge fusion,” in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD ’14, New York, NY, USA: Association for Computing Machinery, Aug. 2014, pp. 601–610, ISBN: 9781450329569. DOI: [10.1145/2623330.2623623](https://doi.org/10.1145/2623330.2623623). [Online]. Available: <https://doi.org/10.1145/2623330.2623623>.
- [25] K. Thakur and V. Kumar, “Application of Text Mining Techniques on Scholarly Research Articles: Methods and Tools,” *New Review of Academic Librarianship*, vol. 28, no. 3, pp. 279–302, Jul. 2022, ISSN: 1361-4533. DOI: [10.1080/13614533.2021.1918190](https://doi.org/10.1080/13614533.2021.1918190). [Online]. Available: <https://doi.org/10.1080/13614533.2021.1918190>.
- [26] J. Yang, S. C. Han, and J. Poon, “A survey on extraction of causal relations from natural language text,” en, *Knowledge and Information Systems*, vol. 64, no. 5, pp. 1161–1186, May 2022, ISSN: 0219-3116. DOI: [10.1007/s10115-022-01665-w](https://doi.org/10.1007/s10115-022-01665-w). [Online]. Available: <https://doi.org/10.1007/s10115-022-01665-w>.

- [27] H. J. Kockaert and F. Steurs, *Handbook of Terminology*, en. John Benjamins Publishing Company, Mar. 2015, Google-Books-ID: MQZoBwAAQBAJ, ISBN: 9789027269560.
- [28] G. Tür, D. Hakkani-Tür, and K. Oflazer, “A statistical information extraction system for Turkish,” en, *Natural Language Engineering*, vol. 9, no. 2, pp. 181–210, Jun. 2003, ISSN: 1469-8110, 1351-3249. DOI: [10.1017/S135132490200284X](https://doi.org/10.1017/S135132490200284X). [Online]. Available: <https://www.cambridge.org/core/journals/natural-language-engineering/article/abs/statistical-information-extraction-system-for-turkish/7C288FAFC71D5F0763C1F8CE66464017>.
- [29] Y. Ko and J. Seo, “An effective sentence-extraction technique using contextual information and statistical approaches for text summarization,” en, *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1366–1371, Jul. 2008, ISSN: 0167-8655. DOI: [10.1016/j.patrec.2008.02.008](https://doi.org/10.1016/j.patrec.2008.02.008). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167865508000676>.
- [30] S. Miller, H. Fox, L. Ramshaw, and R. Weischedel, “A novel use of statistical parsing to extract information from text,” in *1st Meeting of the North American Chapter of the Association for Computational Linguistics*, 2000.
- [31] M. Garg, “A survey on different dimensions for graphical keyword extraction techniques,” en, *Artificial Intelligence Review*, vol. 54, no. 6, pp. 4731–4770, Aug. 2021, ISSN: 1573-7462. DOI: [10.1007/s10462-021-10010-6](https://doi.org/10.1007/s10462-021-10010-6). [Online]. Available: <https://doi.org/10.1007/s10462-021-10010-6>.
- [32] K. B. Cohen and L. Hunter, “Getting Started in Text Mining,” en, *PLoS Computational Biology*, vol. 4, no. 1, e20, 2008, ISSN: 1553-734X, 1553-7358. DOI: [10.1371/journal.pcbi.0040020](https://doi.org/10.1371/journal.pcbi.0040020). [Online]. Available: <https://dx.plos.org/10.1371/journal.pcbi.0040020>.
- [33] W. Pan, E. Zhong, and Q. Yang, “Transfer Learning for Text Mining,” en, in *Mining Text Data*, C. C. Aggarwal and C. Zhai, Eds., Boston, MA: Springer US, 2012, pp. 223–257, ISBN: 9781461432234. DOI: [10.1007/978-1-4614-3223-4_7](https://doi.org/10.1007/978-1-4614-3223-4_7). [Online]. Available: https://doi.org/10.1007/978-1-4614-3223-4_7.
- [34] A. Gupta, V. Dengre, H. A. Kheruwala, and M. Shah, “Comprehensive review of text-mining applications in finance,” *Financial Innovation*, vol. 6, no. 1, p. 39, Nov. 2020, ISSN: 2199-4730. DOI: [10.1186/s40854-020-00205-1](https://doi.org/10.1186/s40854-020-00205-1). [Online]. Available: <https://doi.org/10.1186/s40854-020-00205-1>.

- [35] F. Hogenboom, F. Frasincar, U. Kaymak, and F. De Jong, “An overview of event extraction from text.,” *DeRiVE@ ISWC*, pp. 48–57, 2011.
- [36] Z. Ji, H. Pi, W. Wei, B. Xiong, M. Woniak, and R. Damasevicius, “Recommendation Based on Review Texts and Social Communities: A Hybrid Model,” *IEEE Access*, vol. 7, pp. 40 416–40 427, 2019, ISSN: 2169-3536. DOI: [10.1109/ACCESS.2019.2897586](https://doi.org/10.1109/ACCESS.2019.2897586).
- [37] A. S. H. Basari, B. Hussin, I. G. P. Ananta, and J. Zeniarja, “Opinion Mining of Movie Review using Hybrid Method of Support Vector Machine and Particle Swarm Optimization,” en, *Procedia Engineering*, Malaysian Technical Universities Conference on Engineering & Technology 2012, MUCET 2012, vol. 53, pp. 453–462, Jan. 2013, ISSN: 1877-7058. DOI: [10.1016/j.proeng.2013.02.059](https://doi.org/10.1016/j.proeng.2013.02.059). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877705813001781>.
- [38] Z. Ghahramani, “Unsupervised Learning,” en, in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, ser. Lecture Notes in Computer Science, O. Bousquet, U. von Luxburg, and G. Rätsch, Eds., Berlin, Heidelberg: Springer, 2004, pp. 72–112, ISBN: 9783540286509. DOI: [10.1007/978-3-540-28650-9_5](https://doi.org/10.1007/978-3-540-28650-9_5). [Online]. Available: https://doi.org/10.1007/978-3-540-28650-9_5.
- [39] *What is unsupervised learning?* [Online]. Available: <https://www.ibm.com/topics/unsupervised-learning>.
- [40] *Introduction to Machine Learning and Three Common Algorithms*, en, Jul. 2021. [Online]. Available: <https://bootcamp.pe.gatech.edu/blog/introduction-to-machine-learning-and-three-common-algorithms/>.
- [41] A. Graves and J. Schmidhuber, “Framewise phoneme classification with bidirectional LSTM and other neural network architectures,” eng, *Neural Networks: The Official Journal of the International Neural Network Society*, vol. 18, no. 5-6, pp. 602–610, 2005, ISSN: 0893-6080. DOI: [10.1016/j.neunet.2005.06.042](https://doi.org/10.1016/j.neunet.2005.06.042).
- [42] R. L. Abduljabbar, H. Dia, and P.-W. Tsai, “Development and evaluation of bidirectional LSTM freeway traffic forecasting models using simulation data,” *Scientific Reports*, vol. 11, p. 23 899, Dec. 2021, ISSN: 2045-2322. DOI: [10.1038/s41598-021-03282-z](https://doi.org/10.1038/s41598-021-03282-z). [Online]. Available: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8668885/>.

- [43] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, arXiv:1810.04805 [cs], May 2019. DOI: [10.48550/arXiv.1810.04805](https://doi.org/10.48550/arXiv.1810.04805). [Online]. Available: <http://arxiv.org/abs/1810.04805>.
- [44] S. Ravichandiran, *Getting Started with Google BERT: Build and train state-of-the-art natural language processing models using BERT*, en. Packt Publishing Ltd, Jan. 2021, Google-Books-ID: CvsWEAAAQBAJ, ISBN: 9781838826239.
- [45] C. Sun, X. Qiu, Y. Xu, and X. Huang, “How to Fine-Tune BERT for Text Classification?” en, in *Chinese Computational Linguistics*, M. Sun, X. Huang, H. Ji, Z. Liu, and Y. Liu, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2019, pp. 194–206, ISBN: 9783030323813. DOI: [10.1007/978-3-030-32381-3_16](https://doi.org/10.1007/978-3-030-32381-3_16).
- [46] S. Yu, J. Su, and D. Luo, “Improving BERT-Based Text Classification With Auxiliary Sentence and Domain Knowledge,” *IEEE Access*, vol. 7, pp. 176 600–176 612, 2019, ISSN: 2169-3536. DOI: [10.1109/ACCESS.2019.2953990](https://doi.org/10.1109/ACCESS.2019.2953990).
- [47] K. S. Hasan and V. Ng, “Automatic Keyphrase Extraction: A Survey of the State of the Art,” en, in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Baltimore, Maryland: Association for Computational Linguistics, 2014, pp. 1262–1273. DOI: [10.3115/v1/P14-1119](https://doi.org/10.3115/v1/P14-1119). [Online]. Available: <http://aclweb.org/anthology/P14-1119>.
- [48] R. Campos, V. Mangaravite, A. Pasquali, A. M. Jorge, C. Nunes, and A. Jatowt, “YAKE! Collection-Independent Automatic Keyword Extractor,” en, in *Advances in Information Retrieval*, G. Pasi, B. Piwowarski, L. Azzopardi, and A. Hanbury, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2018, pp. 806–810, ISBN: 9783319769417. DOI: [10.1007/978-3-319-76941-7_80](https://doi.org/10.1007/978-3-319-76941-7_80).
- [49] A. Zehtab-Salmasi, M.-R. Feizi-Derakhshi, and M.-A. Balafar, “FRAKE: Fusional Real-time Automatic Keyword Extraction,” 2021. DOI: [10.48550/ARXIV.2104.04830](https://doi.org/10.48550/ARXIV.2104.04830). [Online]. Available: <https://arxiv.org/abs/2104.04830>.
- [50] *MaartenGr/KeyBERT: BibTeX*, Jan. 2021. DOI: [10.5281/ZENODO.4461265](https://doi.org/10.5281/ZENODO.4461265). [Online]. Available: <https://zenodo.org/record/4461265>.
- [51] Y. Vasiliev, *Natural Language Processing with Python and spaCy: A Practical Introduction*, en. No Starch Press, May 2020, Google-Books-ID: IVv6DwAAQBAJ, ISBN: 9781718500525.

- [52] Channabasamma, Y. Suresh, and A. Manusha Reddy, “A Contextual Model for Information Extraction in Resume Analytics Using NLPs Spacy,” en, in *Inventive Computation and Information Technologies*, S. Smys, V. E. Balas, K. A. Kamel, and P. Lafata, Eds., ser. Lecture Notes in Networks and Systems, Singapore: Springer, 2021, pp. 395–404, ISBN: 9789813343054. DOI: [10.1007/978-981-33-4305-4_30](https://doi.org/10.1007/978-981-33-4305-4_30).
- [53] C. Chantrapornchai and A. Tunsakul, “Information extraction on tourism domain using spacy and bert,” *ECTI Trans. Comput. Inf. Technol*, vol. 15, no. 1, pp. 108–122, 2021.
- [54] M. M. Haider, M. A. Hossin, H. R. Mahi, and H. Arif, “Automatic Text Summarization Using Gensim Word2Vec and K-Means Clustering Algorithm,” in *2020 IEEE Region 10 Symposium (TENSYP)*, ISSN: 2642-6102, Jun. 2020, pp. 283–286. DOI: [10.1109/TENSYP50017.2020.9230670](https://doi.org/10.1109/TENSYP50017.2020.9230670).
- [55] J. C. Dunn, “A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters,” en, *Journal of Cybernetics*, vol. 3, no. 3, pp. 32–57, Jan. 1973, ISSN: 0022-0280. DOI: [10.1080/01969727308546046](https://doi.org/10.1080/01969727308546046). [Online]. Available: <http://www.tandfonline.com/doi/abs/10.1080/01969727308546046>.
- [56] H. Zhao and Q. Xie, “An Improved TextRank Multi-feature Fusion Algorithm For Keyword Extraction of Educational Resources,” *Journal of Physics: Conference Series*, vol. 2078, no. 1, p. 012 021, Nov. 2021, ISSN: 1742-6588, 1742-6596. DOI: [10.1088/1742-6596/2078/1/012021](https://doi.org/10.1088/1742-6596/2078/1/012021). [Online]. Available: <https://iopscience.iop.org/article/10.1088/1742-6596/2078/1/012021>.
- [57] F. Barrios, F. López, L. Argerich, and R. Wachenchauser, *Variations of the Similarity Function of TextRank for Automated Summarization*, arXiv:1602.03606 [cs], Feb. 2016. DOI: [10.48550/arXiv.1602.03606](https://doi.org/10.48550/arXiv.1602.03606). [Online]. Available: <http://arxiv.org/abs/1602.03606>.
- [58] S. Rose, D. Engel, N. Cramer, and W. Cowley, “Automatic Keyword Extraction from Individual Documents,” en, in *Text Mining*, M. W. Berry and J. Kogan, Eds., Chichester, UK: John Wiley & Sons, Ltd, Mar. 2010, pp. 1–20, ISBN: 9780470689646 9780470749821. DOI: [10.1002/9780470689646.ch1](https://doi.org/10.1002/9780470689646.ch1). [Online]. Available: <https://online.library.wiley.com/doi/10.1002/9780470689646.ch1>.
- [59] *Doc ũ spaCy API Documentation*, en. [Online]. Available: <https://spacy.io/api/doc?#ents>.

- [60] R. Mihalcea and P. Tarau, “Textrank: Bringing order into text,” in *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004, pp. 404–411.
- [61] E. Partalidou, E. Spyromitros-Xioufis, S. Doropoulos, S. Vologianidis, and K. Diamantaras, “Design and implementation of an open source Greek POS Tagger and Entity Recognizer using spaCy,” in *IEEE/WIC/ACM International Conference on Web Intelligence*, ser. WI ’19, New York, NY, USA: Association for Computing Machinery, Oct. 2019, pp. 337–341, ISBN: 9781450369343. DOI: [10.1145/3350546.3352543](https://doi.org/10.1145/3350546.3352543). [Online]. Available: <https://doi.org/10.1145/3350546.3352543>.
- [62] J. Ramos *et al.*, “Using tf-idf to determine word relevance in document queries,” in *Proceedings of the first instructional conference on machine learning*, New Jersey, USA, vol. 242, 2003, pp. 29–48.
- [63] S. R. El-Beltagy and A. Rafea, “Kp-miner: Participation in semeval-2,” in *Proceedings of the 5th international workshop on semantic evaluation*, 2010, pp. 190–193.
- [64] N. Reimers and I. Gurevych, “Sentence-bert: Sentence embeddings using siamese bert-networks,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Nov. 2019. [Online]. Available: <https://arxiv.org/abs/1908.10084>.
- [65] D. Bahdanau, K. Cho, and Y. Bengio, *Neural Machine Translation by Jointly Learning to Align and Translate*, arXiv:1409.0473 [cs, stat], May 2016. DOI: [10.48550/arXiv.1409.0473](https://arxiv.org/abs/1409.0473). [Online]. Available: <http://arxiv.org/abs/1409.0473>.
- [66] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, ISSN: 0899-7667. DOI: [10.1162/neco.1997.9.8.1735](https://doi.org/10.1162/neco.1997.9.8.1735).
- [67] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017. [Online]. Available: <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547de91fbd053c1c4a845aa-Abstract.html>.
- [68] F. Liu, X. Huang, W. Huang, and S. X. Duan, “Performance evaluation of keyword extraction methods and visualization for student online comments,” English, Nov. 2020, ISSN: 2073-8994. [Online]. Available: <https://opus.lib.uts.edu.au/handle/10453/148215>.

- [69] L. Ajallouda, F. Z. Fagroud, A. Zellou, and E. B. Lahmar, “KP-USE: An Unsupervised Approach for Key-Phrases Extraction from Documents,” en, *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 4, 2022, ISSN: 21565570, 2158107X. DOI: [10.14569/IJACSA.2022.0130433](https://doi.org/10.14569/IJACSA.2022.0130433). [Online]. Available: <http://thesai.org/Publications/ViewPaper?Volume=13&Issue=4&Code=IJACSA&SerialNo=33>.
- [70] S. Jones, S. Lundy, and G. Paynter, “Interactive document summarisation using automatically extracted keyphrases,” in *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, Jan. 2002, pp. 1160–1169. DOI: [10.1109/HICSS.2002.994038](https://doi.org/10.1109/HICSS.2002.994038).
- [71] E. Papagiannopoulou and G. Tsoumakas, *A Review of Keyphrase Extraction*, arXiv:1905.05044 [cs], Jul. 2019. DOI: [10.48550/arXiv.1905.05044](https://doi.org/10.48550/arXiv.1905.05044). [Online]. Available: <http://arxiv.org/abs/1905.05044>.