Reducing the dependency of having prior domain knowledge for effective online information retrieval

Omar Zammit*[1] | Serengul Smith[2] | David Windridge[2] | Clifford De Raffaele[1]

[1]Faculty of Computer Science, Middlesex University, Pembroke, Malta
[2]Faculty of Computer Science, Middlesex University, London, UK

Correspondence
*Omar Zammit,
Middlesex University
Pembroke Malta.
Email: ozammi@ieee.org

Abstract

Sometimes Internet users struggle to find what they are looking for on the Internet due to information overload. Search engines intend to identify documents related to a given keyphrase on the Internet and provide suggestions. Having some background knowledge about a topic or a domain will help in building effective search keyphrases that will lead to accurate results in information retrieval. This is further pronounced amongst students that rely on the internet to learn about a new topic. Students might not have the required background knowledge to build effective keyphrases and find what they are looking for. In this research, we are addressing this problem, and aim to help students find relevant information online. This research furthers existing literature by enhancing information retrieval frameworks through keyphrase assignment, aiming to expose students to new terminologies, therefore reducing the dependency of having background knowledge about the domain under study. We evaluated this framework and identified how it can be enhanced to suggest more effective search keyphrases. Our proposed suggestion is to introduce a keyphrase Ranking Mechanism that will improve the keyphrase assignment part of the framework by taking into consideration the part-of-speech of the generated keyphrases. To evaluate the proposed approach, various datasets were downloaded and processed. The results obtained showed that our proposed approach produces more effective keyphrases than the existing framework.

KEYWORDS:
Keyphrase Assignment, Information Retrieval, Part-of-Speech Tagging, and Dictionary Accuracy.

1 | INTRODUCTION

Nowadays, knowledge has more importance within the information society and information is more accessible than ever (Vivekavardhan, Chakravarthy, & Ramesh 2020). Information overload as explained in Mahdi, Ahmad, Ismail, Natiq, and Mohammed (2020) refers to the difficulty of identifying relevant content within a large amount of content. Search engines have their deficiencies, but they try to identify documents related to a given keyphrase on the Internet, thus assisting with information overload (Cheng & Tsai 2017). Having some background about the topic being searched will help Internet users to build useful search keyphrases and find what they are looking for while searching online (Chen 2020; Monchaux, Amadieu, Chevalier, & Mariné 2015; Sanchiz et al. 2017). The rationale behind this research is to assist students during online information retrieval. Students are considered a novice in the domain under study since they are still learning and therefore might lack the background knowledge required to build useful search keyphrases (Tsai 2009).

To address this issue, in this research we implemented an existing framework that was proposed in Zammit, Smith, Windridge, and De Raffaele (2020) that amongst other components, has a *Keyphrase assignment mechanism* that generates keyphrases pertinent to the domain being searched by a student.

This framework aims to expose students to new terminologies and, therefore, assist students who lack background knowledge about the domain being researched. During this research, experiments showed that this framework can be improved by modifying the keyphrases being extracted and selected. Our proposed approach considers the part-of-speech tag of each word within the keyphrases to decide which keyphrases are more pertinent to the domain. The proposed approach was evaluated using pre-build datasets containing documents and keyphrases selected by their authors. The datasets were processed using both the original framework and our approach. The results indicate that our approach increased the accuracy of the quality of the extracted keyphrases. Following a brief background in Section 2 on existing studies, the framework and the requirements analysis are explained in Section 3 and Section 4. The suggested modifications are explained in Section 5. Lastly a conclusion is drawn in Section 7.

2 | BACKGROUND

Internet users have so much information online, that sometimes they struggle to find what they are looking for (Kraft 2002). For the last two decades, the problem of information overload has been addressed since it affects our day-to-day activities (Mahdi et al. 2020). The use of a search engine is common amongst Internet users because although they are biased and target a generic audience (Introna & Nissenbaum 2000) they try to identify documents related to a given keyphrase on the Intrenet (Cheng & Tsai 2017) and ease information retrieval. The search process to retrieve some information about a topic includes, creating a keyphrase consisting of words related to the topic, submitting the keyphrase to the search engine, and visiting websites that surface in the result (J. Y. Kim, Collins-Thompson, Bennett, & Dumais 2012). The process is repeated until the required information about the topic is found (Usta, Altingovde, Vidinli, Ozcan, & Ulusoy 2014). During each repetition, an Internet user will refine the keyphrase and seek to make a more effective search. Students are also Internet users who rely on search engines during their studies to learn more about a topic. By the time students will get acquainted more with the search process, they will improve their searching strategy and create more effective keyphrases (Kilbride & Mangina 2005). Having the right keyphrases sometimes is still not enough in information retrieval, because not all keyphrases can generate relevant results (Sendurur & Yildirim 2015). Besides, other qualities are required, for example, patience (Wu & Cai 2016), the capability of locating relevant content (Zhou 2015) and prior domain knowledge (Monchaux et al. 2015; Sanchiz et al. 2017). Having background knowledge about the domain being researched is an advantage to find relevant information online (Chen 2020; Monchaux et al. 2015; Sanchiz et al. 2017) because the more acquainted one is with a domain, the more effective are the keyphrases. Since students are still learning, they are considered a novice in the domain under study (Tsai 2009), and therefore they might lack background knowledge. Various studies tried to address this problem by proposing a *Keyphrase assignment* framework (Zammit, Smith, De Raffaele, & Petridis 2019) capable of suggesting keyphrases that one can use to perform effective online searches. Keyphrases consist of words included as parameters within a query string and submitted to search engines or websites with search capability (Usta et al. 2014). The query string will contain various information about the search request, including the keyphrase searched by the student. The location of the searched keyphrase within a URL is dependent on the website being requested. For example, the below list shows how Google and Stackoverflow include the keyphrase searched by the student in the query string, while Wikipedia appends the searched keyphrase to the URL:

- https://www.google.com/search?q=expert+system&oq=expert+system&aqs=chrome.1.7777&sourceid=chrome&ie=UTF-8
- https://stackoverflow.com/search?q=expert+system
- https://en.wikipedia.org/wiki/expert_system

Extracting such keyphrases might be challenging because the techniques to parse the URL and extract the keyphrase is not common for all visited websites. In this research, we created a function, outlined in Listing 1, that is using some Python libraries to extract keyphrases from different URLs. This function will return the keyphrase searched by the student if the URL is valid and matches one of the rules defined within the function.

Listing 1: Extracting keyphrases from URLs

```
import re
from urllib import parse

def extract_keyphrase_from_url(url: str) -> str:
    keyword = ""
```

```
        query_strings = dict(parse.parse_qsl(parse.urlsplit(url).query))
        if re.search("google.com(.*)/search", url) and 'q' in query_strings:
            # URL is a Google URL, get keyword.
            keyword = query_strings['q'].lower()
        elif re.search("scholar.google.com", url)  and 'q' in query_strings:
            # URL is a Google scholar URL, get keyword.
            keyword = query_strings['q'].lower()
        elif re.search("wikipedia.org/wiki/", url):
            # URL is a Wikipedia URL, get keyword.
            keyword = url.split('/')[-1].replace("_", " ")
        elif re.search("stackoverflow.com/", url) and 'q' in query_strings:
            keyword = query_strings['q'].lower()
        elif re.search("stackoverflow.com/", url):
            keyword = url.split('/')[-1].replace("-", " ")
        return keyword.lower()

    # Calling the method with different URLs. The output should always be 'expert system' in this example.
    google_url = "https://www.google.com.mt/search?q=expert+system&sxsrf=wiz&ved=0ah736537sjc&uact"
    print(extract_keyphrase_from_url(google_url))

    google_scholar = "https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=expert+system&btnG="
    print(extract_keyphrase_from_url(google_scholar))

    wikipedia = "https://en.wikipedia.org/wiki/Expert_System"
    print(extract_keyphrase_from_url(wikipedia))

    stackoverflow = "https://stackoverflow.com/search?q=expert+system&s=4585-f4445f-1454-9260-cc5fc04772"
    print(extract_keyphrase_from_url(stackoverflow))
```

In text mining such keyphrases are also referred to as N-grams (Ribeiro, Henrique, Ribeiro, & Neto 2017) and various research has been done to explore their use, including for non-English languages (Ahmad, Rub Talha, Ruhul Amin, & Chowdhury 2018; Gledec, Soic, & Dembitz 2019). A possible solution for an effective keyphrase assignment framework is to gather data and train a classification algorithm, but some authors outlined this as a disadvantage since it has a dependency on external documents (Gledec et al. 2019) and training requires time and effort. *Educational Search Engines* targeting a specific domain has also been suggested (Vidinli & Ozcan 2016) but this might be a problem since students are already trusting a search engine of their choice, convincing them to move away from it might be challenging (Cheng & Tsai 2017; Zammit et al. 2019). A framework that collects URLs visited by a student and using various similarity analysis finds previously searched keyphrases by the student was proposed in Zammit et al. (2019). This study was extended in Zammit et al. (2020) with a framework that amongst others includes *Keyphrase assignment* functionality and can suggest new terminologies pertinent to the domain being researched by the student. *Keyphrase assignment* is done using a *Keyphrase extraction function* that can generate keyphrases of various lengths and a *Ranking mechanism* that is based on the keyphrases occurrence.

Although the *Keyphrase assignment* process suggested in Zammit et al. (2020) can retrieve effective keyphrases, the experiments in this research showed that it can be improved by taking into consideration the part-of-speech tags of the keyphrase words while performing keyphrase ranking. Having higher accuracy will lead to more effective keyphrases that novice students can use to find relevant information about a topic. Part-of-speech taggers have been available for a long time (Brill 1995), since keyphrases in search engine queries are not structured, some used part-of-speech to understand the semantic structure of a query and improve information retrieval (Li 2010). Keyphrases can have some unique structure, in Barr, Jones, and Regelson (2008) the authors identified tags for various English language web search-engine queries and noticed that the majority of the tags constitute Nouns. Several other studies showed a degree of success when using part-of-speech tags (Barr et al. 2008; Chowdhury & McCabe 1998; Dinçer & Karaouglan 2004; Zukerman & Raskutti 2002).

3 | FRAMEWORK OVERVIEW

To understand what students are searching for and assist them in finding information online, one must collect data about their browsing activities and searched keyphrases (Feild, Allan, & Glatt 2011; Usta et al. 2014; Vidinli & Ozcan 2016). In this research, we implemented a system based on the framework developed by Zammit et al. (2020) and included additional functionality. We opted for this approach since the aim of the existing framework was to assist students during information retrieval and it contains all the processes required to gather, store and process data. Figure 1 shows the framework's main components and how they interact with each other.

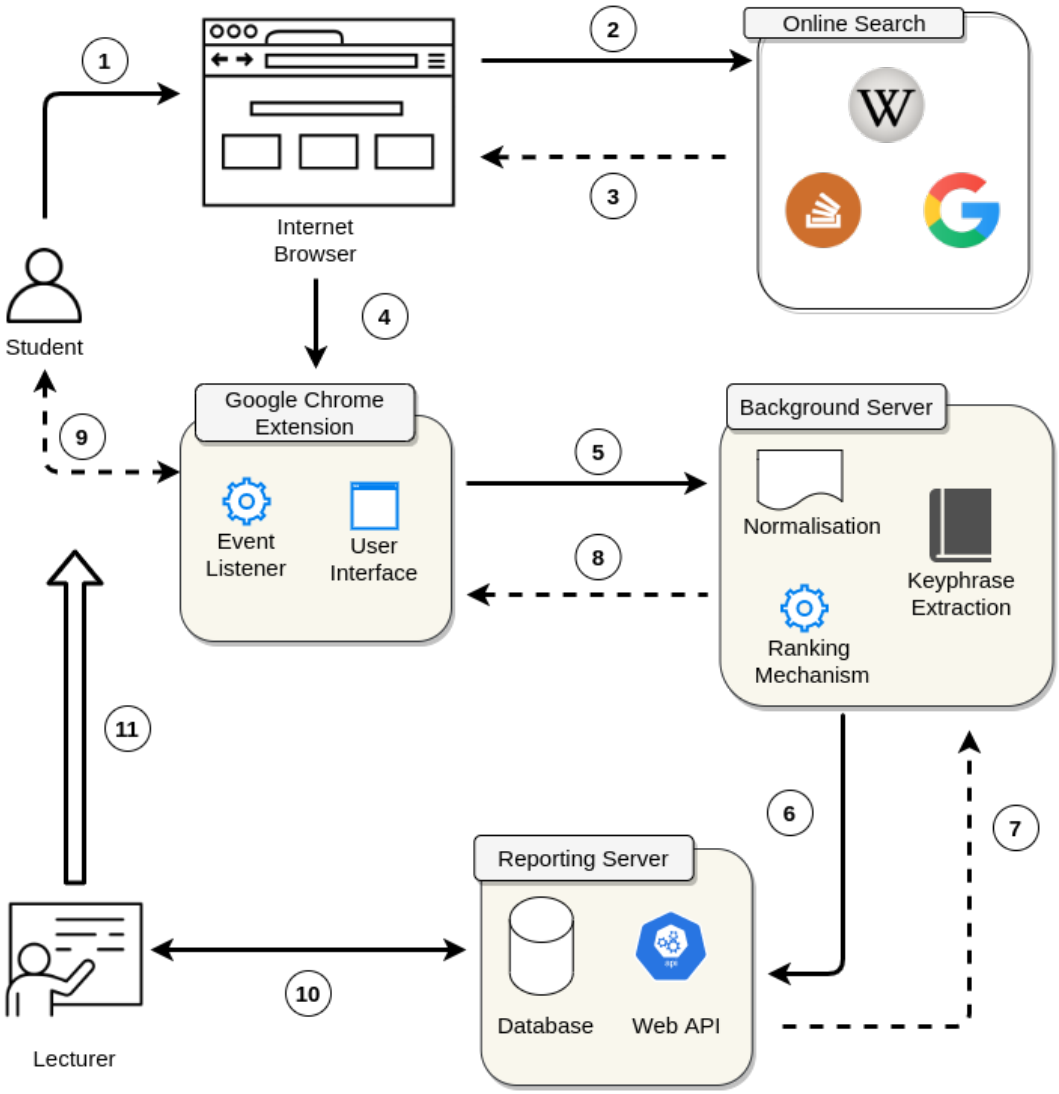

FIGURE 1 Framework implemented in this resarch Zammit et al. (2020)

As shown in Figure 1, the framework presented consists of three main components; a *Google Chrome extension*, a *Background Server* and a *Reporting server*. The user interface of the extension was structured using HTML and CSS, while the background logic was developed in Javascript. The *Background server* and the *Reporting server* were implemented using Python and MySQL as the database engine. The framework provides the correct functionality to; collect keyphrases searched by students, process requests, predict suitable content, and display results. The process of data collection starts when a student visits a website that allows searching, like Wikipedia, Stackoverflow, or Google (1). Such websites allow students to search for content using keyphrases related to the topic being searched (2). Upon receiving a search request, the website will respond

with a Search Engine Result Page (SERP), that is, a webpage containing results pertinent to the search done by the student (3). The *Event listener* will automatically be notified that a search was done and the SERP content and the URL are collected (4) and sent to the *Background server* for further processing (5).

Using various normalisation techniques as explained by Zammit et al. (2019) the *Background Server* will clean the SERP HTML content and will extract automatically various keyphrases using a *Keyphrase extraction function*. Since this function can tend to extract a large number of keyphrases, a *Ranking mechanism* is used to select the top 10 most relevant keyphrases. In addition, the *Background server* sends the keyphrase searched and the URL visited by the student to the *Reporting server* using a web API (6). The web API will store the keyphrase and the URL in a centralised database and will return a list of similar keyphrases searched by other students (7). The *Google Chrome extension* will then display the keyphrases extracted and the similar keyphrases in the *User interface* (8) so that students can view and interact with the results (9).

Since the *Reporting server* is accessible using a web API, lecturers can interact with the collected data, perform data mining techniques and learn more about their student searching strategies. This approach is beneficial, since a lecturer can assist students to refine the keyphrases searched and support students in obtaining better search results (11). In this research, we are referring to this as the *'feedback loop'* since lecturers can manage keyphrases and provide feedback to students.

The *Keyphrase extraction function* can generate keyphrases of various lengths and has an important role in *Keyphrase assignment*. This is because it exposes students to new terminologies pertinent to a topic being searched by the student thus reducing the dependency on background knowledge. Since the function generates various keyphrases, to avoid the *Curse of dimensionality* (Fan & Fan 2008) and not end up with a large amount of keyphrases, in Zammit et al. (2020) a *Ranking mechanism* was suggested. This mechanism selects the top 10 most pertinent keyphrases generated from the keyphrases extracted by the function. The decision is based on the occurrence of a keyphrase, that is, the number of times the keyphrase appears in the original text. The mechanism works as follows:

1 Function generates a list of keyphrases having different lengths.

2 For every keyphrase the frequency of the keyphrase is counted and a list containing the keyphrase and its frequency is constructed $L = [(k_1, f_1), (k_2, f_2), \cdots, (k_{n-1}, f_{n-1})(k_n, f_n)]$.

3 Calculate the mean of all frequencies as $\bar{x} = \text{mean}([f_1, f_2, \cdots, f_{n-1}, f_n])$.

4 Remove keyphrases from L where f is less than $\bar{x}$.

5 Sort keyphrases L by frequency f in descending order and select top 10 keyphrases.

The main focus of this research is to reduce the dependency of having prior domain knowledge about a topic by suggesting better results than the framework proposed by Zammit et al. (2020). To achieve this, we improved the *Ranking mechanism* accuracy by taking into consideration the part-of-speech tags within a keyphrase and included a *Reporting server* that amongst other functionality it allows keyphrase collaboration between students. In addition, we opted also to improve the *User interface* component since we received a lot of suggestions from students while evaluating to improve the user experience.

4 | EMPIRICAL ANALYSIS

Keyphrase assignment is the process that can help to reduce the dependency on having prior domain knowledge about a specific topic being searched for since it intends to expose students to new terminologies that the student never searched before (Zammit et al. 2020). Since improving such a process is one of the main aims of this research we conducted an experiment to analyse and understand if this process can be improved, focusing mainly on the *Ranking mechanism*.

We implemented the *Keyphrase extraction function* and the *Ranking mechanism* as proposed by Zammit et al. (2020) as shown in Figure 2, but configured the function to extract keyphrases with different word counts.In our implementation, we opted to generate keyphrases from unigrams to 4-grams, since this is the recommended range to use for this framework (Zammit et al. 2020). The function assumes that the text being processed for keyphrase assignment is normalised, converted to lower case, without stop words, without punctuation (Hu, Tang, Gao, & Liu 2013) and containing only words having more than 3 characters. Lemmatisation or stemming was not included in the text since we need to keep the words in their original format to build valid keyphrases. In addition, sample documents having approximately 150 words were downloaded from the Internet and normalised, these will be processed by the function to extract keyphrases.

During the first experiment, the sample documents were processed by the *Keyphrase extraction function* and the top keyphrases were selected using the *Ranking mechanism* as suggested in Zammit et al. (2020) and explained in Section 3). In this experiment, the *Keyphrase extraction function*

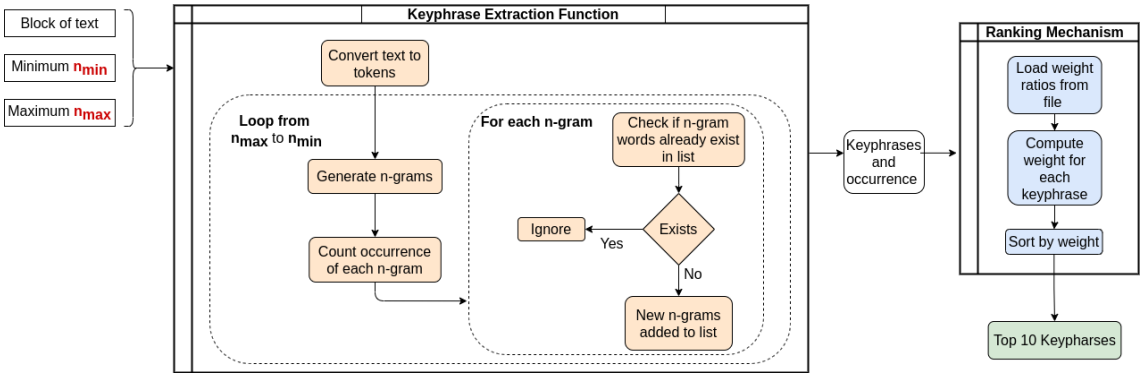

FIGURE 2 Modified *Keyphrase extraction function* and *Ranking mechanism*

TABLE 1 Keyphrases Extracted During First Experiment

	Original function keyphrases			
	1-gram	2-gram	3-gram	4-gram
1	cards	insertion sort		
2	pseudocode	left hand		
3	algorithm	real code		
4	array			
5	number			
6	table			
7	sorted			

was configured to extract keyphrases having between 1 and 4 words and to select the top 10 most relevant keyphrases. An example list of keyphrases is shown in Table 1.

During the second experiment we wanted to investigate what is the impact of the *Ranking mechanism* on the quality of the results. Therefore, we disabled the *Ranking mechanism* and modified the *Keyphrase extraction function* to retrieve all keyphrases identified irrespective of their occurrence frequency and number of words in the keyphrase. Table 2 shows an example list of the keyphrases obtained when the filtering by the ranking mechanism was disabled. Naturally, this increased the number of keyphrases displayed to the students, irrespective of their relevance.

From both experiments conducted it was observed that:

1 Keyphrases that were extracted during the first experiment were still present in the second experiment.

2 Some valid keypharses extracted during the second experiment were not present in the first experiment. Therefore the current ranking mechanism proposed in Zammit et al. (2020) can omit valid keyphrases.

3 If the ranking mechanism is disabled completely from the framework, the list of displayed extracted keyphrases will increase drastically (in this example by a factor of 3), thus making it difficult for students to browse through the list.

These observations outline the necessity of the ranking mechanism within the framework since its absence will lead to large and impractical amount of keyphrases. Nevertheless, it was concluded that the tuning of this component is critical since the ranking mechanism directly leads to a better selection of keyphrases that are more suitable for students to research the sought-after domain.

5 | PROPOSED SOLUTION

5.1 | Proposed Keyphrase Extraction Function

The experiment described in Section 4 showed that the original *Keyphrase extraction function* and *Ranking mechanism* ignore valid keyphrases that occur less than the mean occurrence. Therefore. in this research, we propose a new *Ranking mechanism* that, instead of relying only on a keyphrase

TABLE 2 Keyphrases Extracted During the Second Experiment

	Proposed function keyphrases			
	1-gram	2-gram	3-gram	4-gram
1	time	procedure called	**pseudocode real code**	empty **left hand cards**
2	separates	denoted length	illustrated figure times	start **insertion sort** efficient
3	section	sometimes clearest		takes parameter **array** containing
4	engineering	another difference		convey essence **algorithm** concisely
5	playing	originally top		sorting small **number** elements
6	insert	stored outside		face **table** remove one
7	compare			sorts input **number**s place
8	already			contains **sorted** output sequence
9	right			find correct position card
10	held			employ whatever expressive method
11	pile			phrase sentence embedded within
12	present			typically concerned issues software
13	rearranges			english surprised come across
14	constant			clear concise specify given
15	finished			data abstraction modularity error

occurrence, considers the keyphrase words part-of-speech tags. And in order to achieve this, we modified the original *Keyphrase extraction function* to return all keyphrases extracted no matter their occurrence and then apply the new Ranking Mechanism suggested in this research. Figure 2 shows the proposed *Keyphrase extraction function* and the *Ranking mechanism*.

When the *Keyphrase extraction function* proposed in this research is called, the framework will pass three parameters, a block of text, the minimum (n_{min}) and maximum (n_{max}) n-grams to take in consideration. The function will loop from n_{max} to n_{min} and for each iteration n-grams will be generated according to the iteration value. To avoid repetitive words, for each generated keyphrase, the function checks if the words contained in the new keyphrase exist in the keyphrases already extracted by the function. If the keyphrase exists in a longer keyphrase, then the new keyphrase is ignored. The final output, that is a list of keyphrases and their occurrence is then forwarded to the *Ranking mechanism*, where the weight is computed, sorted, and the top 10 keyphrases are returned. Keyphrases extracted are then displayed in the form of a link in the *Google Chrome extension* to enable students to view and navigate by clicking on the keyphrase.

5.2 | Using Part-of-Speech in Ranking Mechanism

The *Natural Language Toolkit* Python library (Bird, Steven & Ewan Klein 2009), uses the Penn Treebank annotated corpus to assign tags to words within a sentence. The library has more than 50 tags (Marcus, Santorini, & Marcinkiewicz 1993) were as stated in Barr et al. (2008) some tags are more common than others in query keyphrases. Since tags determine the grammatical structure of a keyphrase (Barr et al. 2008), in this research we used them to understand the structure of the extracted keyphrases and improve the *Ranking mechanism*. In Barr et al. (2008) the authors stated that some tags are common within keyphrases. If one determines what tags are most popular within keyphrases, then the *Ranking mechanism* can be implemented to give priority to keyphrases having the popular tags. Tagging functionality is implemented in the *Natural Language Toolkit* and an example of how tags are identified is depicted in Listing 2. When the library function `pos_tag(List: tokens)` is called with a list of tokens, the function will assign a part-of-speech tag for each word. `help.upenn_tagset(string: tag)` displays information about a given tag.

Listing 2: NLTK tag sentence example

```
import nltk
# Sample sentence
sentence = "John is feeling good"
# Tokenize the sentence into words
tokens = nltk.word_tokenize(sentence)
# Tag tokens with part-of-speech
tags = nltk.pos_tag(tokens)
```

```
# Display the word and information about the tag
for word, tag in tags:
    print(f"Word: {word}")
    nltk.help.upenn_tagset(tag)

# Sample Output:
Word: John
NNP: noun, proper, singular
    Motown Venneboerger Czestochwa Ranzer Conchita Trumplane Christos
    Oceanside Escobar Kreisler Sawyer Cougar Yvette Ervin ODI Darryl CTCA
    Shannon A.K.C. Meltex Liverpool
Word: is
VBZ: verb, present tense, 3rd person singular
    bases reconstruct marks mixes displeases seals carps weaves snatches
    slumps stretches authorises smolders pictures emerges stockpiles
    seduces fizzes uses bolsters slaps speaks pleads
Word: feeling
VBG: verb, present participle or gerund
    telegraphing stirring focusing angering judging stalling lactating
    hankerin' alleging veering capping approaching traveling besieging
    encrypting interrupting erasing wincing
Word: good
JJ: adjective or numeral, ordinal
    third ill-mannered pre-war regrettable oiled calamitous first separable
    ectoplasmic battery-powered participatory fourth still-to-be-named
    multilingual multi-disciplinary
```

In Barr et al. (2008) the authors outlined that 40% of the query terms created by internet users are made up of proper nouns and 70% are proper nouns and nouns together. This indicates that there is a pattern and the validity of a keyphrase depends on its tag structure. If keyphrases that are known as valid are analysed and the popular tags are determined, one can use such information to assess the validity of newly generated keyphrases. The popularity of a tag can be computed as a weight, in this research, we refer to this as the *weight-ratio* and it can be used to determine the importance of a particular tag. To determine the *weight-ratio* of different part-of-speech tags, we downloaded various datasets (listed in Table 4), consisting of documents and key files with keyphrases selected by the document author. Since an author selects the keyphrases, we used these are ground truth to compute the *weight-ratio*. All keyphrases were extracted and using the Python *Natural Language Toolkit* library (Loper & Bird 2002) we tokenised each keyphrase and extracted a list of words. For each word we determined the part-of-speech tag using the `nltk.pos_tag()` function in the library and calculated the *weight-ratio*. The *weight-ratio* r_p for a specific tag p is computed as shown in Equation 1:

$$r_p = \frac{x_p}{y} \tag{1}$$

Where x_p is the total number of words tagged as p and y is the total number of words extracted from the author keyphrases. Once the *weight-ratio* of all tags were computed, the result was saved into a file to make it available to *Ranking mechanism* when needed. Figure 3 shows the identified tags and their respective *weight-ratio*.

The proposed *Ranking mechanism* was implemented to take into consideration the keyphrase words *weight-ratios*. As shown in Figure 2 once the *Keyphrase extraction function* extracts the list of keyphrases, the final list consisting of keyphrases and their occurrence are forwarded to the *Ranking mechanism*. The *Ranking mechanism* loads the list of *weight-ratios* from file and computes the ranking weight for each keyphrase. If k is a keyphrase that is made up of words $[w_1, w_2, \cdots, w_{n-1}, w_n]$. Then for each word the part-of-speech *weight-ratio* is found and grouped in a list $[r_1, r_2, \cdots, r_{n-1}, r_n]$. If O_k is the occurrence of the selected keyphrase k in a block of text and N_k is the sum of all keyphrases occurrence. Then the final weight W_k used to sort keyphrases extracted by the function can be computed as shown in Equation 2.

$$W_k = \sum_{i=1}^{n} r_{p_i} \times \frac{O_k}{N_k} \tag{2}$$

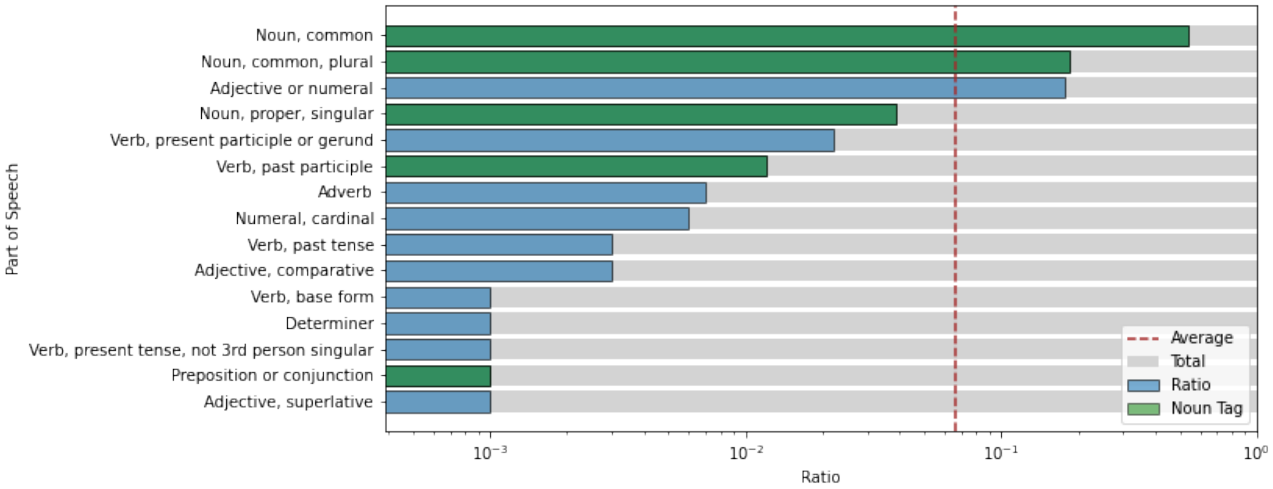

FIGURE 3 Part-of-speech weights based on datasets.

The final list will be sorted by the ranking weight and the top 10 keyphrases are selected. The higher the weight associated with a keyphrase the higher the probability for the keyphrase to be added within the top 10. Listing 3 shows how a keyphrase is tokenised and how after checking the validity of the word, the *weight-ratio* is assigned for each word. Table 3 shows an example of a block of text that was processed using the original and the new *Ranking mechanism*. The table shows that when the new *Ranking mechanism* was applied, the majority of the keyphrases changed their sorting order while some did not even make it in the top 10. As explained in Section 5.3 the modifications we did to the *Ranking mechanism* improved the overall accuracy of the framework.

Listing 3: Compute ratio

```
import pickle
import nltk
# Load dictionary from file
pos_ratio = pickle.load(open("dictionary.dta", "rb"))
sentence = "Max is a  good dog"
# Tokenize the sentence into words
tokens = nltk.word_tokenize(sentence)
# Tag tokens with part-of-speech
tags = nltk.pos_tag(tokens)
for word, tag in tags:
    ratio = 0
    # Check if the word is in the corpus
    if nltk.corpus.wordnet.synsets(word):
        ratio = pos_ratio[tag] if tag in pos_ratio.keys() else 0
    print(f"{word:5} ({tag:3}) -> {ratio}")

# Sample Output:
# Max   (NNP) -> 0.039
# is    (VBZ) -> 0.0
# a     (DT ) -> 0.001
# good  (JJ ) -> 0.177
# dog   (NN ) -> 0.541
```

TABLE 3 Comparing the original and the proposed *Ranking mechanisms* results

Ranking Mechanism		Change	Keyphrase	Ranking Weight	
Proposed	Original			Original	**Proposed**
1	2	↑	start insertion sort efficient	93	25.6
2	1	↓	empty left hand cards	104	12.0
3	3	-	pseudocode real code	74	11.8
4	5	↑	convey essence algorithm concisely	45	9.3
5	4	↓	takes parameter array containing	48	6.7
6	9	↑	contains sorted output sequence	36	5.8
7	10	↑	find correct position card	34	5.4
8	7	↓	face table remove one	38	5.3
9	6	↓	sorting small number elements	43	5.0
10	11	↑	employ whatever expressive method	29	4.6
11	16	↑	data abstraction modularity error	19	4.4
12	8	↓	sorts input numbers place	36	4.2
13	12	↓	phrase sentence embedded within	29	4.0
14	13	↓	typically concerned issues software	26	3.0
15	18	↑	works way many people	19	2.6
16	15	↓	clear concise specify given	19	1.8
17	17	-	handling often ignored order	19	1.4
18	19	↑	illustrated figure times	10	1.1
19	14	↓	english surprised come across	23	1.0
20	20	-	procedure called	6	0.4

Ranking Mechanism: Proposed The ranking number that was assigned by the new Ranking Mechanism.
Ranking Mechanism: Original The ranking number that was assigned by the original Ranking Mechanism as proposed in Zammit et al. (2020).
Change Depicts the change in order when the new weight was applied.
Ranking Weight: Original The number of times the keyphrase occurred in a block of text. Also the weight used by the original Ranking Mechanism.
Ranking Weight: Proposed The new weight as proposed in this research multiplied by 100.

5.3 | Keyphrase Assignment: Measuring Accuracy

The evaluation aimed to determine if the accuracy improved when the new *Ranking mechanism* was introduced in the framework. Various datasets were downloaded (refer to Table 4) that consist of documents and keyphrases selected by their respective author. The methodology adopted was to process the datasets and compare the accuracy of the original framework as suggested in Zammit et al. (2020) and the proposed approach suggested in this research. The keyphrases extracted by both frameworks were matched and compared to the ones selected by the authors. If the keyphrases extracted from a dataset document by both functions are explicitly compared with the keyphrases chosen by the author of the document, one will encounter very low accuracy. The reason for this is that some author words might not appear in the document (Witten, Paynter, Frank, Gutwin, & Nevill-Manning 1999) and therefore both functions can't include these words in the extracted keyphrases. To avoid this and still measure the quality of the keyphrases, the *Dictionary accuracy* was used as an accuracy measure, since this provides a value that reflects the number of words both functions managed to identify that exist in the author words. When a function accuracy is measured, the approach compiles a dictionary of unique words within the author keyphrases D_k. Furthermore, a separate dictionary is compiled with the unique works from the top 10 extracted keyphrases by the framework D_f. Once both dictionaries are identified, the words that intersect between both dictionaries are used to measure the accuracy, computed as shown in Equation 3

$$\text{accuracy} = \left(\frac{|D_k \cap D_f|}{|D_k|}\right) * 100 \tag{3}$$

We processed the datasets documents using the original framework and our proposed approach and measured the dictionary accuracy for both. The results are listed in Table 4. As seen in the result, there is an improvement in accuracy when our approach was applied. This shows that the part-of-speech *weight-ratio* did a positive impact on the results when included in the *Ranking mechanism*. Whilst the evaluation was undertaken on

the dataset sizes available in literature so as to provide a comparable metric, the proposed framework is not sensitive to the dataset size provided therefore the framework could be used with equal effectiveness on larger datasets.

TABLE 4 Evaluation results: Comparing original approach and the proposed approach

Dataset	Total Files (Keyphrases)		Extracted Keyphrases	Original Approach		Proposed Approach		
				Dictionary	Accuracy	Dictionary	Accuracy	
Gollapalli and Caragea (2014)	755	3093	1543	1827	14.63	5927	18.98	+4.35
Hulth (2003)	2000	28220	7191	8820	15.58	27782	31.27	+15.69
S. N. Kim, Medelyan, Kan, and Baldwin (2010)	243	3785	2430	7246	40.26	9031	40.94	+0.68
Krapivin (2008)	2304	12296	23040	69180	55.44	85959	55.63	+0.19
Nguyen and Kan (2007)	209	2507	2090	5609	51.64	7785	54.90	+3.26
Schutz and Others (2008)	1231	55718	12304	34245	29.52	47329	35.11	+5.59
Witten et al. (1999)	29	236	290	1001	37.60	1119	39.59	+1.99

Dataset Dataset reference.
Total Files The total files contained in the dataset.
Author Keyphrases The total author keypharses in a dataset.
Extracted Keyphrases The total extracted keyphrases by the *Keyphrase extraction function* a maximum of 10 was specified.
Original Approach The function and Ranking Mechanism as proposed by Zammit et al. (2020).
Proposed Approach The function and Ranking Mechanism proposed in this study using part-of-speech and occurrence weight.
Dictionary The total number of words identified by a function.
Accuracy The dictionary accuracy obtained when the extracted keyphrases were compared with the author keyphrases.

5.4 | Maximum Keyphrase Word Count

Different studies recommend the number of words to use in a keyphrase (Ahmad et al. 2018; Gledec et al. 2019; Zammit et al. 2020). In this research, we wanted to understand how the size of a keyphrase impacts accuracy and performance. As seen in Figure 2 the *Keyphrase extraction function* can be configured to generate keyphrases between n_{min} and n_{max}. While n_{min} will always start from one, we conducted an experiment to determine how the n_{max} affects the accuracy of the framework. We processed all the documents in the dataset using our proposed approach and measured the accuracy for an n_{max} ranging from 1 to 7. Figure 4 shows how an increase in the number of words will affect accuracy and performance. It was noted that the accuracy and the time taken are directly proportional to the maximum number of words. This means that longer keyphrases will lead to high accuracy, since they have a higher variety of words they will have a higher probability of intersecting with authors' words. In addition, the *Keyphrase assignment* will take more time to generate longer keyphrases and thus longer keyphrases will reduce performance.

Changing n_{max} from 1 to 2 words, increased the accuracy substantially. After these iterations, the fluctuation in accuracy decreased and thus, in this research, we opted to choose an n_{max} of 4 since it is a good balance between accuracy and time taken. In addition, having keyphrases longer than 4 was leading to a cluttered wordcloud in the *Google Chrome extension* making it difficult for students to understand.

6 | IMPLEMENTATION

6.1 | Google Chrome Extension

The proposed *Ranking mechanism* was implemented based on the framework suggested by Zammit et al. (2020) as shown in Figure 1. The *Reporting server* was hosted online so that it can collect data over the Internet while the extension and the *Background server* were distributed to ten students. Such students were instructed to install and use the suggestions predicted by the extension during their lectures and studies. Based on this feedback, the extension was improved as shown in Figure 5. The *Activity* tab shown in Figure 5 left screenshot, enables students to; view a list of their last searched keyphrases and also a list of auto-generated keyphrases extracted by the *Keyphrase extraction function* and sorted by the

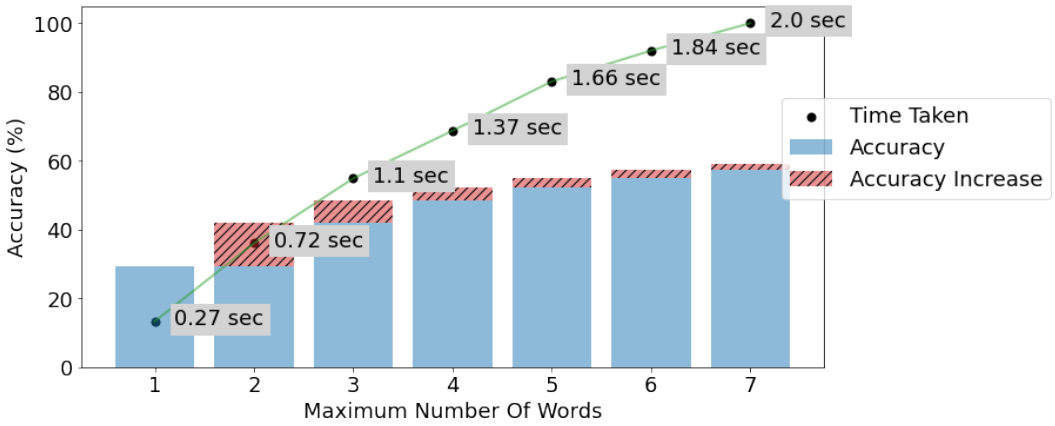

FIGURE 4 Relationship between n-grams and increase in accuracy.

Ranking mechanism proposed in this research. Similarly, the *Suggestions* tab shown in Figure 5 right screenshot shows similar keyphrases searched by other students using dynamic SQL queries computed by the *Background server*.

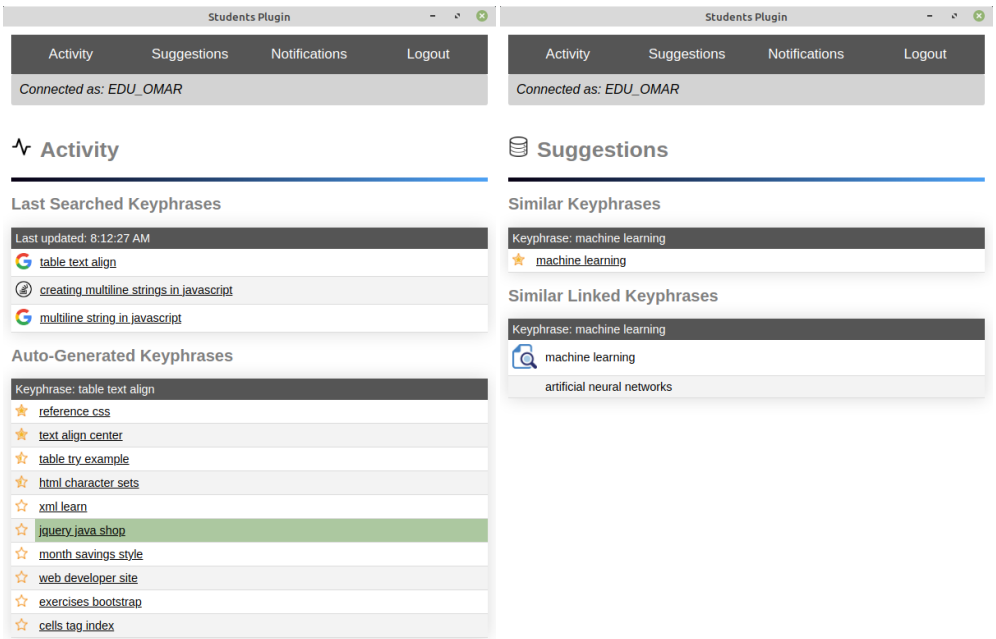

FIGURE 5 Google Chrome Extension Tabs

6.2 | Callback Functions

Within the framework *Callback* functions were implemented and used to measure how much students rely on the *Google Chrome extension* during their research to enrich their search. When a student clicks a suggestion within the extension, a new Google search with the clicked keyphrase is triggered to surface a result page as shown in Figure 6. The action is captured by the *Background Server* and is sent to the *Reporting server*. The *Reporting server* will save such requests in the database for reporting. In order to identify that such URLs are suggestions clicked by the student, callback data is added into the query string. The following URL shows an example of a callback containing the Callback ID (cbs) and the

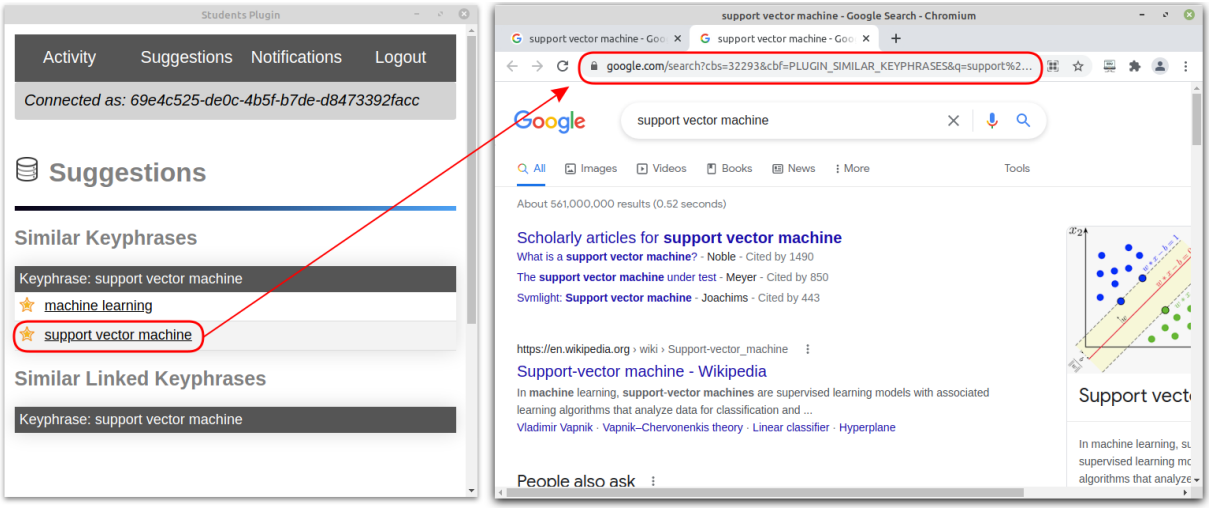

FIGURE 6 Google Chrome Extension Tabs

Callback function (cbf), these are the database ID of the keyphrase searched by the student and from where it was clicked within the user interface respectively:

https://www.google.com/search?cbs=1250&cbf=PLUGIN_SIMILAR_KEYPHRASES&q=support%20vector%20machine

Parameter *cbs* contains the ID of the keyphrase searched by the student that suggested 'support vector machine', in this scenario 1250 is the ID of keyphrase 'machine learning'. Parameter *cbf* contains information from where the suggested keyphrase was clicked, in this scenario it was clicked from the 'Similar Keyphrases' section. The following list shows the call back functions that can occur while using the framework:

a) PLUGIN LAST SEARCHES: The callback is triggered when a student clicks a keyphrase from the *Last Searched Keyphrases*.

b) PLUGIN EXTRACTED KEYPHRASES: The callback is triggered when a student clicks a keyphrase from the *Auto-Generated Keyphrases*.

c) PLUGIN SIMILAR KEYPHRASES: The callback is triggered when a student clicks a keyphrase from the *Similar Keyphrases*.

d) PLUGIN SIMILAR LINKED KEYPHRASES: The callback is triggered when a student clicks a keyphrase from the *Similar Linked Keyphrases*.

Callback function data contributes to the evaluation of the proposed framework because one can understand how students are using the proposed framework and also measure the relevance and quality of the predicted results. Figure 7 depicts the callback function source and outlines the fact that students relied more on the results predicted by the framework by clicking suggestions from the list of keyphrases extracted by the *Keyphrase extraction function* rather than other suggestions, including keyphrases searched by other students (Similar keyphrases). In addition, Figure 7 shows that students refer to their own previously searched keyphrases (Last searches), this means that students are not only using the *Google Chrome extension* to find new keyphrases but also as a tool to keep track of their previously searched keyphrases.

6.3 | Similar Linked Keyphrase

In order to improve the student's experience, the framework is taking into consideration the callback function calls to improve the suggested results. In this research, we are assuming that if a student clicks a suggested keyphrase, it means that the student identified some value in the keyphrase. Therefore, the framework is using the callback functions to validate the results and enrich the suggestions.

The process undertaken to support the student through keyword search, depicted in Figure 8, outlines how the proposed framework enables the provision of similarly related keyphrases and how the student interaction is used to expose students to new terminologies. It is important to note that the source keyphrase, that is the keyphrase searched by the student, has been normalised to cater for various combinations. For example, the terms 'machine learning', 'learning machine learning', or 'learning machine' will lead to the same source keyphrase that is 'learn machine'. The source keyphrase is normalised to its lemma, duplicate words are removed and remaining words are sorted alphabetically.

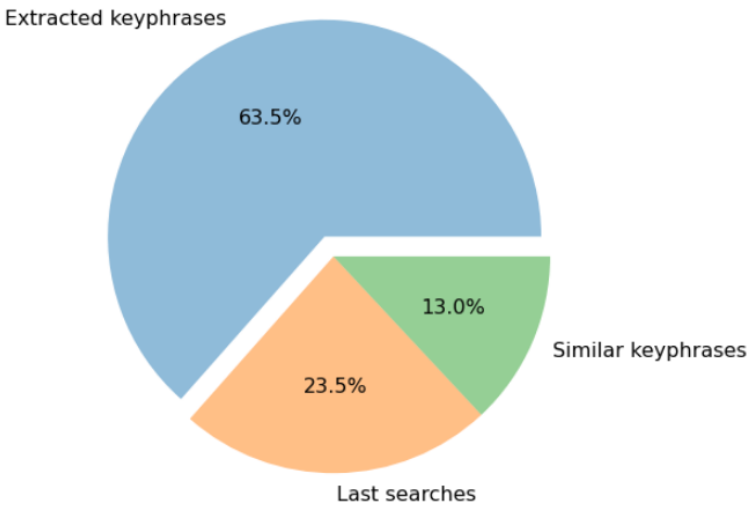

FIGURE 7 Google Chrome Extension callback function response

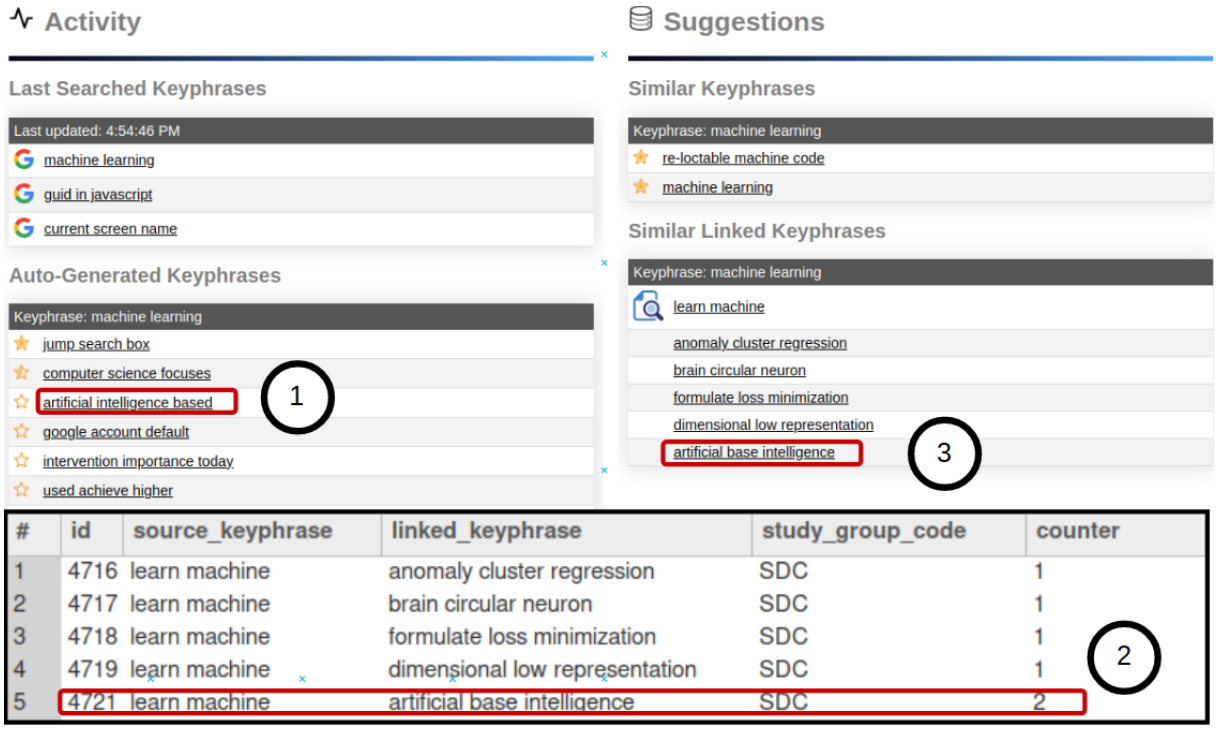

FIGURE 8 Process for students.(1) The system predicts result based on students keyword search. (2) Student clicks a suggestion and the link is stored. (3) Linked keyphrases are visible to all students

6.4 | Implementation Discussion

Students contributing to the framework evaluation were instructed to use the *Google Chrome extension* and provide feedback not only on the results suggested by the framework but also on their experience. Although the results proved to be very helpful, some students encountered delays while waiting for the results to pre-populate. The reason behind this was that the *Background server* was required to do various tasks to process, compute and populate the extension with results. To eliviate this, a thread-pool was introduced in the *Background server* to handle each result on a different thread. The size of the thread pool is equal to the number of processors available on the server. Although this solution did not provide a real-time feel, since the server needs time to compute the results, the solution improved the timing to display all the predictions into the extension.

Another consideration was taken into account during the evaluation is that since students are clicking on the results and clicked keyphrases are considered while suggesting content to other students. There is a possibility that a student might click a suggestion by mistake or click suggestions that are not valid to the domain being researched. Although this cannot be handled in real-time, to mitigate such a situation, a portal was introduced where lecturers can moderate and validate such keyphrases by deleting invalid entries.

Since the whole framework is based on anonymisation of students, which are only identified by a random unique identifier, therefore data mining in relation with other data sources is difficult. That is, the data gathered by the framework cannot be used in line with other datasets for example students' marks or test results to extract more information.

7 | CONCLUSION

To build effective keyphrases that will lead to valid results while searching online, background knowledge about the topic or domain is required. Having prior domain knowledge will assist students to build rich keyphrases but unfortunately, this fact can be lacking since students are still learning and are considered a novice in the domain under study. In this research, we have addressed this problem by extending an existing framework dealing with assisting students while doing online research. The framework described in Zammit et al. (2020) collects student browsing data and already implements a *Keyphrase assignment* functionality that can generate keyphrases. In this research, we believe that this functionality is crucial to assist students and reduce the dependency on having background knowledge since the functionality will suggest keyphrases pertinent to a domain. Therefore we are focusing on this functionality and improve its accuracy by adding context to the generated keyphrases.

The proposed solution is improving the *Ranking mechanism*, that is, how important keyphrases are selected within the framework. A new weight that takes into consideration the part-of-speech of a keyphrase was introduced and evaluated. Various datasets were downloaded and processed using the original, and the proposed approach was suggested. The accuracy obtained was compared, and the results showed how the new *Ranking mechanism* improves the accuracy of the keyphrases, leading to more effective keyphrases. Besides, we did some experiments to understand how the number of keyphrases generated affects the accuracy and the time taken by the function to complete. It is concluded that longer keyphrases will lead to better accuracy but will reduce performance.

8 | BIBLIOGRAPHY

Ahmad, A., Rub Talha, M., Ruhul Amin, M., & Chowdhury, F. (2018, nov). Pipilika N-Gram Viewer: An Efficient Large Scale N-Gram Model for Bengali. In *2018 international conference on bangla speech and language processing, icbslp 2018*. Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/ICBSLP.2018.8554474

Barr, C., Jones, R., & Regelson, M. (2008). The linguistic structure of English web-search queries. In *Emnlp 2008 - 2008 conference on empirical methods in natural language processing, proceedings of the conference: A meeting of sigdat, a special interest group of the acl* (pp. 1021–1030). Stroudsburg, PA, USA: Association for Computational Linguistics. doi: 10.3115/1613715.1613848

Bird, Steven, E. L., & Ewan Klein. (2009). *Natural Language Processing with Python*. O'Reilly Media Inc.

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, *21*(4), 543–565. doi: 10.1155/2013/410609

Chen, K. T. C. (2020). University efl students' use of online english information searching strategy. *Iranian Journal of Language Teaching Research*, *8*(1), 111–127.

Cheng, Y. H., & Tsai, C. C. (2017). Online research behaviors of engineering graduate students in Taiwan. *Educational Technology and Society*, *20*(1), 169–179.

Chowdhury, A., & McCabe, M. (1998). Improving Information Retrieval Systems using Part of Speech Tagging..

Dinçer, B. T., & Karaouglan, B. (2004). he Effect of Part-of-Speech Tagging on IR Performance for Turkish. In C. Aykanat, T. Dayar, & b. Körpeo\uglu (Eds.), *Computer and information sciences - iscis 2004* (pp. 771–778). Berlin, Heidelberg: Springer Berlin Heidelberg.

Fan, J., & Fan, Y. (2008). HIGH-Dimensional classification using features annealed independence rules. *Annals of Statistics*, *36*(6), 2605–2637. doi: 10.1214/07-AOS504

Feild, H., Allan, J., & Glatt, J. (2011). CrowdLogging: Distributed, private, and anonymous search logging. *SIGIR'11 - Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 375–384. doi: 10.1145/2009916.2009969

Gledec, G., Soic, R., & Dembitz, S. (2019). *Dynamic N-Gram system based on an online croatian spellchecking service* (Vol. 7). Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/ACCESS.2019.2947898

Gollapalli, S. D., & Caragea, C. (2014). Extracting keyphrases from research papers using citation networks. In *Proc. natl. conf. artif. intell.* (Vol. 2, pp. 1629–1635).

Hu, X., Tang, J., Gao, H., & Liu, H. (2013). Unsupervised sentiment analysis with emotional signals. In *Www 2013 - proceedings of the 22nd international conference on world wide web* (pp. 607–617). doi: 10.1145/2488388.2488442

Hulth, A. (2003). Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on empirical methods in natural language processing* (pp. 216–223). USA: Association for Computational Linguistics. Retrieved from https://doi.org/10.3115/1119355.1119383 doi: 10.3115/1119355.1119383

Introna, L. D., & Nissenbaum, H. (2000). Shaping the web: Why the politics of search engines matters. *Inf. Soc.*, *16*(3), 169–185. doi: 10.1080/01972240050133634

Kilbride, J., & Mangina, E. (2005). Automated keyphrase extraction: Assisting students in the search for online materials. In P. S. Szczepaniak, J. Kacprzyk, & A. Niewiadomski (Eds.), *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (Vol. 3528 LNAI, pp. 225–230). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/11495772_35

Kim, J. Y., Collins-Thompson, K., Bennett, P. N., & Dumais, S. T. (2012). Characterizing Web content, user interests, and search behavior by reading level and topic. In *Wsdm 2012 - proc. 5th acm int. conf. web search data min.* (pp. 213–222). New York, NY, USA: ACM. doi: 10.1145/2124295.2124323

Kim, S. N., Medelyan, O., Kan, M.-Y., & Baldwin, T. (2010, jul). {S}em{E}val-2010 Task 5 : Automatic Keyphrase Extraction from Scientific Articles. In *Proceedings of the 5th international workshop on semantic evaluation* (pp. 21–26). Uppsala, Sweden: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/S10-1004

Kraft, R. (2002). A maschine learning approach to improve precision for navigational queries in a Web information retrieval system.

Krapivin, M. (2008). *Large Dataset for Keyphrase Extraction* (Tech. Rep. No. May 2008). University of Trento.

Li, X. (2010). Understanding the semantic structure of noun phrase queries. In *Proceedings of the 48th annual meeting of the association for computational linguistics* (pp. 1337–1345).

Loper, E., & Bird, S. (2002). NLTK: The Natural Language Toolkit. *arXiv preprint cs/0205028*. Retrieved from http://arxiv.org/abs/cs/0205028 doi: 10.3115/1118108.1118117

Mahdi, M. N., Ahmad, A. R., Ismail, R., Natiq, H., & Mohammed, M. A. (2020). Solution for Information Overload Using Faceted Search–A Review. *IEEE Access*, *8*, 119554–119585. doi: 10.1109/ACCESS.2020.3005536

Marcus, M., Santorini, B., & Marcinkiewicz, M. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational linguistics - Association for Computational Linguistics (Print)*, *19*(2), 313–330. Retrieved from https://www.aclweb.org/anthology/J93-2004

Monchaux, S., Amadieu, F., Chevalier, A., & Mariné, C. (2015). Query strategies during information searching: Effects of prior domain knowledge and complexity of the information problems to be solved. *Information Processing and Management*, *51*(5), 557–569. doi: 10.1016/j.ipm.2015.05.004

Nguyen, T. D., & Kan, M. Y. (2007). Extraction in scientific publications. In D. H.-L. Goh, T. H. Cao, I. T. Sølvberg, & E. Rasmussen (Eds.), *Lect. notes comput. sci. (including subser. lect. notes artif. intell. lect. notes bioinformatics)* (Vol. 4822 LNCS, pp. 317–326). Berlin, Heidelberg: Springer Berlin Heidelberg.

Ribeiro, J., Henrique, J., Ribeiro, R., & Neto, R. (2017, jun). NoSQL vs relational database: A comparative study about the generation of the most frequent N-grams. In *2017 4th international conference on systems and informatics, icsai 2017* (Vol. 2018-Janua, pp. 1568–1572). Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/ICSAI.2017.8248535

Sanchiz, M., Chin, J., Chevalier, A., Fu, W. T., Amadieu, F., & He, J. (2017). Searching for information on the web: Impact of cognitive aging, prior domain knowledge and complexity of the search problems. *Information Processing and Management*, *53*(1), 281–294. doi: 10.1016/j.ipm.2016.09.003

Schutz, A. T., & Others. (2008). Keyphrase extraction from single documents in the open domain exploiting linguistic and statistical methods. *Masters of Applied Science M. App. Sc.*

Sendurur, E., & Yildirim, Z. (2015, feb). Students' Web Search Strategies With Different Task Types: An Eye-Tracking Study. *International Journal of Human-Computer Interaction*, *31*(2), 101–111. doi: 10.1080/10447318.2014.959105

Tsai, M. J. (2009). Online Information Searching Strategy Inventory (OISSI): A quick version and a complete version. *Computers and Education*, *53*(2), 473–483. doi: 10.1016/j.compedu.2009.03.006

Usta, A., Altingovde, I. S., Vidinli, I. B., Ozcan, R., & Ulusoy, Ö. (2014). How K-12 students search for learning? Analysis of an educational search engine log. In *Sigir 2014 - proceedings of the 37th international acm sigir conference on research and development in information retrieval* (pp. 1151–1154). doi: 10.1145/2600428.2609532

Vidinli, I. B., & Ozcan, R. (2016). New query suggestion framework and algorithms: A case study for an educational search engine. *Information Processing and Management*, *52*(5), 733–752. doi: 10.1016/j.ipm.2016.02.001

Vivekavardhan, J., Chakravarthy, A., & Ramesh, P. (2020). Search engines and meta search engines great search for knowledge: A frame work on keyword search for information retrieval. In *Advances in decision sciences, image processing, security and computer vision* (pp. 435–443).

Springer.

Witten, I. H., Paynter, G. W., Frank, E., Gutwin, C., & Nevill-Manning, C. G. (1999). *KEA: Practical Automatic Keyphrase Extraction* (Tech. Rep.). Retrieved from http://arxiv.org/abs/cs/9902007

Wu, D., & Cai, W. (2016). An empirical study on Chinese adolescents' web search behavior. In *Journal of documentation* (Vol. 72, pp. 435–453). Bradford: Emerald Group Publishing Limited. doi: 10.1108/JD-04-2015-0047

Zammit, O., Smith, S., De Raffaele, C., & Petridis, M. (2019). Exposing Knowledge: Providing a Real-Time View of the Domain Under Study for Students. In M. Bramer & M. Petridis (Eds.), *Lect. notes comput. sci. (including subser. lect. notes artif. intell. lect. notes bioinformatics)* (Vol. 11927 LNAI, pp. 122–135). Cham: Springer International Publishing. doi: 10.1007/978-3-030-34885-4_9

Zammit, O., Smith, S., Windridge, D., & De Raffaele, C. (2020). Exposing Students to New Terminologies While Collecting Browsing Search Data (Best Technical Paper). In M. Bramer & R. Ellis (Eds.), *Artificial intelligence xxxvii* (pp. 3–17). Cham: Springer International Publishing.

Zhou, M. (2015). SCOOP: A measurement and database of student online search behavior and performance. *British Journal of Educational Technology*, *46*(5), 928–931. doi: 10.1111/bjet.12290

Zukerman, I., & Raskutti, B. (2002). Lexical Query Paraphrasing for Document Retrieval. In *{COLING} 2002: The 19th international conference on computational linguistics*. Retrieved from https://www.aclweb.org/anthology/C02-1161