Exposing Knowledge: Providing a Real-Time View of the Domain Under Study for Students

Omar Zammit[1], Clifford De Raffaele[1], Serengul Smith[2], and Miltos Petridis[2]

[1] Faculty of Computer Science, Middlesex University Malta
Block A, Alamein Road Pembroke PBK 1776 Malta
o.zammit@mdx.ac.uk
cderaffaele@ieee.org

[2] Faculty of Computer Science, Middlesex University
The Burroughs, Hendon, London, NW4 4BT
s.smith@mdx.ac.uk
m.petridis@mdx.ac.uk

Abstract. With the amount of information that exists online, it is impossible for a student to find relevant information or stay focused on the domain under study. Research showed that search engines have deficiencies that might prevent students from finding relevant information. In this research, we are proposing a solution that takes the personal search history of a student into consideration and provides a holistic view of the domain under study. A bag-of-words for each searched keyword is created and used to query third-party APIs to find relevant papers and construct a Word Cloud. To archive this we built a user interface to present the aggregated results to the student. In order to evaluate our approach, we used some of the commonly used datasets and compared our results to existing research.

Keywords: Search Engine Keywords · Similarity Analysis · Text Enrichment

1 Introduction

Confidence in search engines has increased, some Internet users nowadays tend to give high veracity to a website because of its inclusion or high ranking in a search engine result [5]. Some authors also stated that with the amount of information that exists on-line it is more difficult for users to find accurate information [25] and therefore it is impossible for an Internet user to find information without the use of a search engine [11]. Unfortunately, such search engines have deficiencies. Some authors state that search engines tend to be biased and favor certain web sites over others [22]. And that such discrepancies between search engines makes it difficult for Internet users to decide which search engines to trust. It is already challenging for Internet users to judge relevance of on-line content [26] and ignore fake news [47] let alone having such inconsistency and doubts about validity.
By design, search engines are targeting a generic audience and search results might not be suitable for a specific group of people [49] like students. We are

focusing on providing a better search experience to students while they are doing research, and try to overcome some of the deficiencies imposed by search engines. To achieve this we are proposing a solution that aims to help students focus more on the domain they are studying by exposing them to various resources related to the domain. The solution takes in consideration previously searched queries related to the current domain being studied using a combination of similarity analysis techniques. A bag-of-words is created that is used to query third party APIs to find relevant papers and construct a Word Cloud. The proposed solution includes a graphical user interface that will allow students to have a holistic view of the domain being studied.

2 Current Solutions

There are various solutions that are trying to assist students in their study. Some suggest to move away from a search engine and focus on an educational search engine that is focusing on a particular domain [49]. Such approach is problematic since students tend to rely on search engines before libraries to search a new term or when they are unfamiliar with a new topic [10], and therefore it might be challenging to convince them to move away from search engines. Various authors focus their studies on users URL visited or browsing clicks to understand browsing habits [14, 43, 48, 49]. We are taking a similar approach but we are focusing mainly on the keywords visited by the student, since these are the entry point of a web search session.
As described by Kim et al.[24], a web search session has three main components, it starts with a student issuing a query, the search engine process the query and surface a website. Then the student visits the website (see Fig.1)[24]. The figure also depicts that actions between entities are bidirectional, this means that students can start with a query and they will continue formulating different queries until the required result is obtained [48]. In addition, during a study on an education search engine, the authors also noted that some of the queries submitted by students are most likely to be repeated [48]. Queries are often appended to

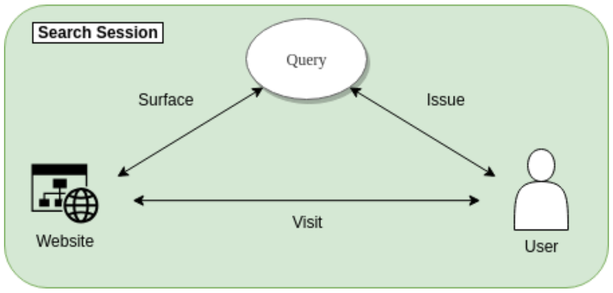


Fig. 1. A web search session as explained in [24]

URLs together with session parameters and form data [50] and some studies already showed that such data can be used to learn about users browsing behaviors [46]. Various studies has been done to understand how students search with

the purpose of learning, for example in their research Usta et al.[48] focused on *K-12*[3] students search behavior in a learning environment and compared such behavior with general web search engines trends [48]. Vidinli and Ozcan[49] proposed a general modular framework for query suggestion algorithm development to overcome the issue that search engines are targeting a very diverse population [49]. Their research focused mainly on *K-12* students, since such students have difficulty in formulating queries. In their research they reduced the query suggestion problem by comparing queries using various comparison algorithms. The only issue with such framework is that they are targeting *K-12* student only and not focusing on undergraduate or post-graduate students.
Smith et al.[43] did an exploratory study on query auto completion usage during a search session with assigned tasks. In order to monitor user activity they used a modified version of CrowdLogger[4], a Google Chrome extension to collect searched keywords. Searched keywords where re-submitted using Google QAC API and all `Search Engine Results Pages` (SERP) were scraped and cleaned from images and other elements before displaying them to the user. Such approach can be used to collect data and evaluate a system that its main aim is to learn more about browsing habits [43]. We moved away from creating a Internet browser extension, since this requires a student to install such extension. We opted to take a seamless approach, that reads directly from the browser local history database and thus is invisible to the student.
Various work was cited by Smith et al.[43] on feature extraction when learning on browsing behavior. In Bast and Weber[6] the authors mined query logs and used query frequency to predict query completion and rank suggestions [6]. Some authors also took context into Python method to convert text to tokens like seasonal adjustment [42], demographics and individual search history [41].

3 Proposed Solution

As stated by Rathod[39], keywords search terms play an important role to understand user's psychology [39] and we are using keywords searched by the student to understand which domains relate to the student. Past students keywords are taken in consideration since some studies show that previous information requests originated by a user determine the context of the research [7, 16]. To learn more about the student search preferences, and provide a holistic view of the domain being researched by the student, we are extending the search session as described in Kim et al. [24] by adding an additional framework (see Fig. 2).
Internet browsers, like Google Chrome and Mozilla Firefox, are storing history data on the local computer in SQLite[5] databases [39] for reference. Such databases contain data about URLs visited by the student and keywords searched online. Unless special applications (for example, SQLite Database Browser[6] or

[3] From kindergarten to secondary school.
[4] https://crowdlogger.cs.umass.edu/
[5] https://www.sqlite.org/index.html
[6] urlhttps://sqlitebrowser.org/

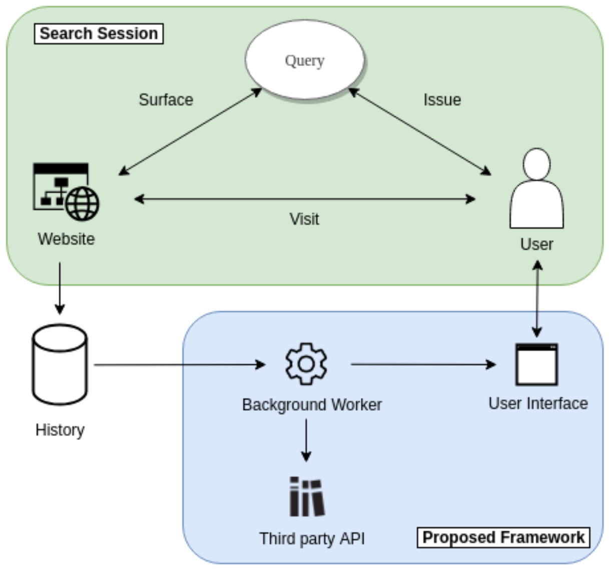


Fig. 2. Proposed framework showing additional components

SQLite Studio[7]) are used, such data is not easily accessible for students. We are focusing on Google Chrome in this research since it is the leading Internet browser [33].

3.1 Background Worker

Keywords Extraction from Local History We implemented a background worker to extract information from the local history files and predict suitable content. To achieve this the SQLite history file is copied into an accessible location (since this cannot be read while being used by the browser) and SQL statements are executed to extract keywords. Listing 1.1 shows the SQL query used to extract keywords from the local history file. Note that `last_vist_time` and `limit` are parameters specified by the background worker to retrieve data in batches. The background worker will execute similar queries to extract information on what the student searched in the past and identify in real time what the student is currently searching during the search session.

Listing 1.1. SQLite query to get all keywords searched by the user from Google Chrome history.

```
select *
from urls
inner join keyword_search_terms on
           urls.id = keyword_search_terms.url_id
where urls.last_visit_time > ?
```

[7] https://sqlitestudio.pl/index.rvt

```
order by urls.last_visit_time asc limit ?;
```

Text Enrichment using Third Party API Keywords are not enough to determine what students are searching for. Research shows that search queries tend to be short ambiguous and under specified [37]. Having short text is less effective due to its brevity and less sparsity of words and when dealing with such data, enriching the semantics using external entities is essential [40]. Various studies used third party like Wikipedia to enrich text or find similarity between keywords [3, 15, 38, 40, 43].
In this research, text enrichment was implemented in the background worker, so that more insight is known about the domain related to the keywords submitted by the student. To achieve this, for each keyword searched by the student that is stored in the local history, a Google search URL is created (see Listing 1.2) and sent to Google.

Listing 1.2. Google request URL format

```
url = f"https://www.google.com.mt/search?" \
      f"q={keyword}&oq={keyword}&client=ubuntu"
```

The HTML response obtained contains search results in the form of HTML anchor tags consisting of a URL to an external source and a short description (anchor text). The background worker will parse the HTML tags and identify anchor tags. Since results may contain trending and e-commerce anchor tags other than sponsored links [38], Python libraries based on Levenshtein distance[8] were used to determine if the anchor tag is relevant to the keyword being searched. This was done by comparing the anchor tag description to the keyword itself.
As explained by Haldar and Mukhopadhyay[19], Levenshtein distance is the number or deletions, insertions or substitutions required to transform a source string s into a target string t [19]. The algorithm steps to compute distance lev:

Step 1: Let n be the length of s and m be the length of t.
Step 2: If $min(n, m) = 0$ then $lev = max(n, m)$. No more steps.
Step 3: Create a matrix d containing $0..m$ rows and $0..n$ columns.
Step 4: Set the first row to $0..n$ and first column to $0..m$.
Step 5: Process each $s[i]$ value from 1 to n
Step 6: Process each $t[i]$ value from 1 to m
Step 7: If $s[i] = t[i]$ then $cost = 0$
Step 8: If $s[i] \neq t[i]$ then $cost = 1$
Step 9: Set $d[i, j]$ as follows:

$$d[i,j] = min \begin{cases} d[i-1,j]+1 \\ d[i,j-1]+1 \\ d[i-1,j-1]+cost \end{cases} \quad (1)$$

Step 10: Repeat from step 5 until $d[n, m]$ value is found.

[8] https://github.com/seatgeek/fuzzywuzzy

Step 11: $lev = d[n, m]$

The smaller the Levenshtein distance between the keyword and the anchor text description, the more similar the two strings are [19]. Once anchor texts having high degree of similarity are identified, a web request is done for each anchor tag link and a bag-of-words based on their HTML content is created. Each HTML response obtained from anchor tags link was cleaned as described by Hu et al.[21], that is, removing HTML tags from the response, identify tokens, removing stop words and eliminating punctuation [21]. The normalization steps done in this research are similar to the steps suggested by Gowtham et al.[17]. But we took a different approach, we used the Python Natural Language Toolkit, since this includes functions to convert text to tokens, has a list of stop words, can perform part of speech tagging and can convert a word to its lemma [28]. In addition, the toolkit contains an implementation of the WordNet lexical database [23] used to check the validity of the words. WordNet was selected since it models the lexical knowledge of an English native speaker and defines Nouns and Verbs in a well-defined hierarchy [35][32][23]. Some studies show that the majority of queries submitted by users over the internet are a structured collection of noun-phrases, in fact 70% of the query terms are made up mainly of nouns and proper nouns [4] while other words like helping verbs and pronouns are considered as stopwords [38]. As stated by Barr et al.[4] part-of-speech tagging on query keywords can be significant when extracting features in machine learning [4]. We considered this fact and in addition to text normalization, tokens that are not nouns and verbs were removed from the bag-of-words.

3.2 Local Database

Once text enrichment is done, a local database is created that stores all keywords searched by the user and their respective bag-of-words. We took this approach so that text enrichment is only done once for a given keyword. As explained in Section 3.4, similarity analysis is done using the bag-of-words stored in the local database. Figure 3 shows the table that contains the keyword and the bag-of-words.

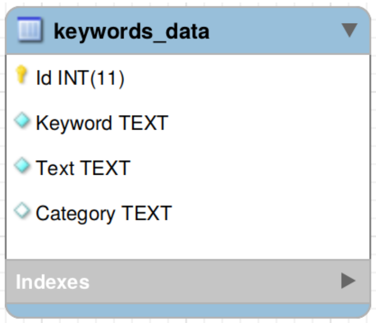
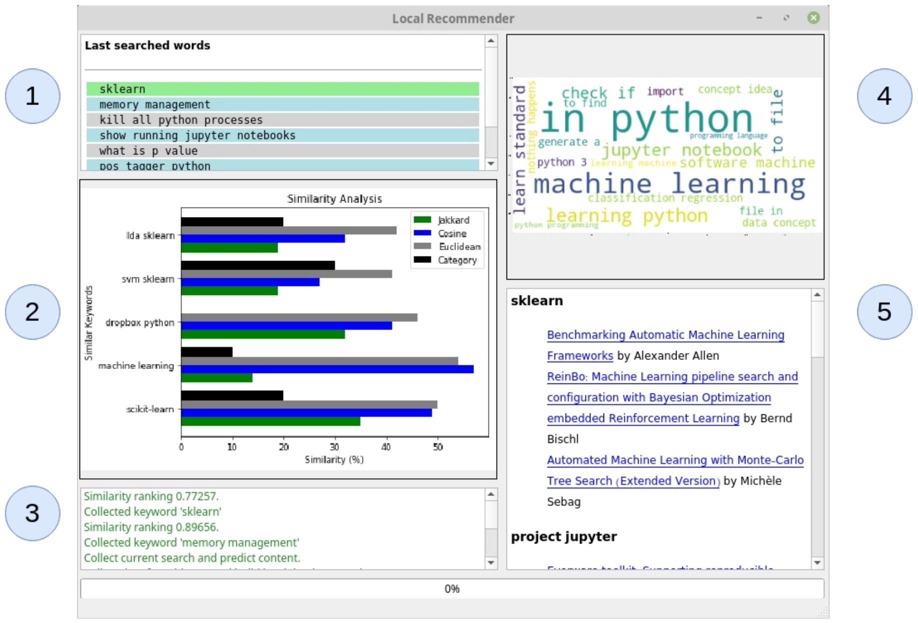


Fig. 3. Local database table details

3.3 User Interface

When dealing with large amount of data, the focus point should not just be the collection of data but the analysis and the ability to find meaningful results from it [1]. In order to assist students a user interface was created that will allow the students to view the results and the predictions computed by the background worker (see Figure 4). The user interface is divided as follows:



Last searched words

```
sklearn
memory management
kill all python processes
show running jupyter notebooks
what is p value
pos tagger python
```

```
Similarity ranking 0.77257.
Collected keyword 'sklearn'
Similarity ranking 0.89656.
Collected keyword 'memory management'
Collect current search and predict content.
```

sklearn

Benchmarking Automatic Machine Learning Frameworks by Alexander Allen

ReinBo: Machine Learning pipeline search and configuration with Bayesian Optimization embedded Reinforcement Learning by Bernd Bischl

Automated Machine Learning with Monte-Carlo Tree Search (Extended Version) by Michèle Sebag

project jupyter

Fig. 4. Evaluation User Interface to Display Predicted Data

1. *Current Search*: Provides a list of the last keywords searched by the student.
2. *Similar Searches*: Displays similar keywords searched by the student in the past. Explained in Section 3.4.
3. *System Logs*: Contains system logs.
4. *Domain Word Cloud*: A word cloud representing the most commonly used `ngrams` in the domain currently being searched by the student. Explained in Section 3.5.
5. *Academic References*: Papers relevant to the domain of study collected from Arxiv database. Explained in Section 3.5.

3.4 Similarity Analysis

Feature Extraction for Similarity Analysis In order to find similar keyword searches similarity analysis was used. Hansen and Jaumard(1997) stated that in order to group data, a dataset $O = \{O_1, O_2, ..., O_n\}$ of N entities is needed [20]. The dataset is made up of the keywords searched by the user and their respective bag-of-words. Hansen and Jaumard(1997) also stated that to classify samples O, one should identify p characteristics of each sample and end up with a matrix X of $N \times p$. Since these characteristics define and will determine the dissimilarities between entities. Perez-Tellez et al.(2014) identified various features that helped them in characterization and categorization of Weblogs and other short texts. Most of the features rely on the words (tokens) within the text [36]. For effective transformation and for representation, word frequencies must be normalized in terms of their frequency within a document and within the entire collection [35]. To achieve this Bafna et al.(2016) used TF-IDF with K-means and hierarchical algorithms to classify news, emails and research papers on different topics [2]. TF-IDF was used since this is a technique used to reduce the importance of common terms in a collection so that it ensures that the matching of documents is more influenced by discriminative words having low frequency [35]. Such technique has already been used in various studies, for example some authors used TF-IDF with K-means and hierarchical algorithms to classify news, emails and research papers on different topics [2]. The idea behind this technique is to normalize the words taking in consideration their frequency within a document and within the entire collection [35]. As described by Erra et al.[13] TF-IDF measure for a term t will be [13]:

- A higher value when t appears many times within few documents.
- A low value if t appears many times in many documents or fewer times in one document.
- A low value if t appears in all documents.

Let $D = \{d_1, d_2, ..., d_n\}$ be a collection of documents or block of texts. TF-IDF for word t can be computed as followin order to find similar keyword searched by the student for a given keyword, similarity analysis was used. Hansen and Jaumard[20] stated that in order to group data a dataset $O = \{O_1, O_2, ..., O_n\}$ of N entities is needed [20]. The dataset is made up of the keywords searched by the student and their respective bag-of-words. Hansen and Jaumard[20] also stated that to classify samples O one should identify p characteristics of each sample and end up with a matrix X of $N \times p$. Since these characteristics define and will determine the dissimilarities between entities. Perez-Tellez et al.[36] identified various features that helped them in characterization and categorization of Weblogs and other short texts. Most of the features rely on the words (tokens) within the text [36]. For effective transformation and for representation, word frequencies must be normalized in terms of their frequency within a document and within the entire collection [35]. To achieve this Bafna et al.[2] used TF-IDF with K-means and hierarchical algorithms to classify news, emails and research papers on different topics [2]. TF-IDF was used since this is a technique s [13]:

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D) \tag{2}$$

Where $tf(t, d)$ is the number of instances of t in a document d. And $idf(t, D)$ which is the inverse document frequency can be described as [13]:

$$idf(t, D) = \log_{10} \left(\frac{|D|}{|\{d|t \in d\}|} \right) \tag{3}$$

Where the total amount of documents D is divided by the number of documents containing term t. Scikit-learn[9] has a TF-IDF implementation that can convert raw text into a TF-IDF matrix [10].

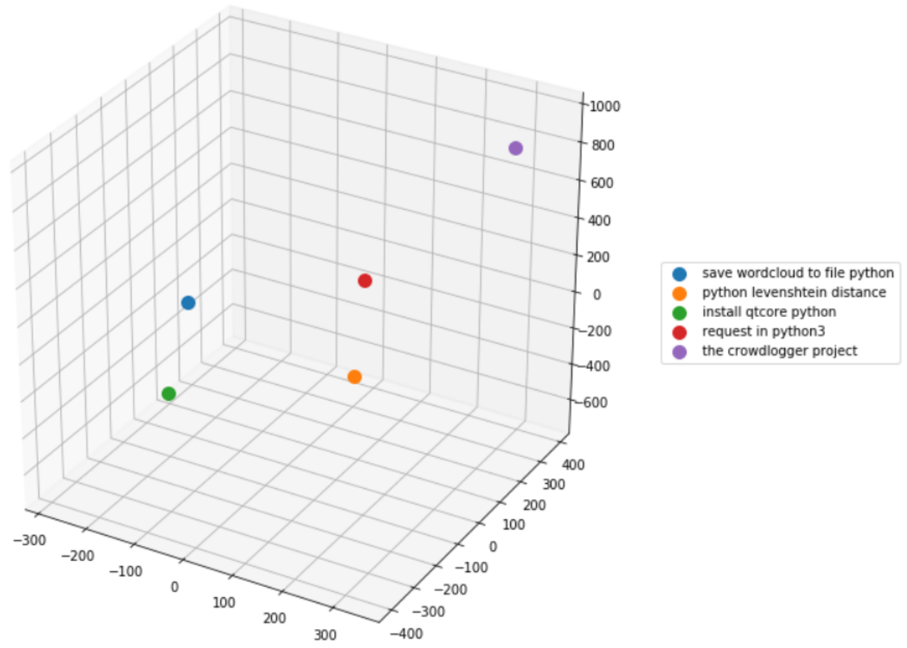


Fig. 5. Plotting TF-IDF values for some keywords.

In this research each bag-of-word for a given keyword was treated as a document and converted into a TF-IDF matrix. Figure 5 shows a scatter plot of a TF-IDF matrix for three keyword searches. Since TF-IDF has high dimensional data t-distributed Stochastic Neighbor Embedding (t-SNE) was used as a dimension reduction technique. t-SNE creates a location in a two or three dimensional map for each datapoint and therefore reduces the dimensions using a probabilistic technique [29]. van der Maaten and Hinton[29] outlined that t-SNE showed high performance and a much better job revealing the classes when compared to other techniques [29]. An implementation of t-SNE is also available in Scikit-learn[11]. In order to find similar keywords, we used three similarity measures. Cosine similarity, Euclidean distance and Jaccard similarity. The first two measures take as an input the TF-IDF vector to compute the similarity, while the latter takes the actual bag-of-words. Every time a student searches a new keyword, the background worker will detect the keyword and creates a bag-of-words and a TF-IDF vector. Cosine similarity, Euclidean distance and Jaccard similarity are

[9] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
[10] https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html
[11] https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

computed comparing the searched keyword bag-of-words with existing keywords. Similar keywords are selected as follows:

- Let $C = \{x : x$ keyword having high cosine similarity$\}$
- Let $J = \{x : x$ keyword having high jaccard similarity$\}$
- Let $E = \{x : x$ keyword having high eucledean distance$\}$
- Display only entries from $C \cap J$, $C \cap E$ and $E \cap J$.

Cosine Similarity Chaithanya and Reddy[9] explained how Cosine similarity can be used with TF-IDF to measure the similarity between two vectors and outlined that such measure is suitable since it focuses on the orientation of the document rather than the magnitude [9].

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} {A_i}^2}\sqrt{\sum_{i=1}^{n} {B_i}^2}} \tag{4}$$

Cosine similarity ranges from -1 (exactly opposite), to 1 (exactly the same) [9]. An implementation of this measure is available in Scikit-learn[12].

Euclidean Distance If $X_i = (x_{i,1}, ..., x_{i,D})$ and $X_j = (x_{j,1}, ..., x_{j,D})$ are D-dimensional vectors representing two bag-of-words for two keywords that need to be compared. The Euclidean distance η between both vectors is computed as [31]

$$\eta = \sqrt{\sum_{d=1}^{D} (x_{i,d} - x_{j,d})^2} \tag{5}$$

An implementation of the Euclidean distance is available in Scikit-learn[13]. Since distance range can vary, normalization of the result was done using $\frac{1}{1+\eta}$.

Jaccard Similarity Niwattanakul et al.[34] explained that Jaccard similarity can determine the similarity between two data sets and is computed by dividing the number of features that are common between two datasets by the number of features that are not common [34]. Let A and B be two bag-of-words for two keywords that need to be compared. Jaccard similarity can be computed as.

$$jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{6}$$

Weighted Score Based Aggregation A weighted score based aggregation was used as suggested by Vidinli and Ozcan [49] to aggregate the three algorithms

[12] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html
[13] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.euclidean_distances.html

used and compute the final similarity score. Each similarity score V was assigned a coefficient k and the final score was computed as.

$$Score = k_{cos} \times V_{cos} + k_{jak} \times V_{jak} + k_{euc} \times V_{euc} \qquad (7)$$

A score is assigned for each keyword searched by the student from the local database. The top 10 keywords having the highest scores are displayed in the user interface and presented to the student as top similar keywords.

3.5 Wordcloud and Academic References

In order to provide an overview of the domain being researched and to expose student to new terminologies within the domain, we added a word cloud to the user interface. The word cloud is constructed using bigrams extracted from the bag-of-words associated with the similar keywords identified by the background worker. In addition, a third party provider for academical journals was used to display research papers and expose student to research material related to the domain under research. The approach involves, identifying the top most used bigrams from the bag-of words, find similar titles in Wikipedia (to endure that the bigrams are valid), search bigrams in the Arxiv database [30] and display top results returned in the user interface. The following list shows how some bigrams from the bag-of-words were transformed into more meaningful titles using Wikipedia:

- document preparation $\rightarrow$ document preparation system
- latex project $\rightarrow$ latex
- preparation system $\rightarrow$ document preparation system
- language processing $\rightarrow$ natural language processing

4 Evaluation

In order to evaluate the proposed solution and assess the semantic relationship validity we used **Mturk-771**[14][18], **Rel-122**[15][45] and **WordSimilarity-353**[16][16], since these are some of the most commonly used datasets for similarity analysis [27]. Datasets are composed of two terms and their similarity score based on human judgment. A grid search approach similar to the one described by Buitinck et al. [8] was conducted on each dataset to determine the best similarity coefficient and to validate the results obtained. A nested loop was created that allowed to iterate through 11 coefficient values (from 0 to 2 with 0.2 increment) for each similarity analysis totaling a 11^3 combinations. For each combination a similarity score was computed by our system for each word pair in the dataset. Pearson product-moment correlation coefficient [12] and Spearman

[14] http://www2.mta.ac.il/ gideon/mturk771.html
[15] http://www.cs.ucf.edu/ seansz/rel-122/
[16] http://www.cs.technion.ac.il/ gabr/resources/data/wordsim353/

rank-order correlation coefficient [44] was used to compare the similarity score obtained by our system to the human judgment score in the dataset. Both correlations are implemented in the `scipy.stats` library[17]. For every grid search iteration, the highest Pearson and Spearman rank was noted together with the iteration coefficients. Table 1 shows the highest Pearson and Spearman rank obtained for each dataset and the similarity coefficient that was used by the grid search. We used bold to indicate the highest value obtained. Results in table 1

WordSimilarity-353 grid search results				
Cosine Coefficient	Jakkard Coefficient	Euclidean Coefficient	Pearson	Spearman
1.4	0	0	**0.456809**	0.566421
1.8	0	1.2	0.437034	**0.568721**

Mturk-771 grid search results				
Cosine Coefficient	Jakkard Coefficient	Euclidean Coefficient	Pearson	Spearman
0.8	0	0	**0.441912**	**0.528315**

Rel-122 grid search results				
Cosine Coefficient	Jakkard Coefficient	Euclidean Coefficient	Pearson	Spearman
2	0	0	**0.438949**	0.469933
0.6	0	0.2	0.338858	**0.479553**

Table 1. Grid search results for all datasets

show that Jakkard Similarity did not contribute in improving the accuracy of the similarity analysis, while Euclidean distance contributed in improving the Spearman Correlation for some datasets.
In order to evaluate our approach, we compared our similarity results to existing research mainly focusing on the work done by Li et al.[27]. In their research the authors used Wikipedia features to find the similarity between terms and they compared their results with existing benchmarks.

	Pearson		Spearman	
Dataset	Benchmark	Our System	Benchmark	Our System
Mturk-771	**0.56**	0.44	**0.62**	0.53
Rel-122	**0.64**	0.44	**0.65**	0.47
WordSimilarity-353	**0.56**	0.46	**0.76**	0.57

Table 2. Comparison of Pearson and Spearman as Li et al.[27]

As shown in Table 2, our results are lower than the benchmarks identified in Li et al.[27] but this does not mean that the proposed solution is not robust enough to assist students. One should note that the evaluation approach is measuring the ability of our system to compute the similarity between two keywords or for a given keyword find only one similar keyword from a list of previously

[17] https://docs.scipy.org/doc/

searched keywords. Therefore the evaluation is not taking in consideration that our similarity analysis is configured to return top 10 similar keywords.
For example, assume that for a given keyword, according to human judgment *'Keyword A'* and *'Keyword B'* are the top two most similar keywords from a list of keywords. Since our proposed solution is configured to return the top 10 similar keywords, it will go through the list, perform similarity analysis and assign a ranking. Keywords are sorted in descending order by ranking and the top 10 items are identified and presented to the student. The evaluation is assessing the capability of our system to predict *'Keyword A'* and *'Keyword B'* as the top two items and not that *'Keyword A'* and *'Keyword B'* are actually in the list of top 10 keywords and therefore they are still visible to the student. Having both keywords in the list means that the student still holds visibility of *'Keyword A'* and *'Keyword B'*.

5 Conclusion

It is difficult for students to keep focus while research and find relevant information, various factors including query formulation and search engines deficiencies. We took advantage of the browsing history database to extract keywords and try to understand what students are searching for. We proposed a solution that captures the keywords searched by the student in real-time and provides a holistic view of the domain under study. For a given keyword, using various similarity analysis, we are identifying similar previously searched keywords, creating a domain word cloud and retrieve academical papers related to the domain under study. Results and predictions are aggregated and presented to the student in a user interface that can be used alongside an Internet browser.
The evaluation performed showed that Cosine similarity and Euclidean distance contributed in increasing the accuracy of the proposed solution while Jakkard similarity did not contribute. Although the proposed solution scored less accuracy than existing benchmarks, the fact that the solution is identifying the top 10 similar keywords and displaying words related to the domain in a word cloud makes the solution robust and suitable for students during their study.

Bibliography

[1] Aljrees, T., Shi, D., Windridge, D., Wong, W.: CRIMINAL PATTERN IDENTIFICATION BASED ON MODIFIED K-MEANS CLUSTERING. In: 2016 International Conference on Machine Learning and Cybernetics (ICMLC), vol. 2, pp. 10–13, South Korea (2016), ISBN 2160-1348 VO - 2, https://doi.org/10.1109/ICMLC.2016.7872990

[2] Bafna, P., Pramod, D., Vaidya, A.: Document clustering: TF-IDF approach. pp. 61–66 (2016), https://doi.org/10.1109/ICEEOT.2016.7754750

[3] Banerjee, S., Ramanathan, K., Gupta, A.: Clustering short texts using wikipedia. In: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '07 (2007), ISBN 9781595935977, ISSN 1595935975, https://doi.org/10.1145/1277741.1277909

[4] Barr, C., Jones, R., Regelson, M.: The Linguistic Structure of English Websearch Queries. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1021–1030, EMNLP '08, Association for Computational Linguistics, Stroudsburg, PA, USA (2008), URL http://dl.acm.org/citation.cfm?id=1613715.1613848

[5] Bartlett, J., Miller, C.: Truth, Lies and the Internet a Report Into Young People'S Digital Fluency. Demos (September), 1–59 (2011), ISSN 1603-9629

[6] Bast, H., Weber, I.: Type less, find more: fast autocompletion search with a succinct index. In: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 364–371, ACM (2006)

[7] Bharat, K.: SearchPad: Explicit capture of search context to support web search. Computer Networks **33**(1-6), 493–501 (2000)

[8] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., Varoquaux, G.: API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD Workshop: Languages for Data Mining and Machine Learning, pp. 108–122 (2013)

[9] Chaithanya, K., Reddy, P.V.: A Novel approach for Document Clustering using Concept Extraction. Tech. rep. (2016), URL www.ijirae.com

[10] Cheng, Y.h., Tsai, C.c.: Online Research Behaviors of Engineering Graduate Students in Taiwan **20**, 169–179 (2017)

[11] Chiru, C.: SEARCH ENGINES: ETHICAL IMPLICATIONS. Tech. Rep. 1 (2016)

[12] Dillon, M.: Introduction to modern information retrieval: G. Salton and M. McGill. McGraw-Hill, New York (1983). xv+ 448 pp., $32.95 ISBN 0-07-054484-0 (1983)

[13] Erra, U., Senatore, S., Minnella, F., Caggianese, G.: Approximate TF-IDF based on topic extraction from massive message stream using

the GPU. Information Sciences **292**, 143–161 (2015), ISSN 00200255, https://doi.org/10.1016/j.ins.2014.08.062

[14] Feild, H.A., Allan, J., Glatt, J.: CrowdLogging: distributed, private, and anonymous search logging. In: Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, pp. 375–384, ACM (2011)

[15] Ferragina, P., Scaiella, U.: TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In: Proceedings of the 19th ACM international conference on Information and knowledge management - CIKM '10 (2010), ISBN 9781450300995, ISSN 0740-7459, https://doi.org/10.1145/1871437.1871689

[16] Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., Ruppin, E.: Placing search in context: The concept revisited. ACM Transactions on information systems **20**(1), 116–131 (2002), https://doi.org/10.1145/371920.372094, URL http://doi.acm.org/10.1145/371920.372094

[17] Gowtham, S., Goswami, M., Balachandran, K., Purkayastha, B.S.: An approach for document pre-processing and K Means algorithm implementation. In: Proceedings - 2014 4th International Conference on Advances in Computing and Communications, ICACC 2014, pp. 162–166, IEEE (2014), ISBN 9781479943647, https://doi.org/10.1109/ICACC.2014.46

[18] Halawi, G., Dror, G., Gabrilovich, E., Koren, Y.: Large-scale learning of word relatedness with constraints. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1406–1414, ACM (2012)

[19] Haldar, R., Mukhopadhyay, D.: Levenshtein distance technique in dictionary lookup methods: An improved approach. arXiv preprint arXiv:1101.1232 (2011)

[20] Hansen, P., Jaumard, B.: Cluster Analysis and Mathematical Programming. Math. Program. **79**(1-3), 191–215 (oct 1997), ISSN 0025-5610, https://doi.org/10.1007/BF02614317, URL http://dx.doi.org/10.1007/BF02614317

[21] Hu, X., Tang, J., Gao, H., Liu, H.: Unsupervised sentiment analysis with emotional signals. In: Proceedings of the 22nd international conference on World Wide Web - WWW '13 (2013), ISBN 9781450320351, ISSN 15334880, https://doi.org/10.1145/2488388.2488442

[22] Introna, L.D., Nissenbaum, H.: Shaping the web: Why the politics of search engines matters. Information Society **16**(3), 169–185 (2000), ISSN 10876537, https://doi.org/10.1080/01972240050133634

[23] Kilgarriff, A., Fellbaum, C.: WordNet: An Electronic Lexical Database. Language **76**(3), 706 (2000), ISSN 00978507, https://doi.org/10.2307/417141, URL https://www.jstor.org/stable/417141?origin=crossref

[24] Kim, J.Y., Collins-Thompson, K., Bennett, P.N., Dumais, S.T.: Characterizing web content, user interests, and search behavior by reading level and topic. In: Proceedings of the fifth ACM international conference on Web

search and data mining - WSDM '12 (2012), ISBN 9781450307475, ISSN 9781450307475, https://doi.org/10.1145/2124295.2124323

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

[26] Leeder, C.: Student misidentification of online genres. Library and Information Science Research **38**(2), 125–132 (2016), ISSN 07408188, https://doi.org/10.1016/j.lisr.2016.04.003

[27] Li, P., Xiao, B., Ma, W., Jiang, Y., Zhang, Z.: A graph-based semantic relatedness assessment method combining wikipedia features. Engineering Applications of Artificial Intelligence **65**, 268–281 (2017)

[28] Loper, E., Bird, S.: NLTK: The Natural Language Toolkit (2002), ISSN 00313998, https://doi.org/10.3115/1118108.1118117, URL http://arxiv.org/abs/cs/0205028

[29] van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. Journal of machine learning research **9**(Nov), 2579–2605 (2008)

[30] McKiernan, G.: arXiv. org: the Los Alamos National Laboratory e-print server. International Journal on Grey Literature **1**(3), 127–138 (2000)

[31] Mesquita, D.P.P., Gomes, J.P.P., Junior, A.H.S., Nobre, J.S.: Euclidean distance estimation in incomplete datasets. Neurocomputing **248**, 11–18 (2017)

[32] Miller, G.A.: WordNet: a lexical database for English. Communications of the ACM **38**(11), 39–41 (1995), ISSN 00010782, https://doi.org/10.1145/219717.219748, URL http://portal.acm.org/citation.cfm?doid=219717.219748

[33] NetMarketShare: Mobile/Tablet Browser Market Share (2013), URL https://www.netmarketshare.com/browser-market-share.aspx

[34] Niwattanakul, S., Singthongchai, J., Naenudorn, E., Wanapu, S.: Using of Jaccard coefficient for keywords similarity. In: Proceedings of the international multiconference of engineers and computer scientists, vol. 1, pp. 380–384 (2013)

[35] Patil, L.H., Atique, M.: A novel approach for feature selection method TF-IDF in document clustering. In: Proceedings of the 2013 3rd IEEE International Advance Computing Conference, IACC 2013, pp. 858–862, IEEE (2013), ISBN 9781467345286, ISSN 1703-8847, 1703-8847, https://doi.org/10.1109/IAdCC.2013.6514339

[36] Perez-Tellez, F., Cardiff, J., Rosso, P., Pinto, D.: Weblog and short text feature extraction and impact on categorisation. Journal of Intelligent and Fuzzy Systems **27**(5), 2529–2544 (2014), ISSN 18758967, https://doi.org/10.3233/IFS-141227

[37] Pound, J., Hudek, A.K., Ilyas, I.F., Weddell, G.: Interpreting keyword queries over web knowledge bases. In: Proceedings of the 21st ACM international conference on Information and knowledge management - CIKM '12, p. 305 (2012), ISBN 9781450311564, https://doi.org/10.1145/2396761.2396803, URL http://dl.acm.org/citation.cfm?doid=2396761.2396803

[38] Rajeshwarkar, A., Nagori, M.: Optimizing Search Results using Wikipedia based ESS and Enhanced TF-IDF Approach. International Journal of Computer Applications **144**(12) (2016)

[39] Rathod, D.M.: Web browser forensics: google chrome. International Journal of Advanced Research in Computer Science **8**(7) (2017)

[40] Shirakawa, M., Nakayama, K., Hara, T., Nishio, S.: Wikipedia-Based Semantic Similarity Measurements for Noisy Short Texts Using Extended Naive Bayes. IEEE Transactions on Emerging Topics in Computing **3**(2), 205–219 (2015), ISSN 21686750, https://doi.org/10.1109/TETC.2015.2418716

[41] Shokouhi, M.: Learning to personalize query auto-completion. In: Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval, pp. 103–112, ACM (2013)

[42] Shokouhi, M., Radinsky, K.: Time-sensitive query auto-completion. In: Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval, pp. 601–610, ACM (2012)

[43] Smith, C.L., Gwizdka, J., Feild, H.: The use of query auto-completion over the course of search sessions with multifaceted information needs. Information Processing & Management **53**(5), 1139–1155 (2017)

[44] Spearman, C.: The proof and measurement of association between two things. American journal of Psychology **15**(1), 72–101 (1904)

[45] Szumlanski, S., Gomez, F., Sims, V.K.: A new set of norms for semantic relatedness measures. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 890–895 (2013)

[46] Tikhonov, A., Prokhorenkova, L.O., Chelnokov, A., Bogatyy, I., Gusev, G.: What can be Found on the Web and How. Proceedings of the ACM Web Science Conference on ZZZ - WebSci '15 pp. 1–10 (2015), https://doi.org/10.1145/2786451.2786468, URL http://dl.acm.org/citation.cfm?doid=2786451.2786468

[47] Tredinnick, L., Laybats, C.: Evaluating digital sources: Trust, truth and lies. Business Information Review **34**(4), 172–175 (2017), ISSN 17416450, https://doi.org/10.1177/0266382117743370

[48] Usta, A., Altingovde, I.S., Vidinli, I.B., Ozcan, R., Ulusoy, Ö.: How k-12 students search for learning?: analysis of an educational search engine log. In: Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval, pp. 1151–1154, ACM (2014)

[49] Vidinli, I.B., Ozcan, R.: New query suggestion framework and algorithms: A case study for an educational search engine. Information Processing and Management **52**(5), 733–752 (2016), ISSN 03064573, https://doi.org/10.1016/j.ipm.2016.02.001, URL http://dx.doi.org/10.1016/j.ipm.2016.02.001

[50] West, A.G., Aviv, A.J.: Measuring Privacy Disclosures in URL Query Strings. Internet Computing, IEEE **18**(6), 52–59 (2014), ISSN 1089-7801, https://doi.org/10.1109/MIC.2014.104