Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 44 No. 06 (2023)

Semantic Search for IoT Based Object Recognition Using Clustering and Classification Techniques

Raghu Nandan R.[1], Nalini. N.[2]

[1]*Research Scholar, Visvesvaraya Technological University, Belagavi, Navkis College of Engineering, Hassan, Karnataka, India.*

[2]*Nitte Meenakshi Institute of Technology, Bengaluru, India.*

Abstract: A search engine is a tool that helps to extract our required information with large collections of repositories using data retrieval techniques. As part of related tasks such as searching and data mining, enterprise search engines must be able to analyze a wide range of information resources within the enterprise and employ organizational expertise. Another important group of systems with design goals that differ significantly from commercial search engines are open source search engines. After the initial data collection, it needs to convert these files (code, docstrings) into pairs. These pairs are ideal to collect as training data for code acquisition models. The implementation is Domain-specific optimizations such as B Tree-based P2P can be used to get the most efficiency out of the information in code along with syntax-aware tokenization. After training this model for the frozen version, freeze all layers and continue training the model for a long period of time. This will help us improve the models usage for this work. To quickly find the nearest neighbors, these vectors are inserted into the search index in the final step. To achieve reasonable real-time object recognition, the aim is for at least 10 frames per second. Therefore, Intel NCS cannot reach the 10 fps target. However, it's worth checking out second iteration of the device. This is obviously more powerful and can run arbitrary object detectors in real time. The proposed framework uses semantic and machine learning techniques to extract and model data for specific kinds of IoT applications. Test results show that our system is scalable and can match various IoT applications with a large number of data domains with 80% accuracy.

Keywords: *Search Engine, Semantic Search, Domain-Specific Optimization, P2P model, Tokenization, Concrete pooling.*

1. Introduction

Search Engines are the real time applications used for data extractions from huge text repositories. A web search engine is a common example, but as stated earlier, search engines are part of the various applications such as customized search and enterprise search. “Search Engine” was coined and first used to indicate a particular hardware intended for the text search. Because of the convergence of Internet of Things and Machine Learning prompted large scope for the broad and informative data analysis [1]. IoT generates massive amounts of data every day from the various connected objects. Machine Learning helps in extracting the hidden correlations between the data. Machine Learning also helps in building the best prediction model by analyzing the existing data by applying various existing algorithms. Many existing search engines namely Google, Bing and Yahoo come in various variants which reflect the applications designed for web search upgraded to crawl many terabytes of data by reducing the response time to fewer milliseconds for all queries all over the world.

A typical and inevitable application to search large text repositories is the Search Engine. There are many categories of search engines involving the general purpose search engines and customized search engines. Long ago the term search engine was commonly used to indicate a specific hardware for searching text. The enterprise

search engines must be capable of processing a large variety of information sources inside the organization and leverage the knowledge as a product of information mining and searching. Data mining is a domain of extracting the related patterns which leads to the knowledge representation by means of applying the data mining techniques prominently clustering and classification. Open source search engines are one more significant type of systems with varied design goals compared to the commercial search engines. These systems are many and one can get links to these for data recovery. Modern search engines are not under estimated in their capability. Knowledge mining is a fast job for gathering information from the internet [2][3]. Unluckily, this tremendous capability cannot be used at every place. Searching is not readily available if the searing object is not the text data. Further, strict keyword searches will not allow users to extract information in a knowledgeable way. That is to say that the vital information is missing in the search results. Figure 1 below shows what is interconnected within an organization.

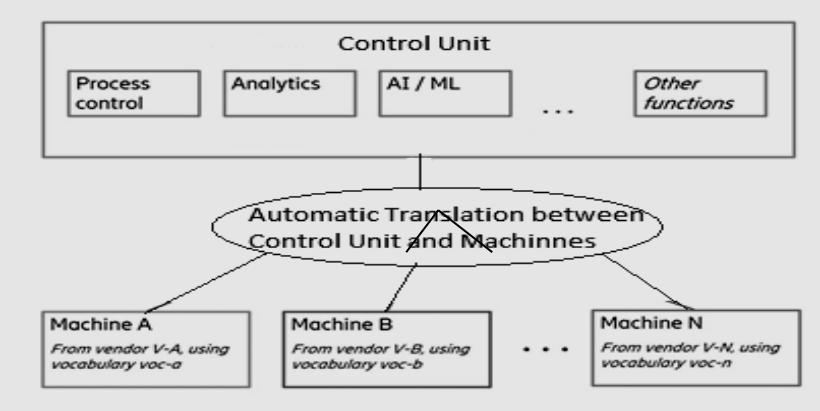

Fig1: Interconnected things in an organization

The above figure depicts the power of semantic search along the keywords, where the chances of getting the information by the user are increased for what they are looking. The advantages of the semantic search is manifold to name it allows developers to search code in the repositories in-spite of unaware of the syntax or predicting the correct syntax or words [4]. Necessarily this approach can be generalized for the IoT-based connected objects including the images, sounds and other things one cannot even thought off. The aim is to correlate code to natural language vector space, in which (code, text) ordered pairs describing the same context will be the close relatives, but the unrelated (code, text) ordered pairs will be estimated using cosine similarity. There exists many methods to arrive at the same point but we have demonstrated how to take a pre-trained model and tune the model to achieve the extracted frames form the code and also to extract the hidden features in the natural language vector space[5]. This process is carried out in five stages. These stages are illustrated in the following figure and will be the road map for the progress through the work. The figure below represents IoT-based semantic search architecture.

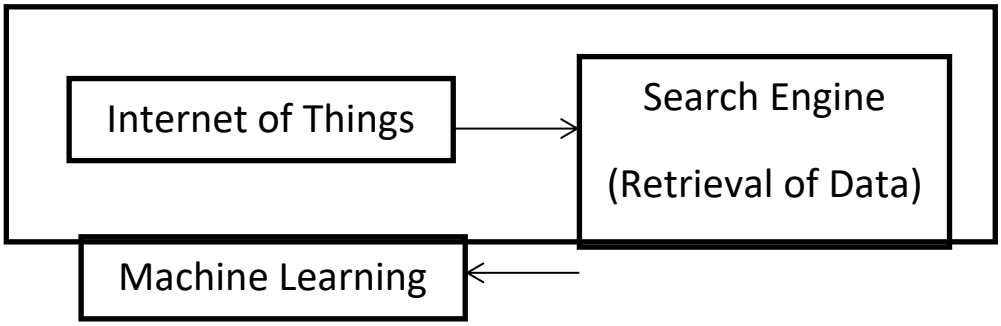

Fig2: Architecture of IoT- based Semantic Search

Step 1: Extract the files from database.

Step 2: Build a model that predicts a string from code such as code encoder.

Step 3: Build language model such as sentence Encoder.

Step 4: Tune code encoder to map code into a shared vector space with natural language.

Step 5: Create a Sprightly search.

2. Obtain the Data

In this stage, data collection is done from the open source GitHub data warehouse and deposited in the BigQuery. The GitHub is a trustable data repository for all types of data science projects. The data that is stored in BigQuery is perfect for the data retrieval using the SQL queries to retrieve the required files and the metadata on these files for further analysis. Instructions for the retrieval of these files are provided for the users, these data entities are represented in the form of (code, docstring) pairs [6]. To make it to work the code needs to be the developed at the top level methods or functions. These pairs are used in the form of training data for the code capture model, which is a tremendous task. Fortunately, Python library posses a package called AST which helps in capture functions, methods and docstrings [7][8]. The data stored in the BigQuery is prepared for modeling, training, validation and testing. The files are also managed to keep track of the original (code, docstring) pair. Then, the same model is applied to the code which does not have the docstring and it is saved separately since it is needed to search for code as well. These series of the stages are to extract the necessary information from the data applying the Machine Learning Techniques.

i. A raw corpus is created and a set of rules based on the corpus is framed to classify whether the crawled text document matches or not.

ii. A classification strategy is used to classify the extracted dataset to see that whether it belongs to a specified class or not. In the process supervised Machine Learning methods are used.

iii. Training data is very essential to run the supervised model. Training data involves manually collected data from various sources across multiple objects. After which, the training data holds the current dataset tagged to represent the category of the file.

iv. This training data is used to run the supervised model. A single application of this model on crawled data can classify a new document that it belongs to which specific category among all extracted.

3. Train a prototype to cipher the Natural Language Phrases

In this stage a mechanism is created for representing code as a vector, next there is a need for a similar mechanism for encode the natural language phrases present in doc string and search queries. There exist a numerous general-purpose models that created high-quality phrase embeddings called sentence embeddings. The proposed work gives a great overview in the field of sentence embeddings. One of the examples is Google's Universal Sentence Encoder works well in many applications and is available on Tensorflow Hub [9]. There are many ways of sentence embeddings, which ranges from the simple methods to say word vector averaging to the more sophisticated methods used in the construction of universal sentence embeddings. In this work Natural Language model is used to create embeddings for sentences. While building the language model, it is important to carefully consider the corpus used for the training.

Ideally it will be helpful to use the corpus that possess a similar domain to the underlying problem so that it can adequately capture the relevant semantics and vocabulary. An example of a correct corpus would be the stack overflow data, as it is an open forum where much discussion takes place on the code. This is suboptimal, since the discussion about the stack overflow often contains more semantic information than that contained in single-line documentation strings. After the language pattern is trained the next task is to apply that pattern to embedded every sentence [10][11]. A simple method could be to capture the hidden pattern of a language such as concrete pooling method. The architectural design for Sprightly Semantic Search is shown in the following Figure 3:

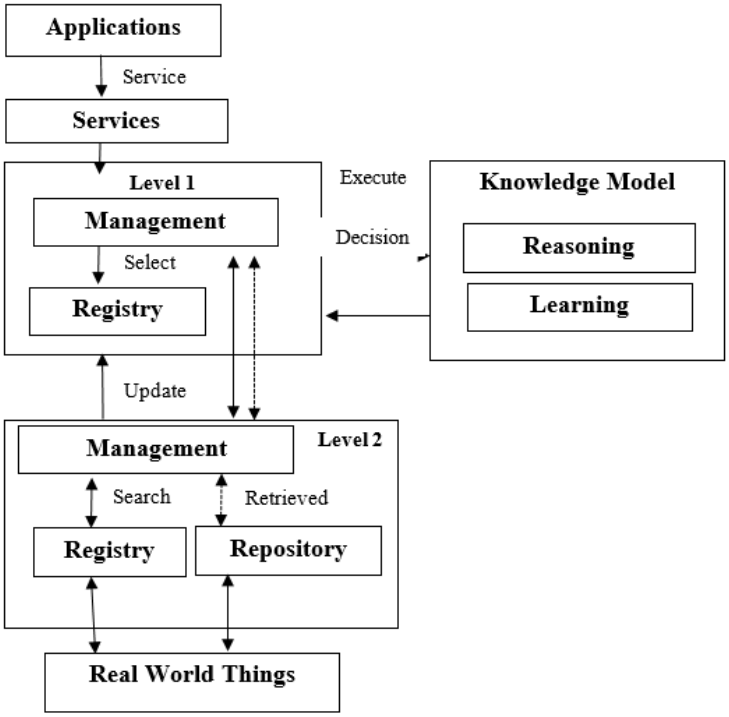

Fig 3: System Architecture for Sprightly Semantic Search

This pattern is used to measure embedding skill based on text or object similarity and is an excellent method for evaluating sentence embeddings [12]. In many cases, measuring the quality of the embedding using common benchmarks are among the one described here. However, most of the data is domain-specific, so the universal benchmarks probably not suitable for this problem. Unfortunately, the set of open source tasks for this domain has not yet been created. In the absence of such a bottom-up task, one needs to take care of sensibly checking whether these embeddings possess semantic information by observing the similarities among phrases that are known to be identical [13]. Note that this is an integrity check only. This is a more rigorous approach to measure the impact of this embedding on various ground tasks, thereby giving a more objective impression of the quality of the embedding. Define them as two fully connected n-layer networks of the form :

$\sigma a(la) = X1 \times ReLU(X2 \times \cdots ReLU(Xn \times la))$

$\sigma b(lb) = Y1 \times ReLU(Y2 \times \cdots ReLU(Yn \times lb))$

Where n is the embedding depth of each vertex, Xi and Yi are p × p dimensional parameter matrices for every layer of two fully-connected networks.

4. Build a Summarizer using a P2P Model:

Model-to-pattern formatting is to capture code theoretically similar to the GitHub Issue. Summarizer previously offered, except that use Python code in place of issue body and docstrings in place of issue names. Code, on the other hand, is not a natural language, unlike GitHub issue text. Domain-specific optimizations like Tree-Based P2P model and Syntax-aware Tokenization [14] [15] can be included to fully utilize the data in the code. In the proposed method the information is kept straightforward and the code in written in plain English. It should be mentioned that there are other methods which can be employed to develop a feature extractor for code [16] besides training the P2P model to extract code. The dataset can be trained, for instance, and discrimination can be used as a feature extractor. The subsequent figure 4 shows the procedure for building a summarizer using a P2P model:

- **Actual input -> Data collected from user (tokenized code)**
- **Original output -> After each test method, cleaning the user instances (Actual Document)**
- **Predicted output -> Clean up the data after each test.**
-

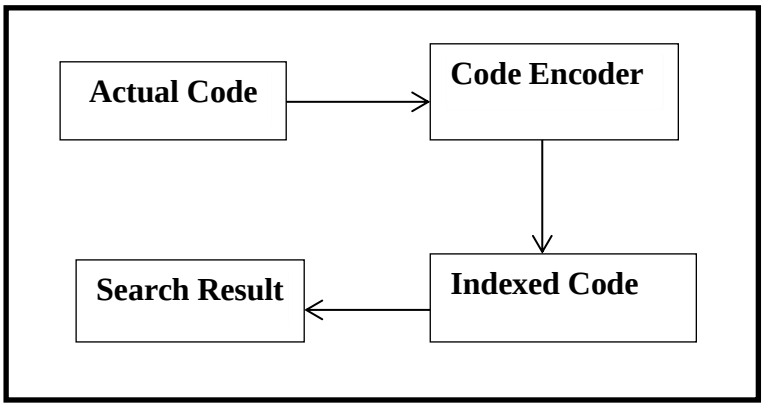

Fig 4: Procedural diagram for build a summarizer

The sample code for build a summarizer using P2P model is as follows:

class search_engine:

def __init__(self, nmslib_index, re_df, P2P_func)

nmslib_index :nmslib object

ref_df :pandas.DataFrame

P2P_func : callable

assert 'url' in re_df.columns

assert 'code' in re_df.columns

self.search_index = nmslib_index

self.ref_df = re_df

self.P2P_func = P2P_func

def search(self, str_search, k=2):

str_search : str

k : int

query = self.P2P_func(str_search)

idxs, dists = self.search_index.knn(query, k=k)

for idx, dist in zip(idxs, dists):

code = self.re_df.iloc[idx].code

url = self.re_df.iloc[idx].url

print(f'cosinedist:{dist:.4f} url: {url}\n----------\n')

print(code)

5. Implementation of sprightly search

The P2P is fine tuned to model so that it estimates the docstrings embedding rather than the docstrings themselves. All layers are freeze after training the model for the frozen version and train the model over a long period of time [17]. This will be very helpful in the model's further improvement for the quoted task. To assess how well this strategy applies to data we haven't previously encountered, we also vector code without docstrings. In order to create a search index, we want the code to be represented in the form of a vector at the end. We utilize the k text

library to pre-process this data using the same techniques we learnt in training [18]. [19]. Map code for the natural language's vector space using the P2P model. We can proceed to the final step now that the vectorized code has been collected. The figure 5 illustrates how to insert these vectors into the search index so that the nearest neighbors are readily found. Nmslib is a useful Python package for quicker neighborhood searches. To benefit from speedier searches while using nmslib, the search index must be pre-computed.

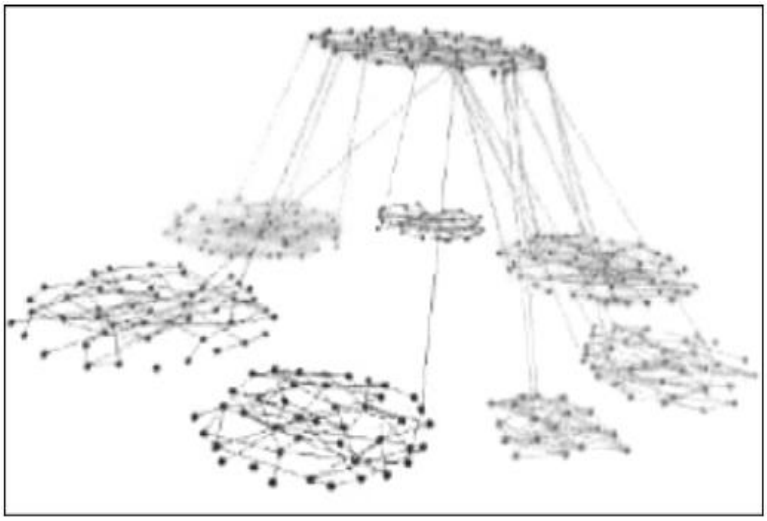

Fig 5: Clustering graph of Semantic Search

After creating the search index from the code vector, we need a way to convert the string (query) to a vector using the proposed model. To facilitate this process, we have provided a utility class called P2P under lang_model_utils.py. Finally, once after converting the string to a query vector, it is easy to get the nearest neighbors of that vector. A search index provides two elements. A list of nearest neighbor integer pointers in the dataset and their distance from the query vector (using cosine distance in this case).

6. Comparison with existing model

It is obvious that the current model will fall short of the demands of the suggested system. We should aim for at least 10 frames per second in order to achieve object detection that is even somewhat close to real-time. Given this, the current model falls short of the 10 fps goal. However, the second iteration of this gadget, which is substantially more potent, is worth attempting [21] [22]. However, in the case of edge processing, the P2P model appears to be a workable option, as illustrated in figure 6. This model allows for a trade-off between inference time and average accuracy to attain the desired goal with 80% accuracy. We can also see that the least and most powerful gadget when we look in terms of frames per second used for testing is the most power inefficient [23].The below table 1 depicts the different models with different performance accuracy.

Model	**Testing (%)**	**Accuracy (%)**
Delta Scoring	58	60
Lexicon based data	62	65
Revised Mutual Information	67	69
Proposed P2P	78	80

Table 1: Different models with different accuracy

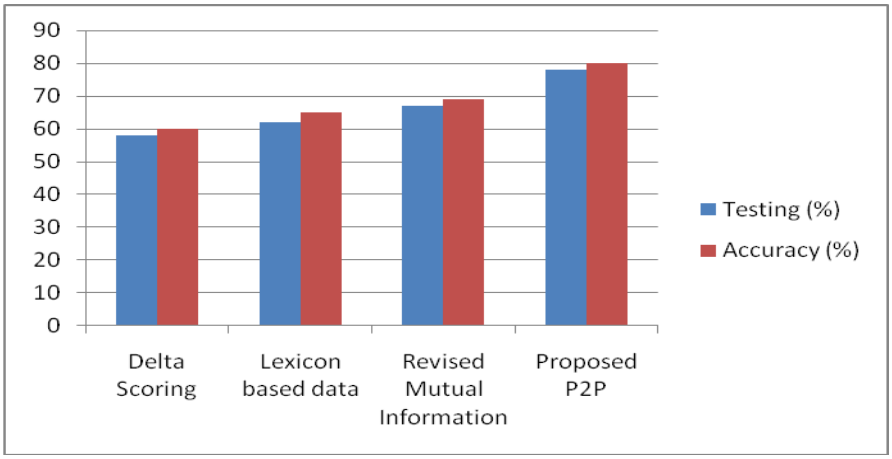

Fig 6: Comparison of Existing model with P2P model

Initially, there is a need to track the object embedded on the client server using the P2P model on any one of the devices. Further, there arises a need for a classifier to process human profiling (i.e., gender, age classification). Later it is needed to compare the edge implementation of the system along with the client-server versions. It is worth investigating the two hybrid version of the above [24]. Once the computer vision pipeline is ready with the required algorithms, there is a need to focus on the GUI applications to create an interface that allows users to easily collect and manipulate the captured data.

7. Conclusion

IoT is the emerging technology which includes all smart devices connected to each other through internet having the computation and communication capability, and possessing the ability of transmitting large amounts of data. The smart organization is one of the fortunate applications of Internet of Things containing the services across various domains including mobility, scheduling, power and many more. These services can be optimized and enhanced by making proper analysis of the smart data collected from all application areas. Many novel data analysis algorithms can be applied on this collected data to extract the meaningful information called as knowledge. In this paper a web browsing-based search engine for the agile semantic Web of Things is proposed and a scalable and convenient way to find their semantic data worldwide using real-time web-connected embedded devices. A thriving semantic search engine that can find data endpoints connected to search engines and search through P2P enabled devices and their services is proposed. This paper describes the framework and implementation of the P2P model and demonstrates its functionality and performance over the Internet through an evaluation process. Finally, a GUI and API are currently under development with a clear and well-defined structure/organization to expose P2P services on the web and make them easily accessible to the common user. Experimental results show that our system is scalable and can match a large number of data domains with various IoT applications with an accuracy of 80%.

8. References

[1] Janani R, Vijayarani S (2019) Text document clustering using spectral clustering algorithm with particle swarm optimization. Expert Syst Appl 134:192–200

[2] Gulnashin F, Sharma I, Sharma H (2019) A new deterministic method of initializing spherical K-means for document clustering. Springer, Berlin, pp 149–155

[3] Kushwaha N, Pant M (2018) Link based bpso for feature selection in big data text clustering. Future GenerComput Syst 82:190–199

[4] Garg N, Gupta R (2018) Performance evaluation of new text mining method based on GA and K-means clustering algorithm. Springer, Berlin, pp 23–30

[5] Karaa WBA, Ashour AS, Sassi DB, Roy P, Kausar N, Dey N (2016) Medline text mining: an enhancement genetic algorithm based approach for document clustering. Springer, Berlin, pp 267–287

[6] Y. Zhou, S. De, W. Wang, and K. Moessner. Search techniques for the web of things: A taxonomy and survey. 16(5):600, 2016.

[7] Lin H, Sun B, Wu J, Xiong H (2016) Topic detection from short text: a term-based consensus clustering method. In: 2016 13th international conference on service systems and service management (ICSSSM), IEEE, pp 1–6

[8] A. Jara et al. Semantic web of things: an analysis of the application semantics for the iot moving towards the iot convergence. International Journal of Web and Grid Services, 10(2-3):244–272, 2014.

[9] Thakran Y, Toshniwal D (2014) A novel agglomerative hierarchical approach for clustering in medical databases. Springer, Berlin, pp 245–252

[10] Zhu F, Patumcharoenpol P, Zhang C, Yang Y, Chan J, Meechai A, Vongsangnak W, Shen B (2013) Biomedical text mining and its applications in cancer research. J Biomed Informatics 46(2):200–211

[11] Davoodi E, Kianmehr K, Afsharchi M (2013) A semantic social network-based expert recommender system. Appl Intell 39(1):1–13

[12] M. Ruta, F. Scioscia, and E. Di Sciascio. Enabling the semantic web of things: Framework and architecture. In ICSC, pages 345–347, 2012.

[13] M. Compton et al. The SSN ontology of the W3C semantic sensor network incubator group. Web Semantics: Science, Services and Agents on the World Wide Web, 17(1):25–32, 2012.

[14] A. Kamilaris, V. Trifa, and A. Pitsillides. The Smart Home meets the Web of Things. IJAHUC Journal, 7(3):145–154, 2011.

[15] M. Sailaja, Abdul Ahad, Ali Hussain, "Machine Learning Medical Resources Allocation", Journal of Physics: Conference Series, AMSE (2021).

[16] D. Guinard, V. Trifa, and E. Wilde. Architecting a Mashable Open World Wide Web of Things. Technical Report No. 663, ETH Zurich, February 2010.

[17] D. Pfisterer et al. Spitfire: toward a semantic web of things. IEEE Communications Magazine, 49(11):40–48, 2011.

[18] Fu-Ming Huang et al. "Intelligent Search Engine with Semantic Technologies" International journal of Web & Semantic Technology (IJWesT) Vol.2, No.1, January 2011

[19] J. Pschorr, C. Henson, H. Patni, and A. Sheth. Sensor discovery on linked data. Kno.e.sis, Technical Report, 2010.

[20] D. Guinard, V. Trifa, and E. Wilde. Architecting a Mashable Open World Wide Web of Things. Technical Report No. 663, ETH Zurich, February 2010.

[21] A. Perez, M. Labrador, and S. Barbeau. G-Sense: A scalable architecture for global sensing and monitoring. IEEE Network, 24(4):57–64, July 2010.

[22] Abdul Ahad et., "The Substructure for estimation of miscellaneous data failures using distriuted clustering techniques", Proceeding of International Conference on Information Technology and Applications, Springer Link (2022).

[23] Aljaber B, Stokes N, Bailey J, Pei J (2010) Document clustering of scientific texts using citation contexts. Inf Retrieval 13(2):101–131

[24] D. Tümer, M. A. Shah, and Y. Bitirim, An Empirical Evaluation on Semantic Search Performance of Keyword-Based and Semantic Search Engines: Google, Yahoo, Msn and Hakia, 2009 4th International Conference on Internet Monitoring and Protection (ICIMP '09) 2009.

[25] H. Dietze and M. Schroeder, GoWeb: a semantic search engine for the life science web. BMC bioinformatics,Vol. 10, No. Suppl 10, pp. S7, 2009.