Ontology based Semantic Query Expansion for Searching Queries in Programming Domain

Manal Anwer Khedr[1], Fatma A. El-Licy[2]
Dept. Computer Sciences
Faculty of Graduate Studies for Statistical Research
Giza, Egypt

Akram Salah[3]
Dept. Computer Sciences
Faculty of Artificial Intelligence
Giza, Egypt

***Abstract*—Information on the web is growing rapidly; learning programming languages has become one of the most widely searched topics. Programmers and people of several backgrounds are now using Web search engines to acquire programming information, including information about a specific language or topic. Nonetheless, due to a lack of programming knowledge, many laypeople have difficulties in forming appropriate queries to articulate their inquiries, so they used to write short queries, which leads to vocabulary problems and contextless query. Besides, semantics is almost neglected in traditional search query since it is just keyword-based searches, which deem their search to be imprecise with irrelevant results, due to the use of unclear keywords. A Semantic query expansion method is proposed for disambiguating queries in computer programming domain using ontology. The integration of Cosine similarity method into the proposed model improved the expanded query, and accordingly, the searching results. The proposed system has been tested with several ambiguous and misspelled queries and the generated extensions proved to retrieve more relevant results, when applied to a search engine. The quality of the retrieved results for the expanded queries, are much higher than that for the crude queries. The proposed technique was implemented, and then tested with external independent testers. They confirmed the mechanical test results and displayed its improvement with an average of precision @10 82.2% and 91.1% respectively. The results were promising and therefore open further research directions.**

Keywords*—Query expansion; ontology; semantic search; short queries; cosine similarity

I. INTRODUCTION

There is a massive amount of data available on the web, and it is growing, freely, by the seconds. Yet, this free information-growth has not been assessed by a reliable information retrieval tool. A large percentage of the results obtained from web-search engines are not satisfactory relevant results. The main reasons for this defect are the lack of searcher's knowledge, the usage of short queries, and the intrinsic complexity of forming appropriate queries. These challenges lead to ambiguous queries, and accordingly, the retrieval of irrelevant results. Surveys indicate that almost ***25%*** of web searchers are, in most cases, unable to find useful results in the first set of URLs that was retrieved by the search engine [1].

It is not always easy to specify the needed information using exact query terms, because of the vocabulary problem. This problem became even harder due to polysemy and synonymy [2].

To obtain more relevant documents, queries must be disambiguated by looking at their context. Query expansion techniques were employed to overcome some types of these ambiguities. These techniques range from employing relevance feedback mechanisms, to, the utilization of knowledge base models, such as ontologies to resolve ambiguities [3].

Some of the search engines have employed query refinement methods to address short query problems. The idea is that, whenever a query is ambiguous or misspelled, the search engine suggests, to the user, some more refined, narrowed and less ambiguous queries. This method, however, suffers from several limitations and shortcomings [4]. First, selecting and retrying these refined queries induced significant additional difficulties and load upon the user. Second, since these refined queries are, usually queries that were submitted by other users in the search log, it most properly, would not reflect his intended searching objective. Third, such refined queries are not utilized to generate or improve the final results presented to the user, and hence do not lead to perceive improvement in the new search results list. Finally, the refined queries, as such, are obtained by keyword matching against the original short query, and thus are unrelated in meaning, or still very ambiguous. The user would have to face many longer but unrelated queries. Therefore, with these presented limitations, the authors propose a system aimed at reducing end users' efforts and tolerating their errors, by expanding their query into a prespecified domain. The specified domain knowledge is extracted from knowledge base.

The proposed approach disambiguated queries and related data retrieval by generating a semantic formulation for the query in the selected domain. This system works as both ontology-based query converter and text-based search engine.

The objective of the presented system is to expand short and vague queries using ontology. The ontology language used in the proposed system is based on the Resource Description Framework (RDF) structure, which is written in eXtensible Markup Language (XML) [5]. To expand vague queries in a prespecified domain, ontological terms are to be extracted from a predefined vocabulary of the specified domain.

This paper is structured as follows: Section 2, presents related work, Section 3 discusses Query Expansion, Section 4, describes the proposed system, named Ob-SQE, Section 5,

demonstrates system design and implementation, Section 6 evaluates the system. Section 7 displays results and discussion. Finally, conclusions are presented in Section 8.

II. Related Work

There is a great deal of work that has been accomplished by researchers in the field of query expansion. Most of them rely on external sources to provide support for query expansion. Some of those resources are syntactic and others are semantic. QE, often, bridges the vocabulary gaps between queries and web resources [6].

B. Sun et. al., [4] presented a method for refining short queries. They used query retrieval models that construct related multiple derived queries for the user's query. They ranked each of the derived queries according to its similarity to the user's query. This method is useful for improving query refinement and constructing the final results.

H. Imran et. al., [7] constructed thesaurus as a help tool for, semantically, selecting related terms to expand a given query.

J. Sarmah et. Al., [8] developed a system for retrieving Assamase unstructured documents from digital libraries. They used Assamase WordNet and query expansion techniques, to extend the user's original query.

R. Khan et al., [9] proposed a novel approach for the query expansion using WordNet and ConceptNet, which discarded the less important words from the query semantic space.

H. Tran [10] provided a method to match a company's human resources with job assignments received from clients. The author used ontologies as a solution to implement a query augmentation that improved the defining of the context by adding suggestions of relevant words.

S. Rajasurya et al., [11], developed a semantic search engine in the University domain. The authors used ontology as a knowledge base for the information retrieval. It was one layer above a search engine that retrieves more relevant results by analyzing just the keywords.

B. Al-Khateeb et al., [12], proposed enhanced system to expand the user's query using WordNet and generate a list of queries. The best result was, then, selected and retrieved to the user.

Based on the above related work, the usage of WordNet in query expansion is still poor for domain specific because it aims at covering general English words. That is to say, it is mainly language relation rather than semantic or context relation.

Utilization of ontology and semantic are obvious answer for establishing a successful and viable query expansion process, whereas semantic web provides tools and languages to facilitate machine readable access to ontology, such as RDF and Web Ontology Language (OWL).

III. Query Expansion

Query Expansion (QE) also known as query augmentation, is the task of adding new meaningful terms to the original query to increase the number of relevant retrieved documents. The significant problem of query expansion is the choice of the expansion keywords. Terms that are similar and relevant to query terms are usually considered as good terms for expansion. The process of adding terms can either be manual, automatic or user assisted. Manual query expansion relies on user expertise to make decisions on which terms to include in the new query. In the case of automatic query expansion, weights are calculated for all terms and the terms which have the highest weight are added to the initial query. With user-assisted query expansion, the system generates a set of possible query expansion terms for the user to select from [3].

Several semantic QE approaches are available, including linguistic, ontology-based and mixed mode, for addressing the vocabulary mismatch problem [13]. The main advantage of semantic QE using the ontology domain is that the knowledge structure is always available in the expansion mechanism, which clarifies the context of the query. Therefore, it was adopted in the proposed system to expand vague queries.

In this semantic QE, the query term is expanded with super categories in the knowledge hierarchy, then the expanded terms are processed by search engines in a Boolean way using *AND* operator.

The main shortcoming of using ontology as a QE approach is that it relies on exact matching between the query and the ontology vocabulary, which is a complex task for user queries that are written in natural language. To overcome this shortage, techniques are used to measure the similarity to decide which term or expression from ontology is the closest to the user's query.

IV. Proposed System

The proposed solution is an Ontology-based Semantic Query Expansion (Ob-SQE) system which is an assistance tool for crude searchers. It facilitates end user's searching process in obtaining their intended information in a specific domain. Ob-SQE is an ontology-based Information Retrieval (IR) system that employs ontology and semantic knowledge to expand the short and/or vague queries. Therefore, increases the chances of the search engine to retrieve true positive and relevant results. Ob-SQE system facilitates the searching tasks for Internet users, by targeting the search onto specific domain, and expanding the search query to be significant and relevant to that domain of interest.

The Ob-SQE system is not intended to change the current keyword-based search, but rather, strengthen it by supplements keyword from the relevant context domain.

A. Modeling the Ontology Domain

Fig. 1 illustrates the main steps of generating ontology for a given domain. In which, the concepts in the domain are identified, organized and represented & evaluated.

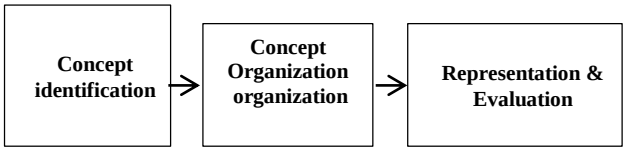

Fig. 1. Summary of Development Ontology.

- Concept identification: Identifies the domain and prepares a list of possible concepts to be included in the domain ontology. A manual mechanism was applied to extract the domain terms from various online resources [14, 15]. The concepts employed in ontology are those core concepts of the domain of interest.
- Concept organization: Determines the key concepts, such as the upper-level concept and the commonly used concepts in this field, in order to establish the core concept sets. Then, it defines the relations and assigns them to concepts, which are usually, "is-a" relations. The ontology organized the domain items into a hierarchical tree, in which nodes of the ontology are concept words, and the edge is the relationship between ontology concepts. Moreover, it selects the top-down structure approach to be utilized to be consistent with the format in which information was provided in the original resources.
- Representation & Evaluation: The generated concepts are organized into the ontology representation using editor and be adopted for defining classes and class hierarchy. The ontology class hierarchy demonstrates the relationships between upper-level and lower-level classes. The type of the created ontology is Light-weight Ontology, which is hierarchical or classificatory. It was selected because a light-weight Ontology does not include too many or too complicated relationships [16]. Then evaluating the consistency and checking the correctness and validity of the generated ontology using the system reasoning to be sure that ontology does not allow any contradictions.

The ontologies (for exercising the proposed system) were developed using the Protégé editor in OWL/RDF to define classes and class hierarchy. There are two created ontologies; one of them covered Java 2 Standard Edition (J2SE) to be used as a context for learning Java Programming and another for learning Python Programming, the user can use them interchangeably. The system is designed for multiple courses with multiple ontologies.

B. System Architecture

Fig. 2 illustrates the architecture and the processes involved in the proposed model, each of which is described in the following subsections.

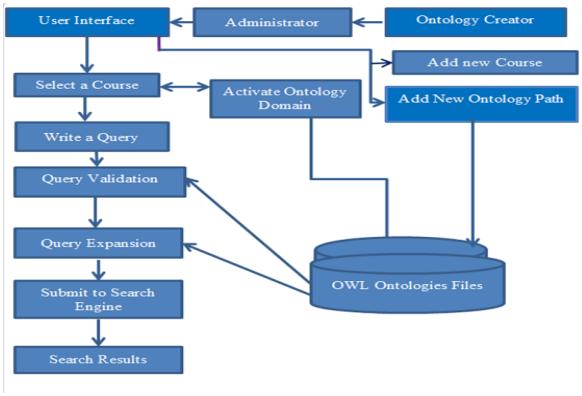

Fig. 2. Architecture of the Proposed System.

To enrich the learning process in general (as this system is portable for any domain topic assuming the availability of the domain ontology), the system architecture and the designed processes were presented as such to help the remotely learning individuals.

The first stage for creating a course is to Modeling the Ontology for the Domain of Interest. The ontologies were developed using the Protégé editor in OWL/RDF to define classes and class hierarchy. There are two created ontologies; one of them covered Java 2 Standard Edition (J2SE) to be used as a context for learning Java Programming and another for learning Python Programming, the user can use them interchangeably. The system is designed for multiple courses with multiple ontologies. The processes involved in the system design are:

1) Creating new course: The administrator can create a new course by adding a new course name and adding a newly created ontology path to these course concepts.

2) Course selection: Through this layer, the user select a course, which invokes and activates the corresponding ontology knowledge base. The user, may then, type a query to be searched.

3) Query validation: In this layer, the query is processed for validation and is utilized to extract a matched ontology term with the user's query. This stage performs the following tasks:

a) Check for a non-empty query.

b) Validate the search query to match ontology classes prior to the expansion phase to guarantee that the query belongs to or close to the domain of interest using a similarity method. An error message is invoked for invalid queries.

c) Cosine similarity method was adopted to measure the highest, syntactically similar key word. This step is to overcome the exact matching to ontology concepts and misspelling problems.

d) Keep the matched term with ontology terms if it does exist.

Result from this phase is, either, the matched concept with the highest similarity, or, a displayed error message.

4) Expanding the query: It provides the semantic for initial query concept by adding new related terms from knowledge ontology files. Expanding process starts with a specific matched term as a bottom-up direction on one path to obtain all super direct and indirect classes to be concatenated together by AND operator [17]. Moreover, it adds any available synonyms for the concept. Synonymy means that concepts are equivalent with the matched term. Consequently the context of the searched concept is determined by an extracted set of, directly and indirectly, linked concepts (in the ontology) by explicit relations including synonyms.

5) Applying the extended query: The expanded query is to be utilized and redirected to Google search engine to obtain the results of its semantic search. Thereafter, the list of the obtained web pages is retrieved to the user.

V. System Design and Implementation for the *Ob-SQE* Model

The proposed system was designed and implemented into the model prototype Ob-SQE as demonstrated in Fig. 3.

The implementation was performed by creating and executing web applications using 'JavaServer Faces' (JSF) as a GUI, and OWL API as a semantic web tool. It is capable of reading, writing, and processing data in domain ontology, via Java programming language within Eclipse editor. The system used the JSF API, to capture and process user's query through user's interface.

For the purpose of this article, the knowledge Ontology of two selected different programming languages were generated to be utilized in the Ob-SQE system, namely, Java 2 Standard Edition (J2SE) and Python Programming.

A. Running Example

This section presents a running example to illustrate the processes involved in the system components, including the creation of new course, query expansion and searching of a vague misspelled query.

1) Create new courses: The system administrator is responsible for creating new courses, by adding their Ontology knowledge file and its path to the system. The paths, files and course name are validated, and an error message is invoked for any inconsistencies.

2) Searching the domain terms: The implemented system is executed through the GUI interface. The following is a typical scenario for an execution session:

a) The user chooses a specific course, say, a Java Course.

b) According to his selection, course ontology will be activated by the system implicitly.

c) A user writes the query "Publik", through the GUI interface. Scenario of a contextual reformulation is applied by a system based on similarity.

d) The query is validated, the matching term "Public", is obtained.

e) OWL's method is applied to search the ontology for related terms with matched term. The matched terms were: "Modifiers", "Access Modifiers" and "Java Programing".

f) The verified query was expanded by combining the matched term with its related terms using AND operator obtaining the logical formula "Public AND Modifiers AND Access Modifiers AND Java Programing."

g) The expanded query was applied to Google search engine.

h) The retrieved websites were displayed for the user.

Fig. 4, illustrates the semantic context of the 'Public' concept as a Java term, which is formed by the concepts 'Modifiers', 'Access Modifiers' and 'Java Programing', (all of them are super classes (hypernyms) for the concept 'Public').

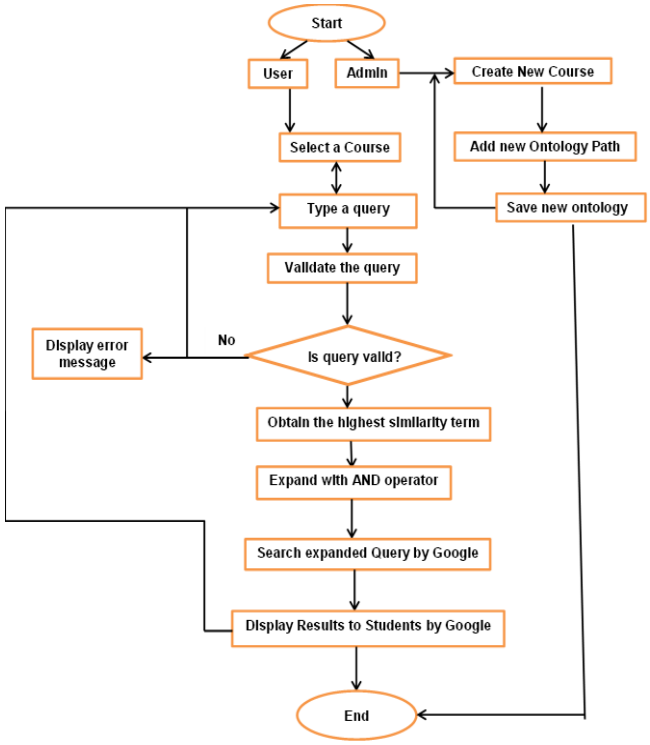

Fig. 3. The Flowchart of *Ob-SQE* Model.

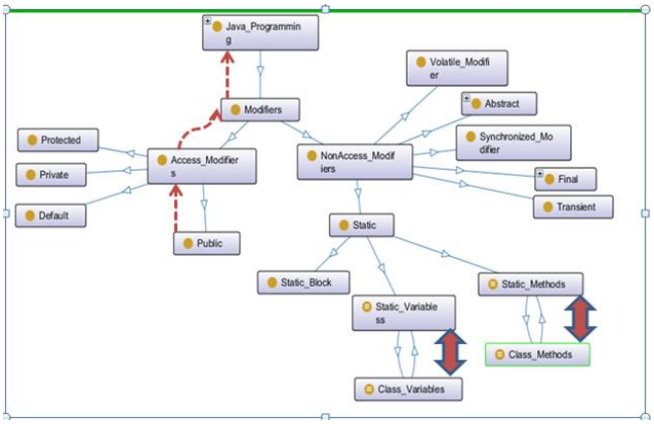

Fig. 4. Semantic Context of Java Concept 'Public'.

In addition, based on the ontology, if the searched concept has synonyms, the system will generate it. For example, Class Methods AND Static Methods are equivalent concepts, so whenever one of them is in the matched list, the other is added to as well. The bidirectional arrows near the bottom of Fig. 4 illustrate two of the equivalent concepts.

Fig. 5 illustrates the semantic context When the query term 'deque' is typed while the Python ontology Knowledge is active, it will be expanded through the super classes with AND operator to be:

"Deque AND Collections AND Python Programming".

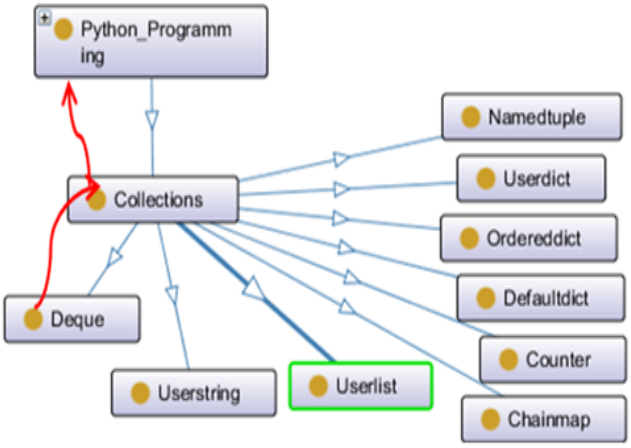

Fig. 5. Semantic Context of the Python Concept 'Deque'.

B. Similarity Method

Given a query and a set of documents, a traditional information retrieval model usually measures the similarity of a query and different documents, and then returns the documents with top-ranked similarities as the results. The Vector space Model (VSM), introduced by Z. Jian, [18], however, represented documents through the words that they are contained. In this model, a query and a set of terms from the ontology knowledge base- each is regarded as set of points (substrings) in the vector space. Every set of points represents a vector. The similarity between the query and the ontology term is realized by the cosine of the angle between these two vectors. The smaller is angle, the greater is similarity. Given two vectors A and B , Cosine similarity is computed using (1).

$$\text{COS}(\theta) = \frac{\sum_{i=1}^{n} ai * bi}{\sqrt{\sum_{i=0}^{n} a_i^2}\sqrt{\sum_{i=0}^{n} b_i^2}} \quad (1)$$

Where: a_i and b_i are the count of the i^{th} substring occurred in the first and second string, respectively.

VI. System Evaluation

A cut -off value of the first 10 results in search engine retrieval effectiveness tests seems reasonable, because search engine users, in most cases, only consider the first page of the retrieved results (usually displaying 10 organic results) [19].

Based on this observation, the effectiveness of the system was evaluated by comparing the average of fitness function and the precision of the top ranked ten results, retrieved by the Google search engine for a given query, and that of its expansion through the application of *Ob-SQE* system. This comparison experiment was repeated for several sets of queries. The system was, also, manually, exercised for evaluation by two external independent users.

A. Fitness Function (F.F)

Fitness function evaluates the quality of the results obtained from Google search engine for the queries before and after the application of the *Ob-SQE* system. The retrieved URLs are tested for judgment using a modified version of Al-Khateeb et al., [13] technique. Name, Title and the Description of each of the retrieved URL is extracted for calculating the similarity and recording the fitness score for the corresponding URL. The similarity (between each retrieved URL contents and user query) was computed using (1), whereas the fitness score was calculated using (2).

$$\text{Fitness} = A \text{ X Order} + B \text{ X Sim1} + C \text{ X Sim2} + D \text{ X Sim3} \quad (2)$$

Where: *Order* is the order of the returned results, *Sim1*, Sim2 and *sim3* are the Jaccard similarity between the query and the URL-Name, URL-Description and URL-Title, respectively. *A, B, C* and *D* are factors representing the degree of importance. *A=0.1, B=0.2, C=0.3 and D=0.4 .*

$$\text{Jaccard\_similarity}(S1, S2) = |S1 \cap S2| / |S1 \cup S2| \quad (3)$$

Where: *S1*and *S2* are the two strings considered for similarity.

Equation (4) computes the average of fitness for top ten ranked results.

$$\text{Average of Fitness} = \frac{\sum_{i=1}^{10} Fitness}{n} \quad (4)$$

B. Precision @ K

Precision is the proportion of retrieved documents that were relevant. Whereas Precision @ K is the proportion of the top-*K* documents that were relevant [20].

The four steps to compute Precision @*K* are:

1) Set a rank threshold *K*.
2) Compute the percentage of relevance in top *K*.
3) Ignore documents ranked lower than *K*.
4) Finally, compute the average of *P@K* for every query using (5).

$$\text{Average of P@K} = \frac{\sum_{1}^{k} p@k}{R} \quad (5)$$

Where *k* is the rank of each relevant document, *R* is a total number of relevant documents and *p@k* is the precision of the top-k retrieved documents [21].

C. System Evaluation

The system was evaluated by the application of the Average Fitness Function and Precision @K, for results retrieved from searching for a crude/initial query and that of its extension.

A prepared sample of queries from Java programming language was utilized to measure the effectiveness and accuracy of the system. Each query was expanded through *Ob-SQE* system. For each couple of queries (the query and its expansion) Google search engine was, then, invoked twice to search for the original crude query and its expanded version. The retrieved URLs for the original and the expanded queries were analyzed to obtain the averages of precision and the averages of fitness for the two set of results. Table I, demonstrates the average of precision and the average of fitness function of each query before and after expanding it through the *Ob-SQE* system.

TABLE I. THE RETRIEVAL ACCURACY OF GOOGLE SEARCH ENGINE BEFORE AND AFTER THE EXPANSION PROCESS (OB-SQE SYSTEM) FOR SELECTED QUERIES, K=10

Google' result before query Expansion			**Google results for the expanded query through *Ob-SQE* system**		
Initial Query	***Average* %**		***Expanded Query***	***Average* %**	
	F.F	***P@K***		***F.F***	***P@K***
Jav / java	61.39	0	Java Programming	74.16	100
Constructor	61.43	90.8	Constructor AND Methods AND Java Programming	70.04	100
Loop	65.5	0	Loop Statements AND Statements AND Control Statements AND Java Programming	74.12	100
List	63.87	20	Lists AND Java Programming AND Collections	73.49	100
Catch	60	0	Try Catch AND Java Programming AND Exception Handling	78.52	100
Abstraction	67.94	54.7	*Abstraction* AND Java Programming AND Object Oriented Concepts	76.87	100
For	60	0	*For* AND Loop Statements AND Statements AND Java Programming AND Control Statements	75.04	100

D. System Evaluation by External Users

The *Ob-SQE* system was evaluated by two users with a good background of java programming language. Every one of them applied 45 queries with a single word. Most of these queries are processed and expanded by ontology. Yet, some of queries could not be exercised by the system, because they were not included into the ontology domain. Table II is the evaluator's feedback, which presents the number of satisfied expanded queries (therefore satisfied retrieved information) and the percentage of their acceptance of *Ob-SQE* system.

For every query, the evaluators exercised *Ob-SQE* system to generate its expanded version, and then they utilized google to test and compare the first ranked 10 results for both the original crude query and its extension.

TABLE II. EVALUATORS' FEEDBACK BASED ON P@K: UTILIZING GOOGLE SEARCH ENGINE BEFORE AND AFTER *OB-SQE* APPLICATION, K = 10

Evaluators	**Number of queries**	**Satisfied retrieved results**	**Unsatisfied results**	**Average p@10 with *Ob-SQE* %**	**Average p@10 with *Google* %**
1	45	37	8	**82.2**	**12.2**
2	45	41	4	**91.1**	**18.6**
Average of acceptance for two users				**86.65**	**15.4**

VII. RESULT AND DISCUSSION

Query expansion is the optimization of the crude query. In this paper, the purpose of the experiment is to verify whether the precision of the query is improved or not. Terms added to the queries have been extracted from the ontology of the Java or Python domain which is partly described in Running Example as illustrated in Fig. 4 and Fig. 5. The tested queries samples showed the effects of utilizing the ontology in improve the accuracy and increases the relevancy of the results. The results of Google alone and those after applying Ob-SQE expansion system were evaluated using the average of fitness function and the average of Precision @10. Table II, illustrates a comparison of the retrieval accuracy of queries before and after its expansion by the proposed system.

Fig. 6 illustrates the average of fitness function for the tested queries within *Qb-SQE* system and Google search engine, whereas Fig. 7 displays the average of precision of the expanded queries through the *Qb-SQE* system and that of the crude queries in Google search engine.

The evaluation of obtained results from the proposed system, a semi-automatic process was used. Since there are no Google APIs that support exact retrieval like Google search engine, and planning in the future to be full-automatic.

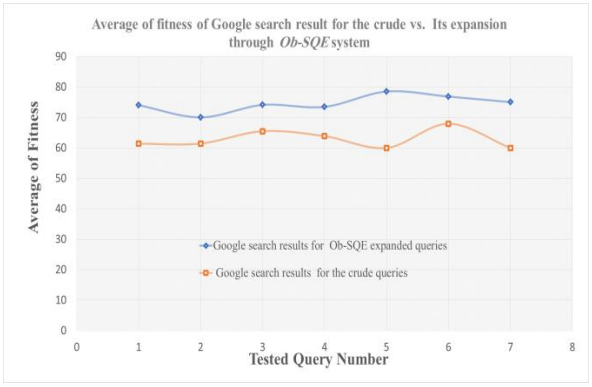

Fig. 6. Average of Fitness for the Google Searched Queries with and without *Ob-SQE* System Application.

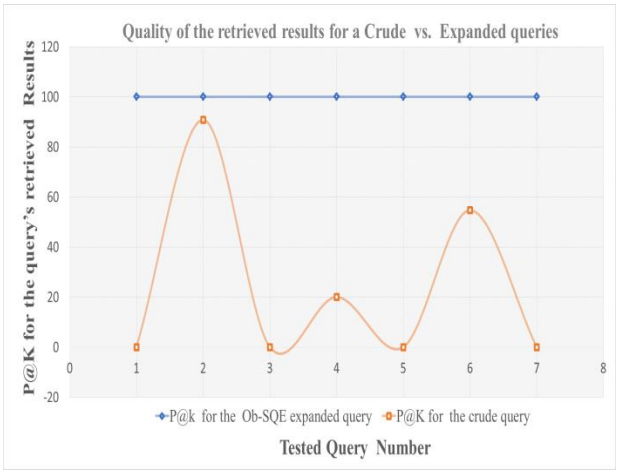

Fig. 7. Average of Precision for Google's Retrieved Results of Crude Query and its Expansion through *Ob-SQE* System.

VIII. Conclusion

The established semantic query expansion method is based on domain ontologies. It employed synonyms expansion, hypernyms expansion, and similar word expansion. The integration of similarity function into the system improved the formulated query and the quality of the searching results. The selection of computer programming language domain allowed for the manual evaluation of the system by independent and experience personals.

The developed domain ontologies improved the search process by adding context to search query. Supplying more related terms from the ontology domain cures the lack of context.

The proposed system has been tested with several number of ambiguous and misspelled queries. The semantic expansion process through the ontology domain generated extended queries, which, when executed, by the Google search engine, more relevant results were retrieved. The fitness function for the expanded queries achieved higher average (74.6) than that of the initial ones (62.30).

Average of precision @10 for the results obtained from applying the expansion (by the two external testers' personals) to a query were 82.2%, 91.1%; compared to the Google results for the original crude queries, which were 2.2%, 18.6%, respectively.

Derived ontologies are shareable and reusable. The proposed framework is applicable for any given domain, assuming the availability of the corresponding domain ontology which is developed based on super and subclasses. In future work, complete total ontology development of the domain concerned may be integrated with learning systems.

A future work may consider the end user to be more interactive with the system by creating a smart interface with him.

Also, one may consider the ontology itself to be generalized or redesigned to accommodate a higher level knowledge (like education, engineering, industry, etc.).

Since the Web is highly dynamic and forever evolving, it is essential to establish a dynamic methodology to update the knowledge ontology. Also, Auto-generated ontologies may be the solution to the time consuming and errors proning due to human errors. Furthermore, created ontologies may be integrated with learning systems to help users in the search process.

Acknowledgment

Our thanks and appreciation to Suzan Alsisi, Principle Advanced Software Engineer for Fusion Middleware at Oracle (Egypt) and Abd Allah Mohmed, .Net Team Leader at Infoglobe Company (Egypt), for their time and effort in testing the proposed system and providing their feedback.

References

[1] Mukhopadhyay, A. Banik, S. Mukherjee, J. Bhattacharya, "A Domain Specific Ontology Based Semantic Web Search Engine", 7th International Workshop on MST,pp. 1–9, 2011.

[2] J. Bhogal, A. MacFarlane, P. Smith, A Review of Ontology-Based Query Expansion," Information processing & management, vol. 43, pp. 866-886. 2007.

[3] S. Khan, J. Mustafa, "Effective semantic search using thematic similarity", Journal of King Saud University – Computer and Information Sciences, vol. 26, pp.161–169. 2014.

[4] B. Sun, P. Liu, Y. Zheng, "Short Query Refinement with Query Derivation", AIRS'08 Proceedings of the 4th Asia information retrieval conference on Information retrieval technology, pp. 620-625, 2008.

[5] K. Munir, M. A. Sheraz , The use of ontologies for effective knowledge modelling and information retrieval". Applied Computing and Informatics vol. 14, pp 116–126, 2018.

[6] M. A. Raza, R. Mokhtar, N. Ahmad, M. Pasha, U. Pasha, A Taxonomy and Survey of Semantic Approaches for Query Expansion, in IEEE, vol. 7, pp. 17823-17833, 2019.

[7] H. Fang, A Re-examination of Query Expansion Using Lexical Resources".Association for Computational Linguistics (ACL) pp.139–147 June 2008.

[8] H. Imran, A. Sharan, Thesaurus And Query Expansion. international Journalof Computer science & Information Technology (IJCSIT), vol 1, No 2, pp. 480-484, 2009.

[9] J. Sarmah, A. K. Barman, S. K. Sarma, WordNet Based Information Retrieval System for Assamese" Proceedings of Computer Modelling and Simulation (UKSim), 15th International Conferenc, pp. 89-97, 2013.

[10] R. Khan, J. Jaafar, Semantic Query Expansion Using Knowledge Based for Images Search and Retrieval. International Journal of Computer Science & Emerging Technologies Vol. 2, pp. 1-5, February 2011.

[11] H. Tran, Human resource matching through query augmentation for improving search context, Master thesis, 2016.

[12] S. Rajasurya, T. Muralidharan, S. Devi, S.Swamynathan, Semantic Information Retrieval Using Ontology in University Domain. International journal of Web & Semantic Technology. vol 3, pp.55-68, 2012.

[13] B. Al-Khateeb, A. J. Hilal, S. Al-Janabi, Web Search Enhancement Using WordNet Query Expansion Technique.Journal of University of Human development, pp.542-547, 2016.

[14] K. Sierra and B. Bates. Head First Java, 2nd Edition. O'Reilly & Associates, Inc., USA. March 2005.

[15] P. Naik, K. Oza. Python with Spyder: An Experiential Learning Perspective. (2019).

[16] T. Slimani, A Study Investigating Typical Concepts and Guidelines for Ontology Building.Journal of Emerging Trends in Computing and Information Sciences.Vol. 5, pp. 886-893, 2014.

[17] M. Bello Aliyu. Efficiency of Boolean Search strings for Information Retrieval, 2017, American Journal of Engineering Research (AJER). vol.(6), Issue-11, PP. , 216-222.

[18] Z., Jian, Web Information Retrieval based on Ontology. Proceedings of SPIE - The International Society for Optical Engineering. Vol. 8768 87683N-1. pp 1-5,2013.

[19] L. Dirk, Evaluating the Retrieval Effectiveness of Web Search Engines Using a Representative Query Sample. Journal of the Association for Information Science and Technology. 66. 10.1002/asi.23304, 2014.

[20] N., Craswell, S., Robertson. Average Precision at n. In: LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-39940-9_487, 2009.

[21] E., Zhang, Y., Zhang. Average Precision. In: LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-39940-9_482, 2009.