Implementing Semantic Search in Decision Support System

Mariya Zhekova[1], George Pashev[2], George Totkov[3]

[1]*University of Food Technologies, 26 Maritsa Blvd., 4000 Plovdiv, Bulgaria*
[2] *University of Plovdiv "PaisiiHilendarski", 24 TzarAsen Str., 4000 Plovdiv, Bulgaria*
[3] *National Evaluation and Accreditation Agency, 125, Tzarigradsko Chaussee Blvd., bl. 5, 1133 Sofia, Bulgaria*

Abstract

A model of a software tool for semantic text search and retrieval of information about objects, implemented in Python, is presented. The computer processing of natural user text and understanding of the semantics of the text so that it gives useful answers for the users is a task of the natural language processing, by means of which the results in the research have been achieved. A main place in the article is a description of an algorithm and software implementation of a tool that with the help of natural language analysis copes with the task of transforming a user question into a vector in a multidimensional orthogonal space and answering user questions asked in natural language. For this purpose, a space of catalogued language objects necessary for linguistic processing has been created, organized in such a way as to provide direct answers to search queries. The presented software can also be accessed through a graphical user interface, but the researchers' work is focused on natural language understanding.

Keywords

natural language comprehension, natural language analysis, information retrieval

1 Introduction & Previous work

In modern technologies for AI, related to the retrieval of information using a natural language query to the database, it is no longer enough just to search by keywords and templates. Sometimes even the searched objects are not textual. Then the information is harder to find. Therefore, the approach proposed by the present study includes semantic text search techniques.

There is no general, motivated consensus for building a semantic search module on the semantic representation of the input user question. The user is free to ask their inquiry in natural language. The input data can be, for example, longer or shorter, single sentences with a linear structure, they can be strings to which more complex formatting is applied. The current article discusses a process algorithm and its software implementation, based on semantic search and answering questions in natural language. The experiments are aimed at typical objects from the database of the system, namely – teachers, disciplines, specialities, students and others. Initially, the experiments were based on hierarchically arranged relational test tables, with several levels of hierarchy, but later the NLI module was redirected to the newly created platform for quality assessment of higher education in Bulgaria, whose database is non-relational (MongoDB) corresponding to components of objects called indicators.

The software module for semantic text search proposed in the research is implemented in Python. The language was chosen because of the comprehensive set of libraries (Python packages), which also offers its functionality in the processing of linguistic data. Numpy libraries were used – for working with data types, nltk – for working and processing in natural languages, pandas – for working with large data sets, math – for mathematical functions and operations, sys – for access to variables and functions.

Education and Research in the Information Society, September 27–28, 2021, Plovdiv, Bulgaria
EMAIL: m_jekova@uft-plovdiv.bg; georgepashev@uni-plovdiv.bg; totkov@uni-plovdiv.bg
ORCID: 0000-0002-2501-4412; 0000-0001-8148-4737, 0000-0001-9413-3565

The English lexical corpus WordNet package in Python contains a list of concepts (synsets) with the same concept. Below in the research is presented a limited linguistic resource for the domain area, which, like the WordNet package, stores the words and phrases used in the software, enriched with metadata of heterogeneous nature, but containing synonymous sets covering the basic concepts forming the relations in the language database. Python is a dynamic and object-oriented language and as such allows you to add attributes, encapsulate and reuse data and methods. In it, each variable is an object that has specific attributes and methods.

There are semantic search tools that provide similar functionality for categorizing, retrieving, linking, and identifying objects designed and adapted for web-based platforms for many areas of knowledge. But a relatively small number of them are in the field of software engineering to assess the quality of higher education.

The **Nordlys** research prototype finds and retrieves specific target object types, using a database (MongoDB), machine learning, evaluation and retrieval tools. It also uses external tools for loading and pre-processing of standard datasets (DBpedia, Freebase, etc.) [3].

SPYSE (Semantic PYthon Search Engine) is a web-based search engine that searches for and retrieves information from the body of Python programs. SPYSE contains a file containing Python definitions and expressions. It is a collection of many program units such as classes, methods and variables [4]. To perform a successful search in the code of a Python application, information is extracted from the modules of each package, ie. a list of program objects for each module is obtained. Such as classes, functions and variables in classes, global functions, repeating internal functions, variables inside functions and global variables. Each type is identified using a separate key. The data extracted from each Python package is treated as a document and indexed. SPYSE also contains an NLP analyzer that normalizes the text before indexing. Based on the matches, it ranks the results and returns the fields as name, summary, keywords, description. Matches with the user request are searched in the following fields: class, class function, class variable, module variable. Different weights can be assigned to matches in different fields depending on the importance of the fields. Based on the matches, SPYSE ranks the results and returns the path to the module where the match occurred [4].

A prototype of the **SemanticFinder** semantic search engine is described in the article [5] for searching for information about software components hosted on various sites. The semantic search here is also based on a sufficiently comprehensive categorized controlled dictionary, including synonyms and acronyms. The dictionary unifies the terminology with appropriate components (tags) and facilitates the uniform search and comparison of objects. Provides a service that allows users to refine and narrow search results themselves. It also provides access to search history, which allows navigation in the search space by changing individual parameters. The architecture of the tool is divided into online and offline. The offline subsystem manages the crawling of the database by periodically updating it, and the online system performs the semantic search [5].

The semantic text search task has been transformed into a new task for converting a user question-query (string) into a vector. For this purpose, it uses its own personalized linguistically categorized limited vocabulary (language model), presented in previous publications of M. Zhekova. Once the string is turned into a vector, the closest hits to that vector in the vector space are searched.

The software tool is implemented in an independent module to the Information System for Quality Assessment of Higher Education in Bulgaria.

This module is intended to assist committee experts in deciding what assessment grades to give by asking questions in natural language about certain quantitative indicators or aggregated data on them and receiving answers that are automatically generated by it. To do this, an appropriate mathematical paradigm and algorithm had to be selected to strike a good balance between speed and the nature of the data being searched and returned. Due to the fact that the data in terms of volume is not very large, we could afford to use a less effective implementation, which, however, is easier as a technical implementation.

For the reasons we set out in the previous paragraph, we chose to use exactly the model of multidimensional orthogonal space with its own distance function and radius vectors in it. This model and algorithm are easy to program, especially in languages such as Python, which support embedded data types, such as lists and sets and operations with them. In addition, by better managing the distance function as well as its coefficients, we could somehow manage the error in finding the closest result, which is impossible in some other models.

2 Semantic search algorithm in NoSQL database

In this study, there is a place for analysis and generation of queries to both relational (SQL) and text (NoSQL) databases. Figure 1 shows the basic similarities and differences in the process of natural language processing of text as a whole. And below in the explanations is presented only one of the aspects, namely the semantic search in an unstructured resource. The steps required to generate an SQL query to a relational database type are described in other authors' articles (together and separately). The database selected for the experiments in the study is an alternative to the traditional relational database, in this case a CSV file (NoSQL), which does not consist of tables, but rather of collections that contain concepts from the domain area at the root level modelled in a tree hierarchical structure.

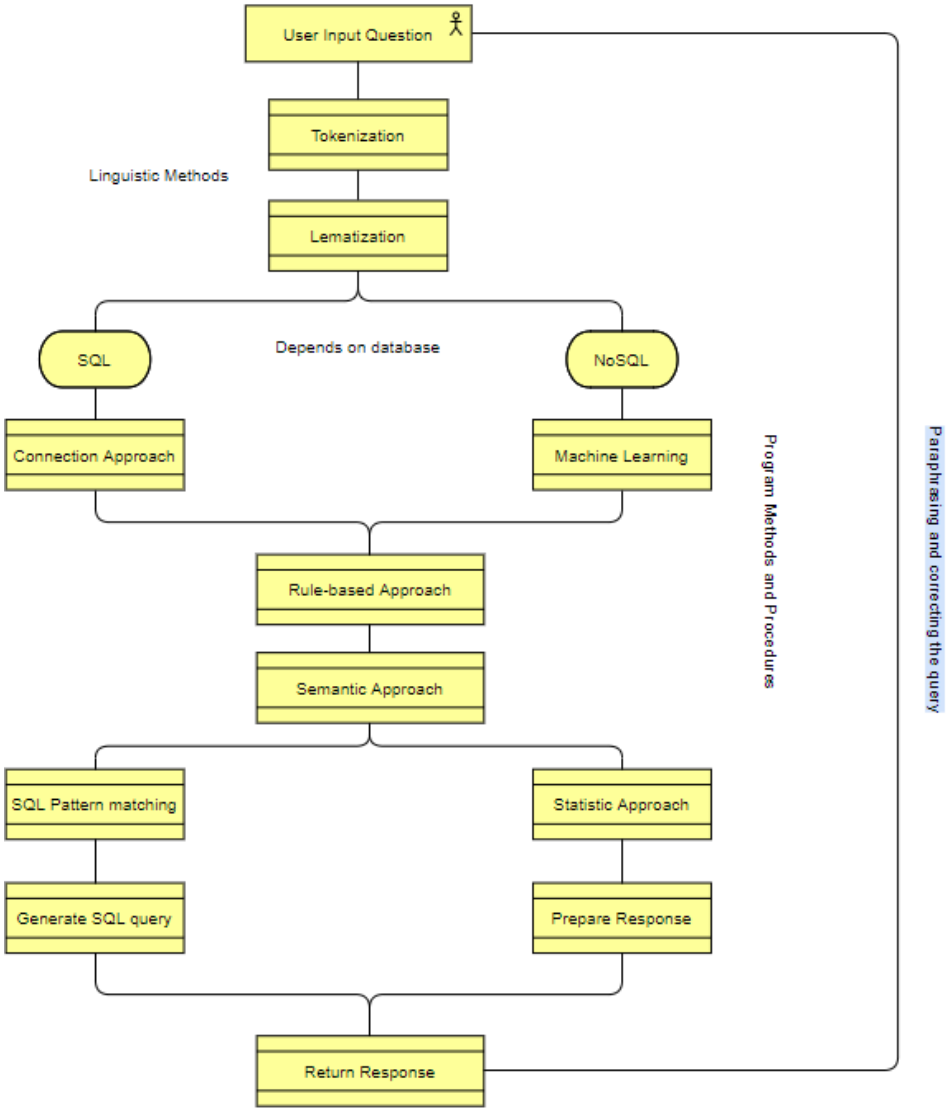

Figure 2. Model of cross-translator for processing natural language question and transforming it into a response from a database

The presented model has a dynamic and cyclical character, separated by the context of the inquiry, the intermediate results, the procedures for avoiding ambiguity and the need for paraphrasing and corrections. The dynamics of the process is presented with opportunities for paraphrasing and correction of the query as a result of accumulated new intermediate results and the course of procedures, observing

the conditions for their course. The cyclicality stems from the fact that natural language comprehension is not just a guide for consistent steps to achieve a certain result, but a systematic approach to combine key activities such as – analysis, identification of concepts and related specific characteristics and relationships, choice and defining a question template, generating and synthesizing a SQL query, methods for executing the query, returning a response. In practice, the methodology of natural language comprehension and the conversion of a custom query text into an SQL query is an iterative process. The key activities are actually iterations, not individual stages in processing. With each new user request (iteration), all activities are performed, each performed at a different level of detail.

The research aims to turn a user-asked question in natural language into a query to a database of an organization in the field of higher education in Bulgaria. Only the statistical approach is described in more detail, the software part of which is the algorithm proposed below. The other modules are not the subject of the researchers in this study.

When a random user text is received, the computer receives a string or vector of values $X = (x_1, x_2, \dots., x_n)$as a sample of a multivariate variable in the n-dimensional Euclidean space controlled by size and value R^n. The real essential data falls into the Phrase and Synsets fields. Synsets attributes are equivalent sets of base values for each concept, expressing the same concept, with which the user can replace a word in his speech. It is possible to load only a subselection of categories without caching the entire language file. Phrase and Synsets attributes are grouped in a tree structure. Hyperonyms or more abstract terms precede others in their category and provide a way to categorize and cluster concepts based on their affiliation. Eg. *Teachers* is a generic term (parent) or hyperonym of *Teachers who have acquired a new scientific degree and taken a new academic position*, also of *Habilitated Teachers*.

The proposed semantic search algorithm follows the following steps:

Step 1: generate vector space

Step 2: vectorization of the input user question-request

Step 3: transfer to vector

Step 4: place the vector in the search index

Step 5: retrieve the nearest neighbours (compare the obtained proximity vectors)

The set of all vectors in **Step 1**, formed as a sequence of tags from the user string, corresponding to its constituent words, phrases, service words (prepositions, conjunctions, particles, pronouns) from the natural language within the study, we will denote by X and will we call vector space. In this sense, X is a vector space, because in it every two vectors $a, b \in X$can be mapped to a non-negative real number called the distance between the vectors.

In the presented software model, the distance between two vectors $a = (\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_n)$ and $b = (\beta_1, \beta_2, \beta_3, \dots., \beta_n)$ the n-dimensional vector space means the number: $\rho(a, b) = |a - b| = \sqrt{(\alpha_1 - \beta_1)^2 + (\alpha_2 - \beta_2) + \cdots \dots + (\alpha_n - \beta_n)^2}$

3 Software implementation

The program code initially contains, loads and reads the linguistic file (1) and (2) with values such as e.g. grammars and reserved objects for further processing. **Step 1** of the algorithm is performed. The file contains a list of objects as a collection of their metadata from definitions, expressions, collocations, synonyms, hyperonyms, hyponyms, parts of speech, etc. The file format is .csv and contains metadata for the searched concepts (objects) in the considered software. The CSV file stores the characteristics of text and numeric data (a set of tokens) in tabular form. The linguistic corpus in Table 1. is organized and composed of several ontologies, providing a hierarchical structure of objects from the domain area (in this case words and phrases from indicators) and is a vocabulary in one with annotations for the accompanying linguistic characteristics (metadata) of the considered fragments [2]. It includes language data – over 3000 Phrase (words and phrases) for the limited subject area – Higher Education (HE), approximately 100 *IndicatorCode*, where these phrases occur, over 9000 of their language characteristics, such as *Synset, PartOfSpeech, Definition, SemClass, Core,* etc., as well as information

on how to calculate the weight of the indicator – *FType*. The linguistic expressions are classified so that after their successful identification and extraction of the searched values, it is possible to calculate the final result, which in some cases is a relative share of (proportion), percent of, in others the total number of, in the third – the average annual numbers.

Table3. Contents of the .csv file with data about the concepts with which the experiments were performed

Phrase	Synsets	IndCode	IndType	Value	FType
student	scholar | learner | candidate student	-		120	
graduate	alum | grad | bachelor | diplomate | Ph.D. | doctor | finish	-		94	
candidate-student for one announced place with the first desire for a speciality	candidates with the first desire for a speciality | candidates entered the speciality at the first request	4.1.	Quantity	65	0.541667
first wish	1st wish | most desired	4.1.	Quantity		
discipline	development | subject | course | plan			80	
disciplines, conducted in a foreign language	disciplines, teaches in a foreign language | disciplines, leads in a foreign language	4.2.	Quantity	12	0.15
students admitted with foreign diplomas	candidate students graduated abroad |	4.2.	Quantity		
speciality	profile	-		35	
specialities in which the diplomas of the graduates are certified in foreign higher schools	specialities in which students have certified diplomas	4.4.	Quantity	2	0.057143
students who participate in forms of practical training in the curriculum in a real environment	students who participated in internships | students involved in student internships |participated in internships in companies or organizations	4.5.	Quantity	20	0.166667
internships	student internship	-			
mobility	program	-			
outbound mobility		-			
incoming mobility		-			
incoming mobility of students lasting more than 1 month	incoming mobility over 1 month | mobility over a month |	4.6.	Quantity	6	0.05
duration	continuance | length | standing | endurance	-			
over	more than | at least	-			
students involved in outbound mobility for more than 1 month	trainees who attended programs in other countries for more than 1 month	4.7.	Quantity	4	0.033333
students with publications or creative performances	students, written research papers | students who wrote publications | students with performances	4.8.	Quantity	18	0.15
performance	creative expression	-			
publication	article | work | monograph | scientific work	-			
students who participated in scientific forums (conferences, round tables, seminars)	students who participated in scientific conferences	4.9.	Quantity	25	0.208333
scientific forum	conferences | round tables | seminars	-			
projects funded by the university	local project | university project	-			
projects funded by the university, with the participation of students	projects involving students	4.10.	Quantity	6	
students, relative to their total number, participated in projects		4.11.	Quantity	21	0.175
scientific project	research project	-			

In the language model shown in Table 1. in addition to concepts specific to the domain area, annotated with metadata, synonymous sets are included. In this way, the searched words / phrases from the string are attached to their basic representations (basic concepts from the corpus). The result is a table with columns corresponding to the metadata of the language model. The Pandas library has a function that converts data into a list, which converts data from fields into a list.

```
FILENAME = "Standart_4_phrases.csv"(1)
dataset = pd.read_csv(FILENAME, header=None)                    (2)
…
dataset['text'] = dataset['Phrase'] + ' ' + dataset['Synsets']  (3)
…
nltk.download('stopwords')                                      (4)
…
filenameUserText = sys.argv[1]                                  (5)
```

In the second step – **Step 2.** the read_csv () method reads the language file using certain parameters such as Phrase and Synonyms (Table 1.) to determine how to analyze the data. In (4) the corpus with the insignificant words – stopwords, is downloaded. The sys package provides variables and functions for manipulating different parts when executing Python code. The sys.argv (5) method returns a list of input arguments.

A function has been created in the program code that accepts the input user string as a string, returns a vector and generates a vector space where it finds the minimum distance to the vector. A method is defined that reads the linguistic resource, another that calculates the distance, and a third that returns the nearest neighbours, setting an acceptable proximity value. For this purpose, a model for extracting specific characteristics of the considered object is created and trained in advance. Programmatically, this model is enriched with additional filters and rules, keywords, etc. Only phrases encoded in the language model are used. Insignificant stopwords are excluded from reasoning.

```
def getVectorFromText(src_text_list):
    my_point = dimensions
for iEl in src_text_list:
        my_point[iEl] = 1
return my_point
```

In the next step **Step 3.**of the algorithm, a vector from the input user string is generated in the body of the getVectorFromText () method. Then there is the minimum distance to the vector from the user string.

```
def translate_func(src_text):
    translator = google_translator()
    translate_text = translator.translate(src_text, lang_src='bg',
lang_tgt='en')
return translate_text
```

The translate_func () method translates custom natural language query text into universal English words, as the SQL query is formed by tags that are attached to the corresponding natural language concepts.

The search index defined in **Step 4.** returns a list of indexes – indexesClosest of the nearest hits and their distance to the searched vector for query. The required distance is a positive number that indicates the distance of two vectors in the vector space. It is also called the Euclidean distance and is defined as the square root of the square of the differences in their coordinates.

In the body of the function vector_distance () in the last step **Step 5.** of the algorithm it is seen that in the iterating operation (for loop) the key-value pair is separated in order to be able to operate separately with the index and the value.

```
def vector_distance(v1, v2):
    summ = 0
for keyV1, val1 in v1.items():
        summ = summ + (v2[keyV1] - val1) * (v2[keyV1] - val1)
return float(math.sqrt(summ))
```

Sys.float_info.maxis a function in the sys embedded module that returns the maximum value that can be stored in a float variable.

```
min_dist = sys.float_info.max
distances = []
.......
if vector_distance(vector, user_request_vector) < min_dist:
        min_dist = vector_distance(vector, user_request_vector)
    distances[i] = vector_distance(vector, user_request_vector)

indexesClosest = []

for idxDist in range(distances):
if distances[idxDist] == min_dist:
        indexesClosest.append(idxDist)
.......
```

The algorithm used for the proximity of two vectors is the basis of the least squares method. Thus, minimizing the square Euclidean distance and the resulting monotone function of nonnegative values solves the problem of optimizing solutions (semantic searches).

4 Conclusion

This article presents the idea and software implementation of a semantic search tool that can be implemented independently in an existing information system and has the potential to improve its quality. A similar technique as in the shown algorithm, for semantic search and retrieval of information from the database, can be applied to other objects, as well as to be implemented in other areas of knowledge.

Efficiency can be achieved both by faster finding and retrieving the necessary information, and by a larger set of results in which the user can find the most relevant. The phase of vector space generation and semantic indexing of data could be started only with a new version of the input data, i.e. after updating the data or when starting with this data for the first time.

The evaluation of the quality of the obtained results will be the subject of additional research after more experiments are performed.

Acknowledgements

The paper is supported within the National Scientific Program “Young scientists and Post-doctoral students” in accordance with Appendix No. 11 of Council of Ministers Decision No. 577 of 17 August 2018.

References

[1] P. Siderov, Notes on algebra, Linear Algebra, ed. Vedi, Sofia, 2001.

[2] M. Zhekova, G. Totkov, Methodology of SQL queries generator to database set by natural language text, Scientific Works of the Union of Scientists in Bulgaria – Plovdiv. Series C. Technics and Technologies. Vol. XVIII, ISSN 1311-9419 (Print); ISSN 2534-9384 (Online), 2020, p. 98-101.

[3] F. Hasibi, K. Balog, D. Garigliotti, S. Zhang (2017). Nordlys: A Toolkit for Entity-Oriented and Semantic Search. Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, p. 1289-1292.

[4] S. K. Imminni et al., SPYSE – A Semantic Search Engine for Python Packages and Modules, 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C), 2016, p. 625-628.

[5] D. Gonzalez, A. Popovich, M. Mirakhorli (2016). TestEX: A Search Tool for Finding and Retrieving Example Unit Tests from Open Source Projects. 2016 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), p. 153-159.

[6] H. Dar, M. Lali, M. Din, K. Malik, S. Bukhari, Frameworks for Querying Databases Using Natural Language: A Literature Review.ArXiv, abs/1909.01822, 2019