A Synergistic Framework for Geographic Question Answering

[1,2]Wei Chen, [1]Eric Fosler-Lussier, [2]Ningchuan Xiao, [1]Satyajeet Raje, [1]Rajiv Ramnath, [2]Daniel Sui
[1]Dept. of Computer Science and Engineering, [2]Dept. of Geography
The Ohio State University
Columbus, OH USA
Email: chen.1381@osu.edu

***Abstract*—QA (question answering) systems designated for answering in-depth geographic questions are highly demanded but not quite available. Previous research has visited various individual aspects of a QA system but few synergistic frameworks have been proposed. This paper investigates the nature of geographic question formation and observes their unique linguistic structures that can be semantically translated into a spatial query. We create a new task of solving non-trivial questions using GIS (Geographic Information System) and test it with an associated corpus. A dynamic programming algorithm is developed for classification and voting algorithm for verification. Two types of ontologies are integrated for disambiguating and discriminating spatial terms. PostGIS serves as the GIS backend to provide domain expertise for spatial reasoning.**

Results show that exact answers can be returned quickly and correctly by our system. Contrast classification results in improved accuracy compared with the baseline which proves the effectiveness of proposed methods.

Keywords: nlp, gis, question answering, ontology, voting algorithm, machine learning, dynamic programming, spatial SQL

I. INTRODUCTION

A. Natural language processing and question answering

Natural language processing (NLP) is concerned with both syntactic and semantic analysis for building a question answering (QA) system. Part-of-speech (POS) tagging often serves as the first step of syntactic analysis of a sentence. Contemporary POS taggers can reach an average accuracy of above 95% on tokens but only 56% on sentences [1]. Low accuracy is often caused by differences between the training and test sets on topics and styles, as well as training size and type of taggers [1]. Efforts have been made in tagging algorithms to improve both performance and accuracy including dynamic programming-based Viterbi, rule-based Brill tagger, maximum entropy tagger and neural network-based tagger. For a complete list of tagger comparison, one may take a look at the ACL wiki page on POS tagging [2].

Parsing is concerned with the semantic aspects of analyzing sentence structures and serves as a pre-step for information retrieval. Often used parsers are chunk parser, full parser and dependency parser. A chunk parser offers a shallow parsing to a sentence and usually targets on structures like noun and prepositional phrases. Each phrase is then called a chunk. Unlike chunk parsers which only annotate part of a sentence, a full parser aims to generate a complete parse tree out of a sentence. However, a full parser can be slow and ambiguous as it is characterized by a recursive procedure and a parse tree may not be unique [3]. A dependency parser is focused on building relations between words rather than positioning them in a tree structure like previous two parsers. Fig. 1 is an example of parsing the same sentence "major cities in Columbus" using three different parsers.

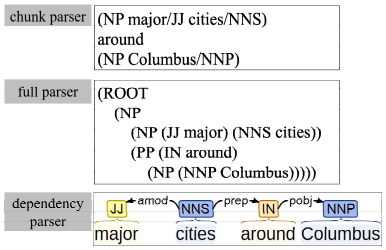

Figure 1. Three types of parsing of a same sentence.

B. Machine learning and question answering

In QA related literature, machine learning is used to classify questions into answerable types [4, 5], locate on-target answers [6] as well as acquire knowledge [7, 8]. Various machine learning algorithms have been developed for automatic question classification among which nearest neighbors, naïve bayes, decision tree, sparse network of winnows, support vector machines are proven to be effective on the Trec QA dataset [4]. However, the Trec QA dataset only includes less than 5% geographic questions among which most are also quite generic. Therefore, it is necessary to create a corpus with more in-depth and often closed domain questions.

Machine learning features employed in previous research are mostly surface text features which are essentially bag-of-words extracted through an unsupervised information retrieval process[4]. Such feature selection methods may be effective in an open domain environment but not in a closed domain one because in closed domains lexical-based features are not as discriminative.

C. Ontological modeling and question answering

Philosophically, ontologies are basic categories and relations of being or conceptions of reality[9]. Computationally, ontologies are concerned with the use of computational models to define these categories and relations. Computational ontologies are often coded in RDFs (resource description framework) and OWLs (ontology web language)

978-0-7695-5119-7/19 $26.00 © 5119 IEEE
DOI 10.1109/ICSC.2013.25

for programmable access[10, 11]. SPARQL is de facto standard for querying RDF encoded ontology database [12].

Use of ontologies is often to attack knowledge intensive problems[13, 14]. Geographic question is knowledge intensive in the sense that it involves knowledge of not only spatial entities and relations but also reasoning with the two [15, 16]. Spatial entities can be concrete objects like cities, places as well as abstract entities like points, lines and polygons. Spatial relations include topological relations such as (dis)joint, overlap and intersect as well as metric relations such as the percentage of overlap.

D. Spatial reasoning and geographic question answering

Recent work on geographic information retrieval (GIR) has been focused on questions of ontologies[17, 18], text summarization[19] and question answering[20, 21]. However, emphasis have been almost entirely put on knowledge acquisition rather than knowledge engineering, the latter as an often requirement for building more sophisticated QA system.

Knowledge engineering should result in generation of new knowledge. The spatial thinking and reasoning capabilities of humans [22, 23] translates a spatial problem into an analytical form, and often involves the creation of new objects as necessary intermediate step to solve the big problem [24]. We argue that only through a synergistic framework can one build a system that mimic human intelligence of geographic problem solving.

E. Objectives of this research

The objective of this research is to propose a synergistic framework for building a geographic QA system. Previous research has investigated aforementioned four aspects in an relatively independent fashion. A holistic view is not quite available but highly needed. IBM's deep learning project Watson is one of the few examples that propose synergistic frameworks for solving open domain questions [25]. However, to answer in-depth questions in a closed domain Watson may be challenged because it lacks domain expertise and designated procedure to solve these questions. Therefore, we find synergistic frameworks highly relevant in this issue.

This paper integrates methods and techniques from machine learning, NLP, ontological modeling and GIS to solve geographic QA problems. We create our own corpus from human survey in order to get relatively more complicated structures. We also propose tractable approaches for classification and verification. Finally, we test the effectiveness of our methods and the performance of our system using standard machine learning evaluation measures.

The remaining of the paper is organized as follows. Section II talks about data. Section III introduces our synergistic framework. Section IV to VII discuss each component in the framework and some experimental results. Section VIII are results and evaluation. Section IX concludes.

II. DATA AND QUESTION SAMPLES

A. Human survey corpus and sptial data

Our experiment data include sample questions from human survey, spatial geometric data, and census attribute data. Survey question data are collected from 50 undergraduate students at the Ohio State University. Each survey subject is given a map with layered information along with 5 example questions. Subjects are then asked to raise another 20 questions and use ArcGIS (a desktop GIS software) to solve them. Their efforts are timed and answers are recorded in required format. As a requirement, only wh- questions are allowed and only one exact answer should be given.

B. Corpus statistics

Corpus questions are summarized in Table I. We obtain about 800 questions from which we generalize 5 major categories that have the most number of questions. These five categories are location question (A), relative location question (B), distance question (C), proximity entity question (D), and proximity buffer question (E). We later use type A, B, C, D, E to denote these five categories. Questions are classified based on the types of answer and operations needed for solving them.

In Table I, [] denotes an array of objects, () denotes a value, * denotes zero or more values of the same pattern. For example, for location question an answer should be a latitude and longitude pair. If locations of multiple cities are asked, comma delimited pairs should be returned. Answer formats for other questions are also included in the table.

Finally, we end up with about 500 questions. We find that although our subjects have certain level of proficiency in GIS and are given examples to help raise questions they are still resistant to tackle more complicated questions (type B, D, E).

For testing purpose, we also include 100 trick questions that mimic the length and structure of a geographic question but not as sensible such as "What is the name of Michael Jordan" (versus "What is the location of Columbus Ohio").

Table I. CORPUS QUESTIONS

Type	QA					
	Example question	***Answer format***	***Example answer***	***Minimum GIS steps***	***Avg. time (mins)***	***% of questions***
A	Where is Columbus?	[coordinates] (double double)*	39.964491 - 82.999191	Enable the toolbar; Make the queried layer selectable; Navigate to or search queried object in attribute table; Click to identify.	1	32%
B	Where is Columbus perspective to Cleveland?	[distance unit direction] (double string string)*	150 miles southwest	Find coordinates of two objects; Write a script to calculate the north azimuth bearing; Use the measure tool to calculate distance.	2	13%
C	How far is Columbus from Cleveland?	[distance unit] (double string)	150 miles	Find coordinates of two objects; Use the measure tool to calculate distance.	1.5	28%
D	Which city is the nearest to Columbus?	[name] (string)	Grandview Heights	Use the near tool to find the distance between all pairs of inputs; Sort the output by distance; Find the destination with the minimum near distance to the origin.	2	12%
E	What cities are within 5 miles from Columbus?	[name] (string)*	Grandview Heights, Bexley	Select origin in the attribute table; Create a buffer based on the radius from the origin; Make a spatial selection of queried layer using the buffer.	2	15%

III. QUESTION ANSWERING FRAMEWORK

Fig. 2 is our proposed framework for building a geographic question answering system. Four integrated components are the NLP component (linguistic model), ML component (learning model), ontology component (knowledge model) and GIS component (spatial model). The arrows indicate shared data flow and relationships between integrated components.

The NLP component examines syntactic and lexical patterns in a sentence, and parses out useful constituent structures for subsequent information retrieval tasks.

The ML component trains the classifier using features extracted from NLP process and classifies new sentences. The ML component should also implement a learning strategy for tuning classification parameters based on new input questions.

The ontology component defines both domain ontologies of spatial terms in a sentence and operational ontologies of spatial operations in a GIS. Through ontological modeling, the system should be able to disambiguate the meaning of spatial terms and correspond them to spatial operations in a GIS.

Finally, the GIS component constructs formal queries using operands, operators and constraints extracted from previous steps and retrieves results from a spatial database.

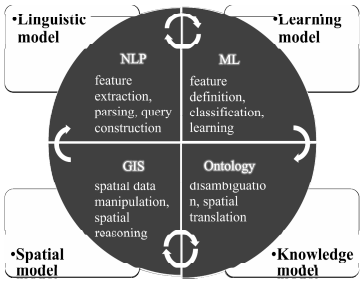

Figure 2. Synergistic framework of a geographic QA system

IV. LINGUISTIC MODEL

A. Lexical analysis and feature extraction

The linguistic model is mainly concerned with syntactic analysis of sentences and extracting features from parsed results. We integrate Java-based Stanford part-of-speech tagger and python-based NLTK library to implement our NLP algorithms. To do this, we write an interface in Python to reference functions in Stanford tagger's jar file.

First, we use frequency analysis to extract features from already classified sentences in the training set. Such features could be lexicals, lexical lemma and lexical stems. For example, for the location question we find most frequently used tokens are "where, location, located, coordinates" (see Table III). Therefore, these terms should be used as features for training a classifier. Using similar methods, we distill most frequent features in each category. Also, for experimental purpose we only implement a unigram model.

B. Template generation

Our program tags each sentence in the training set and generalize their part-of-speech patterns. Through generalization, we merge sentences with the same tag sequences and record such tag sequence. We implement a template matching algorithm for classification because we observe that question classification information is largely encoded to a great degree in tag sequences. Finally, we end up with much fewer templates than the original raw questions (see Table II). We find complicated questions are more likely to result in more templates which conforms to our intuition.

Table II. TEMPLATES GENERATION

Type	# Question	# Templates	Compression Rate
A	160	28	14%
B	65	9	18%
C	140	24	13%
D	60	30	28%
E	75	15	25%

C. Chunk parsing for information retrieval

As discussed previously, a chunk parser can be used to retrieve information which can later be used to construct a SQL query. Here, we define two regular expression-based chunkers using Python syntax: input and output chunkers respectively (see Fig. 3). All tags are standard tags defined in Penn Treebank.

```
cp_input = RegexpParser('''
   input:
         {<IN><CD>*<CC>*<CD>+<NN.*>+<IN>*<TO>*<DT>*<NN>*<IN>*<NNP.*>+<CC><CD>*<CC>*<CD>+<NN.*>+<IN>*<TO>*<DT>*<NN>*<IN>*<NNP.*>+}
         {<IN><CD>*<CC>*<CD>+<NN.*>+<IN>*<TO>*<DT>*<NN>*<IN>*<NNP.*>+<CC><DT>*<NN>*<IN>*<NNP.*>+}
         {<IN><CD>*<CC>*<CD>+<NN.*>+<IN>*<TO>*<DT>*<NN>*<IN>*<NNP.*>+}
         {<IN><DT>*<NN>*<IN>*<NNP.*>+}
         {<DT>*<NN>*<IN>*<NNP.*>+}    ''')

cp_output = RegexpParser('''
   output:
         {<CD>*<CC>*<CD>*<IN>*<RB.*><JJ.*>*<NN.*>+(<IN>|<VBP>)}    ''')
```

Examples

- (… cities) within one hundred and one miles from Columbus and 5 kilometers to the city of Dayton
- (…cities) inside 25 miles from Columbus and Dayton
- (…cities) in 25 miles from Columbus
- (the location) of Columbus
- (where is) Grandview Heights
- one hundred and five nearest very small villages to (Columbus)

Figure 3. Input and output chunkers and example sentences

Once input and output chunks are extracted, the QA system will start looking for more granulated structures to locate operators, operands and constraints associated with each chunk. For this, we also implement several secondary chunkers: number chunker, constraint chunker, operand chunker and operator chunker.

Fig. 4 illustrates how the spatial role of each token is labelled based on input and output chunks extracted from the sentence "5 nearest major cities within 20 miles of Columbus". These roles are domain expertise knowledge ideally provided by GIS experts.

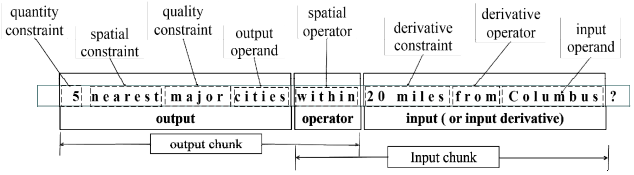

Figure 4. Spatial role labelling in chunks

V. MACHINE LEARNING MODEL

A. Longest common sub tag sequence through dynamic programming

Here, we propose a simple but tractable classifier based on longest common sequence algorithm in dynamic programming. Our approach is able to iteratively train our classifiers and classify with a reasonable computational cost.

Our algorithm tries to match the tag sequence of a new sentence with existing templates using dynamic programming.

The details of dynamic programming version of finding the longest common subsequences (LCS) between two or more input strings can be found in [26]. Here, we provide the basic algorithm structure that we implement. In Fig. 5, x and y are two tag sequences in which we define the stepwise result c[i,j]=LCS(x[1..i],y[1..j]) and the final result is c[m,n]=LCS(x[1..m],y[1..n]). The peudo code in Fig. 6 illustrate our implemented memoization version of the algorithm to boost performance.

$$c[i,j] = \begin{cases} 0 & if\ i = 0\ or\ j = 0 \\ c[i-1, j-1] + 1 & if\ x[i] = y[j] \\ \max\{c[i-1, j], c[i, j-1]\} & otherwise \end{cases}$$

Figure 5. LCS algorithm

Input: reference tag sequence x and new tag sequence y.
Output: length of the LSC c[m,n]

```
1.  LCS (x,y)
2.     m=x.length
3.     n=y.length
4.     let c[0..m, 0..n] be a new table
5.     for i = 0 to m
6.        c[i, 0]=0
7.     for j = 0 to m
8.        c[0, j]=0
9.     for i = 1 to m
10.       for j = 1 to n
11.          if x_i==y_j
12.             c[i,j]= c[i-1,j-1]+1
13.          elseif c[i-1,j]≥ c[i,j-1]
14.             c[i,j]= c[i-1,j]
15.          else
16.             c[i,j]= c[i,j-1]
17.    return c[m,n]
```

memoization

Figure 6. LCS pseudo code

Further, a similarity measure is calculated based on the length of the LCS and that of the template. The similarity between the new sequence (NS) and the i^{th} reference sequence (RS) is defined, as S_i in (1):

$$S_i = LCS\ (RS_i, NS) / length(RS) \qquad (1)$$

The reason the denominator is about RS not NS is because RS is the gold. Further, the maximum similarity is defined, as S_{max} in (2):

$$S_{max} = Max\{Si : i=1,2,\ldots,n\} \qquad (2)$$

In case of a tie among several templates $\{RS_j : j \in [1,\ldots,n]\}$, all these templates are saved for further verification. Given our data, we find 0.5 to be a good similarity threshold for classification.

B. Voting algorithm for classification verification

The classic voting algorithm (also known as Maekawa's algorithm) is introduced in distribution system application to ensure mutual exclusion of concurrent processes [27]. Here, we borrow the idea from Maekawa's and develop a new method to verify previously discussed LCS classification results. We define a feature set to mimic the quorum set in Maekawa's, feature votes to mimic grant messages in Maekawa's and quorum to mimic minimum number of permissions in Maekawa's.

We find voting algorithm relevant because the nature of question classification is also a concurrent process by nature. For example, for location question, words like "where, location, located" are all indicators to question type A given certain tag sequence. Therefore, a program doesn't need to wait to make a decision until all evidences are found. Also, for relative location question, a word like "direction" imposes the same effect as a combination of two words "relative location". Therefore, different features should implement different votes which contribute to quorums for classification (Table III).

In the table, { }denotes a feature set. All features that contribute to the quorum should come from the same set because they are related features. [ ] denotes a feature subset which is a sub collection of semantically related features. ( ) indicates the number of votes of certain feature. Once a feature in a subset is counted, additional features in the same subset will not be double dipped.

When the voting algorithm executes, the program will first find the category of the matched template, and then create threads to verify the classification in parallel by checking whether features in that category satisfy the quorum. If accepted, we may conclude the sentence belongs to that category and later use corresponding SQL template for query.

Table III. VOTING ALGORITHM

Type	Voting Configuration	
	Feature (Votes)	***Quorum***
A	{[location, where, coordinates, located,…](1)}	1
B	{[where, location](1), [relative](1)} {[direction](2)}	2
C	{[distance, how far,…](1)}	1
D	[{[city, street, county,…](1), nearest, closet, furthest,..](1)}	2
E	{[city, street, county,…](1), [within, inside,…](1), [mile, km, meter,…](1)}	3

VI. KNOWLEDGE MODEL

A. Ontologies, DBpedia and GeoNames

Ontologies serves as the fountain of knowledge for building intelligent computer programs. One commonly used online ontological database is DBpedia. DBpedia provides a crowd-sourced community platform for extracting various kinds of structured information in Wikipedia.

GeoNames is another gigantic online knowledge database, populated by geographic entities in particular. It includes over 8.3 million toponyms, and is encoded in RDFs and queryable through SPARQL. We use GeoNames ontologies to disambiguate named entities in a sentence. For example, Columbus may be a city in Ohio, a city in Georgia and a person's last name in different circumstances

B. Operational ontologies

Ontologies from DBpedia and GeoNames provide semantic information about entities but not operations. It features knowledge of beings but not knowledge of logic. Therefore, an array of operational ontologies are needed. Spatial operational ontologies in particular should result in specifications about spatial operators and operands in a spatial database.

Operand specification should include definitions of entity and relation classes. For example, Columbus is a *City* type entity and *City* is a class. Ohio is a *State* type entity and *State* is another class. Therefore, if a *City* is defined as spatially contained in a *State*, then Columbus can be logically determined to be spatially contained in Ohio.

Operand specification should include database schemes which tells a QA system where to look for records. For example, it should define the mapping from entity types to either column name of a table or value under a column.

Operator specification should incorporate classes about spatial relation definitions and relational definitions between these relations. For example, both spatial terms "within" and "inside" should be mapped to the higher level relational class called "in".

As part of a larger project, we are going to formalize all these operational ontologies in RDFs. But for this paper, we only implement some of them for experiment purposes.

VII. SPATIAL MODEL

A. Spatial query templates

To answer geographic questions using spatial SQL, we create SQL templates for each category and fill the templates using parsed data. These templates are created based on typical GIS analytical routine for solving corresponding questions and are provided by GIS experts.

Fig. 7 is an example SQL result for proximity buffer (type E) query with missing values in the template already populated. We end up with five categories of templates corresponding to five types of questions. Through a templative generation process, we construct each specific query.

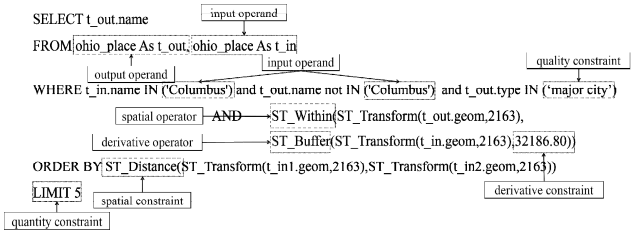

Figure 7. Spatial SQL for a proximity buffer (type E) question

VIII. RESULES AND EVALUATION

Our corpus is split into 70% and 30% for training and testing respectively. Trick questions are only used in the testing set. Table IV and Table V are the results.

Table IV. CLASSIFICATION ACCURACY: PRECISION, RECALL, F VALUE

Classification Accuracy					
Type	**A**	**B**	**C**	**D**	**E**
P-base	0.58	0.56	0.57	0.52	0.50
P-contrast	0.96	0.93	0.94	0.87	0.83
R-base and contrast (similarity>0.5)	0.91	0.88	0.89	0.82	0.80
F1-base	0.71	0.69	0.70	0.64	0.61
F1-contrast	0.93	0.90	0.91	0.84	0.81

Table V. ANSWER GENERATION ACCURACY AND RESPONSE TIME

Type	**A**	**B**	**C**	**D**	**E**
Accuracy	0.96	0.92	0.98	0.88	0.86
Avg. Resp. Time	1.2s	2.2s	1.4s	2.4s	2.6s

Classification verificatin affects precision but not recall therefore we observe a higher precision for contrast cases than the baseline. This is because, without verficiation, trick questions are incorrectly assigned to one of the five classes purely based on their tag patterns.

It is also seen that classificatoin accuracy is higher for A, B, C type of question but lower for other two types. We think this is because the feature structures for the first three classes are relatively simple.

Classification errors also occur when prepositional phrases are used at the beginning of a sentence such as "Relative to Columbus, where is Dayton". Some of these sentences only occur in our test set but not included in our training set.

Moreover, category D and E are similar in lexical structure and can sometimes confuse our system. In a few cases, type D questions can't be correctly classified if a required feature such as "nearest" is missing. In this case, humans may think the return should be a random result but our system gives none.

Overall, our system generates correct answers at an average of accuracy over 90%. There are a few cases that fail our query. First is unkown quality constraints such as "capital (city)". Our system is not as general as encoding knowledge such as *what is a capital* in our database. Second is the structure of *that* cluase which is another more complicated strucutre we may consider in future work. Last is case errors. For example "Columbus" misspelled as "columbus" may fail the system because the case is an indicator for a proper noun.

Performancewise, our system can generate answers normally within 3 seconds given the scale of our dataset. The machine that runs the test has an i5 dual core Intel CPU with 4GB ram.

IX. CONCLUSIONS

Our proposed synergistic framework is proven to be viable and effective for answering non-trivial natural language geographic questions. Our dynamic programming-based template matching classifier can classify a sentence in polynomial time based on tag sequences. A voting algorithm-based verifier checks classification results using semantics in the features. A combination of the two produce a high accuracy for classification.

Evidences are provided that GIS is an indispensable component to answer in-depth geographic questions. It offers exclusive spatial reasoning capabilities through spatial SQL which can encode human expertise of spatial problem solving. With embedded intelligent GIS routine, our system can quickly give exact answers to a large portion of the questions in selected categories.

ACKNOWLEDGMENT

The authors would like to thank Dr. Chris Brew, Dr. Nan Deng and other anonymous reviewers for their suggestions and comments to improve the quality of the paper.

REFERENCES

[1] C. D. Manning, "Part-of-speech tagging from 97% to 100%: is it time for some linguistics?," in *Computational Linguistics and Intelligent Text Processing*, ed: Springer, 2011, pp. 171-189.

[2] *POS Tagging (State of the art)*. Available: http://aclweb.org/aclwiki/index.php?title=POS_Tagging_(State_of_the_art)

[3] B. Katz and J. Lin, "Selectively using relations to improve precision in question answering," in *Proceedings of the workshop on Natural Language Processing for Question Answering (EACL 2003)*, 2003, pp. 43-50.

[4] D. Zhang and W. S. Lee, "Question classification using support vector machines," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, 2003, pp. 26-32.

[5] X. Li and D. Roth, "Learning question classifiers," in *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, 2002, pp. 1-7.

[6] M. Levit, E. Boschee, and M. Freedman, "Selecting On-Topic Sentences from Natural Language Corpora," *Interspeech 2007: 8th Annual Conference of the International Speech Communication Association, Vols 1-4,* pp. 857-860, 2007.

[7] X. Chang and Q. H. Zheng, "Offline definition extraction using machine learning for knowledge-oriented question answering," *Advanced Intelligent Computing Theories and Applications,* vol. 2, pp. 1286-1294, 2007.

[8] H. J. Oh and B. H. Yun, "Sentence topics based knowledge acquisition for question answering," *Ieice Transactions on Information and Systems,* vol. E91d, pp. 969-975, Apr 2008.

[9] T. Gruber. (2008). *What is an Ontology*. Available: http://www-ksl.stanford.edu/kst/what-is-an-ontology.html

[10] G. Antoniou and F. Van Harmelen, "Web ontology language: Owl," in *Handbook on ontologies*, ed: Springer, 2009, pp. 91-110.

[11] D. Beckett and B. McBride, "RDF/XML syntax specification (revised)," *W3C recommendation,* vol. 10, 2004.

[12] E. Prud'Hommeaux and A. Seaborne, "SPARQL query language for RDF," *W3C recommendation,* vol. 15, 2008.

[13] N. Guarino, "Understanding, building and using ontologies," *International Journal of Human-Computer Studies,* vol. 46, pp. 293-310, 1997.

[14] W. N. Borst, *Construction of engineering ontologies for knowledge sharing and reuse*: Universiteit Twente, 1997.

[15] H. J. Miller and J. Han, *Geographic data mining and knowledge discovery*: CRC Press, 2003.

[16] B. Smith and D. M. Mark, "Ontology and geographic kinds," 1998.

[17] C. Jones, H. Alani, and D. Tudhope, "Geographical information retrieval with ontologies of place," *Spatial Information Theory,* pp. 322-335, 2001.

[18] C. Jones, A. Abdelmoty, and G. Fu, "Maintaining ontologies for geographical information retrieval on the web," *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE,* pp. 934-951, 2003.

[19] J. M. Perea-Ortega, E. Lloret, L. A. Urena-Lopez, and M. Palomar, "Application of Text Summarization techniques to the Geographical Information Retrieval task," *Expert Systems with Applications,* vol. 40, pp. 2966-2974, Jun 15 2013.

[20] D. Ferres, A. Ageno, and H. Rodriguez, "The GeoTALP-IR system at GeoCLEF 2005: Experiments using a QA-based IR system, linguistic analysis, and a geographical thesaurus," *Accessing Multilingual Information Repositories,* vol. 4022, pp. 947-955, 2006.

[21] D. Santos, N. Cardoso, P. Carvalho, I. Dornescu, S. Hartrumpf, J. Leveling*, et al.*, "GikiP at GeoCLEF 2008: Joining GIR and QA Forces for Querying Wikipedia," *Evaluating Systems for Multilingual and Multimodal Information Access,* vol. 5706, pp. 894-905, 2009.

[22] A. U. Frank, "Qualitative spatial reasoning: Cardinal directions as an example," *International Journal of Geographical Information Science,* vol. 10, pp. 269-290, 1996.

[23] D. M. Mark, C. Freksa, S. C. Hirtle, R. Lloyd, and B. Tversky, "Cognitive models of geographical space," *International Journal of Geographical Information Science,* vol. 13, pp. 747-774, 1999.

[24] A. U. Frank, "Qualitative spatial reasoning about distances and directions in geographic space," *Journal of Visual Languages & Computing,* vol. 3, pp. 343-371, 1992.

[25] S. Keates, P. Varker, and F. Spowart, "Human-machine design considerations in advanced machine-learning systems," *Ibm Journal of Research and Development,* vol. 55, Sep-Oct 2011.

[26] D. S. Hirschberg, "Algorithms for the longest common subsequence problem," *Journal of the ACM (JACM),* vol. 24, pp. 664-675, 1977.

[27] M. Maekawa, "An algorithm for mutual exclusion in decentralized systems," *ACM Transactions on Computer Systems (TOCS),* vol. 3, pp. 145-159, 1985.