A Synergistic Framework for Geographic Question Answering

Wei Chen
Dept. of Computer Science and Engineering
Dept. of Geography
The Ohio State University
Columbus, OH USA
chen.1381@osu.edu

Authors Name/s per 2nd Affiliation *(Author)*
line 1 (of *Affiliation*): dept. name of organization
line 2: name of organization, acronyms acceptable
line 3: City, Country
line 4: e-mail address if desired

***Abstract*—QA systems designated for answering in-depth geographic questions are of high demand but not quite available. Previous research has visited various individual aspects of question answering but few synergistic frameworks have been proposed. This paper investigates the nature of the formation of geographic questions and observes their unique linguistic structures that can be translated into semantic roles in a spatial SQL. We create a new task of solving advanced questions using GIS and test it with an associated corpus. A dynamic programming based algorithm for classification and voting algorithm for verification are also included. Two types of ontologies are incorporated for disambiguating and discriminating of spatial terms. PostGIS is used as the GIS system to provide domain expertise of spatial reasoning.**

Results show that exact answers can be correctly provided by our demonstration system normally within 3 seconds. Our contrast results for classification are significantly more accurate than the base line and the answer accuracy is also high given the test data.

Keywords-component; nlp; gis; question answering; ontology; voting algorithm; machine learning; dynamic programming; spatial SQL

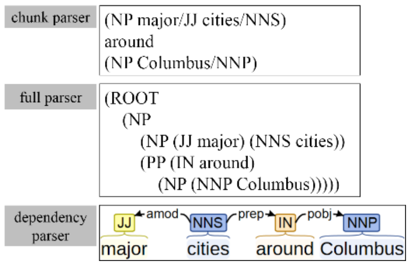
I. Introduction

A. *Natural language processing and question answering*

Natural language processing (NLP) is more relevant than any other key words in the literature on question answering (QA). NLP is concerned with both syntactic and semantic analysis of sentences. Recent research on the syntactic aspect of it includes research efforts on constrained grammar [1], sentence parsing [2, 3] and distributional syntax [4]. However, few efforts have been focusing on the syntax for geographic question answering which is different from the mainstream because certain grammar in geographic question has special functions such as prepositions.

The first step of processing in syntactic analysis is often part-of-speech (POS) tagging. POS tagging offers a shallow linguistic analysis of sentences but lays important foundations for subsequent higher order analysis. Contemporary POS taggers can reach an average accuracy of above 95% on tokens but only 56% on sentences [5]. 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 [5]. 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 [6].

Parsing analyzes constituent structure of a sentence and is often used for information retrieval. Several type of parsers are available in the literature: chunk parser, full parser and dependency parser. A chunk parser offers a shallow or partial parsing to a sentence and usually extracts linguistic structures like noun phrases and prepositional phrases. Each phrase is then called a chunk. Unlike chunk parser which only annotates part of a sentence, a full parser will generate a complete parse tree out of a sentence. The problem is that a full parser can be slower and ambiguous [7] as it carries out reclusive procedures and a parsing tree may not be unique. A dependency parser aims to build relations between words instead of positioning them in a tree structure. Fig. 1 is an example of parsing the same sentence "major cities in Co" using three different parsers.



```
(NP major/JJ cities/NNS)
around
(NP Columbus/NNP)
```

```
(ROOT
   (NP
      (NP (JJ major) (NNS cities))
      (PP (IN around)
         (NP (NNP Columbus)))))
```

Figure 1. Three parsing of a same sentence

While traditional semantic analysis in NLP relies on word sense libraries such as WordNet to discriminate and disambiguate words, recent research looks more into the use of ontologies. Semantic role labeling is a related task of assigning a semantic role to a constituent in a sentence. Concretely, these roles are labels that annotate the function of a constituent. For answering geographic questions, we argue that roles could be defined as operands, operators and constraints which are building blocks of constructing a SQL.

In relation to machine learning, NLP extracts lexical and syntactic patterns from sentences into feature arrays. These feature arrays are then used to train a classifier either using supervised or unsupervised methods. Feature are extracted during a test process too. Finally, NLP is also involved in building knowledge base a question answering system. Knowledge primitives can usually be obtained from online sources such as Wikipedia and DBpedia. This research will introduce an alternative way of creating new knowledge through spatial reasoning.

B. *Machine learning and question answering*

In the literature, machine learning is used to classify questions into answerable types [8, 9], locating on-target answers [10] as well as acquiring knowledge [11, 12]. Various machine learning algorithms have been experimented for automatic question classification among which nearest neighbors, naïve bayes, decision tree, sparse network of winnows, support vector machines are the most experimented and are proven to be effective methods on the Trec QA dataset [8]. However, the Trec QA dataset only includes less than 5% geographic questions among which many are rather basic than in-depth. Therefore, it is necessary to construct a corpus of more in-depth geographic questions.

Machine learning features employed in previous research are mostly surface text features which are essentially continuous word sequences, preferably named bag-of-words[8]. We argue that such unsupervised feature selection process may be more effective to an open domain environment than a closed domain one as lexical patterns in open domains are usually more discriminative. Therefore, granulated classification scheme is needed as questions in a closed domain more often consist of many similar lexical structure that yet has dramatic different meanings.

C. *Ontological modeling and question answering*

Ontologies, from a philosophical perspective, can be defined as basic categories and relations of being or conceptions of reality[13]. Computational ontologies resort to the use of computational models to define these categories and relations normally on a domain specific basis. Computational ontologies are often coded in RDFs (resource description framework) and OWLs (ontology web language) for programmable access[14, 15]. SPARQL is de facto standard for querying RDF encoded ontology database [16].

Use of ontologies is often to attack knowledge intensive problems[17, 18]. Geographic question is knowledge intensive in the sense that it involves knowledge of both spatial entities and spatial relations[19, 20]. Spatial entities include concrete entities like cities, places and abstract entities like point, lines and polygons. Spatial relations include topological relations such as if joint, overlap and intersection as well as metric relations such as the percentage of overlap.

Ontology is also discussed in the literature in relation to the application of a geographic information system [21] however it hasn't yet been studied in a joint context with natural language processing and machine learning which are both necessary components for implementing any practical QA systems.

D. *Spatial reasoning and geographic question answering*

To answer a geographic question, we first need to retrieve geographic information. Recent work on geographic information retrieval (GIR) has been focused on the use of ontologies[22, 23], text summarization[24] as well as question answering[25, 26]. However, emphasis have been almost entirely put on knowledge acquisition instead of knowledge engineering. While the state of art in knowledge acquisition may possibly help generate abundance of world knowledge in a short period of time knowledge engineering is an inevitable way to go if human expertise is a requirement.

Knowledge engineering for geographic question answering should also result in the generation of new knowledge. The way human solve problem is not merely based on what we know but also what we can infer. The cognitive reason that humans are adaptive is because of our faculty in spatial thinking and spatial reasoning [27, 28] which translate a spatial problem into an analytical formulation in order to solve it quantitatively [29].

We observe that some geographic questions cannot be answered without spatial reasoning. This is because knowledge primitives such as direct location information is not enough to answer to a complicated question which often requires mental creation of derivatives from primitives. For example, to solve a proximity buffer question requires first construct an extended area called a buffer and then find queried entities within that buffer.

E. *Objective of this research*

The objective of this research is to propose a synergistic framework for building a geographic question answering system. Previous research has investigated question answering from various individual aspect discussed above but a holistic picture is not available especially in the geographic domain.

Therefore, we resort to methods and techniques in machine learning, NLP, ontological modeling and GIS to build a demonstration system. We create our own corpus from human survey which involves some complicated questions and corresponding answers that can only be generated with spatial expertise. We also propose some tractable new methods for machine learning-based classification. 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 survey data and processing. Section III introduces our synergistic framework. Section IV to VII discuss each component in the framework and experiment results. Section VIII is a summary of overall results and evaluation. Section IX are conclusions.

II. DATA AND QUESTION SAMPLES

A. *Human survey corpus, sptial data*

Our experiment data include sample questions from human survey, spatial geometric data, and census attribute data. Survey question data are collected from about 50 undergraduate students in a cartography class at the Ohio State University. To inspire their thinking, we offer each subject a map (Fig. 2) and provide them with 5 example questions along with correct answers to these questions. Subjects are then asked to raise another 20 similar or different questions and use ArcGIS (a desktop GIS software) to solve them. The process they solve each question is timed and answers are recorded. As a requirement, only wh- questions are allowed and an exact answer should be given.

As we want to experiment on how different geometric representations affect question asking and answering, we make a map with three different layers corresponding to three geometric types in a GIS: a point city layer, a linear street layer and a polygon county boundary layer (Fig. 2). An attribute table is also provided to subjects along with the map so that they can choose the correct names of cities, streets and counties in our database when raising a question.

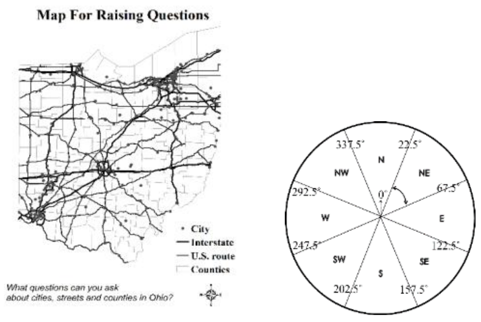

Map For Raising Questions
City
Interstate
U.S. route
Counties
What questions can you ask about cities, streets and counties in Ohio?
337.5°
22.5°
N
NW
NE
0°
292.5°
67.5°
W
E
247.5°
122.5°
SW
SE
S
202.5°
157.5°

Figure 2. Map for raising questions and azimuth definition

B. *Corpus statistics*

Subjects' questions are summarized in Table I. Answer format, example answer, GIS steps, average solution time and distribution of questions are also included in the table. A subject can choose to only raise a solvable question but not solve it if he/she thinks the solving procedure is too difficult for them. The time is then not counted for that question

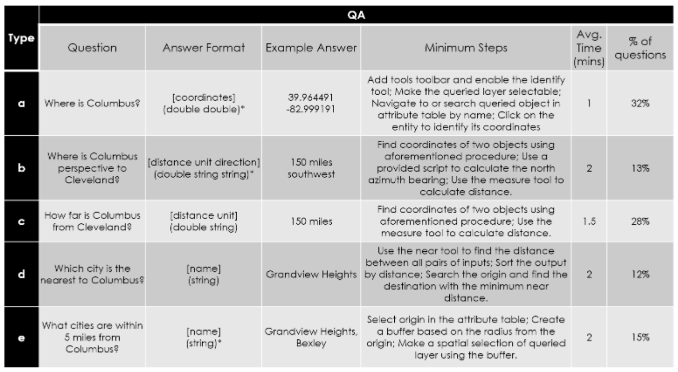
We obtain about 800 questions in total from which we generalize 5 major categories with the most number of questions. These five categories are location question (a), relative location question (b), distance question (c), proximity entity question (d), and proximity buffer question (e). Later, we use a,b,c,e,d to denote these five types. Questions are categorized based on the answer type and types of operations needed for solving them (Table I). Due to space limitations, we only list one example question for each category.

We observe a variety of ways from the subjects to ask a same question and subjects are less likely to raise complicated questions. The time is recorded entirely on problem solving and doesn't include the time for launching the software and loading layers. GIS-based problem solving normally include following consecutive steps of processing: reading maps, searching attribute, defining procedure, finding tools and carrying out analysis. As we can see from the table, it is time consuming for the subjects even to answer even the simple questions using GIS. Time is rounded to every half a minute. Decimal degree numbers are saved up to 6 digits.

In Table I, [] denotes an array of objects, () denotes a value, * denotes zero or more values of the same pattern. For location question, a pair of latitude and longitude coordinates should be recorded as the answer. For relative location question, answers should be a radius and direction pair. The direction is calculated as the angle from origin to destination and then converted into eight encoded directions see Fig. 2. For distance question, an integer number with unit should be provided. For both proximity entity and proximity buffer question, comma delimited names should be returned. We correct all answers if they are not in the required format.

Finally, we end up with about 500 questions and about half of them with verified answers. We find that although our subjects have certain level of proficiency in GIS and are given examples to help them raise question they are still less willing to tackle those more challenging questions. For testing purpose, we also subjects to create 100 trick question that mimic the length and structure of a sentence but makes no sense such as "What is the name of Michael Jordan" (versus "What is the location of Columbus Ohio").

Table I. QUESTION CORPUS STATISTICS



Type	QA					
	Question	Answer Format	Example Answer	Minimum Steps	Avg. Time (mins)	% of questions
a	Where is Columbus?	[coordinates] (double double)*	39.964491 -82.999191	Add tools toolbar and enable the identify tool; Make the queried layer selectable; Navigate to or search queried object in attribute table by name; Click on the entity to identify its coordinates	1	32%
b	Where is Columbus perspective to Cleveland?	[distance unit direction] (double string string)*	150 miles southwest	Find coordinates of two objects using aforementioned procedure; Use a provided 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 using aforementioned procedure; 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; Search the origin and find the destination with the minimum near distance.	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. 3 is the framework we propose 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 data flow or interaction relationships.

The NLP component examines syntactic and lexical patterns in a sentence, and parses out useful constituents for subsequent processing. The ML component exchanges data with the NLP component and is responsible for training classifiers using features extracted from NLP analyzer. The ML component should also implement a learning strategy for tuning classification parameters based on new inputs. The ontology component defines both domain ontologies of spatial terms amd operational ontologies of spatial operations. Through ontological modeling, the system should be able to disambiguate the senses of natural language spatial terms and map them to spatial operations in a GIS. Finally, the GIS component takes inputs processed by ontological models and construct formal query to a spatial database. The formal query is often defined in the form of a spatial SQL.

Identify applicable sponsor/s here. *(sponsors)*

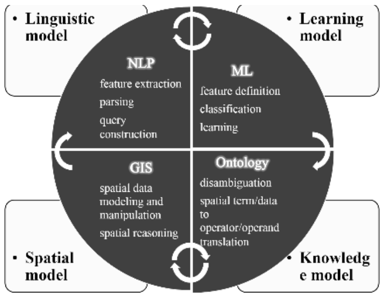


Figure 3. Synergistic framework

IV. Linguistic model

A. *Lexical analysis and feature extraction*

Our linguistic model is mainly responsible for syntactic analysis of sentences and extracting features from analytical results. We use a combination of Java-based Stanford part-of-speech tagger and python-based NLTK library. We find Stanford part-of-speech tagger offers a more robust tagging model than NLTK and therefore produces more accurate part-of-speech tags for sentences in our corpus. To do this, we write an interface in Python to use functions in Stanford tagger's jar file.

First, we use lexical frequency analysis to extract features from already classified sentences in the training set. These features could be lexicals, lexical lemma and lexical stems. For example, for the location question we find most frequently used tokens are lexicals "where, location, located, coordinates" (Table III). Therefore, these terms should be used as features for training a classifier. Using similar methods, we find most frequent features in each category. Also, we only select nouns and propositions as we observer they are good indicators of question types than other POS tags such as determinates and proper nouns. Finally, we only implement a unigram model for lexical analysis.

B. *Template generation*

Our program tags each sentence in the training set and generalize their part-of-speech patterns. By generalization, we mean merging all the part-of-speech sequences of those questions into one if their sequences are the same. We argue that question classification information is encoded in a large extent in these tag sequences. That is why we choose POS tag sequence as the first order filter for our proposed classification method.

Each tag sequence represents a template and each category corresponds to a set of templates. It is also required by a learning component that the template set should be able to expand whenever new sentences become available to a class. Also, there is no guarantee that different classes don't have the same templates. Therefore, lexical features should be used to filter questions in a second round. This is the second step of our classification verification process.

Finally, in Table II it can be seen that we end up with much less templates than the original number of raw sentences. The compression rate is higher for first three categories and lower for the last two categories. This is because complicated questions are more likely to be asked in different ways.

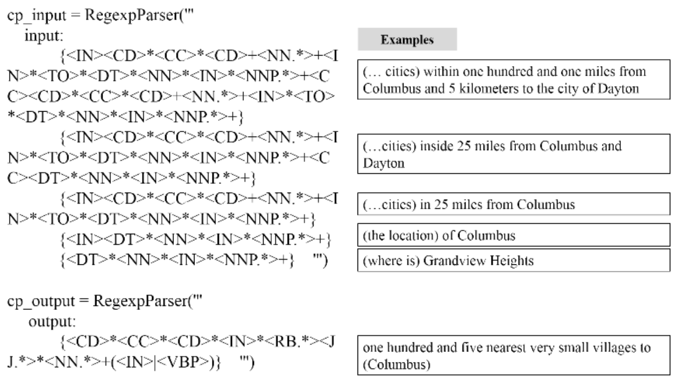
Table II. Lexical analysis

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 from natural language questions. Chunk parsers are most suitable in situations where only some, rather than all token, need to be annotated as certain functional chunks. Here, we define two chunkers in Python as shown in Fig. 4. For a regular expression-based chunker, rules will be applied in order. So if a chunk satisfies the first rule, it will not be evaluated on the second one.

In Fig. 4, example sentences correspond to each chunking rule defined on its left. Information in () indicates an adjacent auxiliary component that may be related to the current chunk. Here, we assume only one main spatial operator exists and these operators are encoded as prepositions. However, multiple spatial operators for calculating derivatives (such as a buffer) are permitted. The tags are standard Penn Treebank tags.



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

cp_output = RegexpParser('''
    output:
         {<CD>*<CC>*<CD>*<IN>*<RB.*><J
J.*>*<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 4. Input and output chunkers

Once chunks are extracted from a classified sentence (we will talk about classification in the next section), our system will start look for more granulated structures to locate operators, operands and constraints associated with that chunk. Worth noting is that not all chunks have three components but they all will include at least some of them. Fig. 5 illustrates how the spatial role of each token is labelled in each chunk given the (partial) sentence "5 nearest major cities within 20 miles of Columbus". Further, we implement several more chunkers: number chunker, constraint chunker, operand chunker and operator chunker to extract tokens according to their semantic roles. For space limitations, we can't include all of our implementations here.

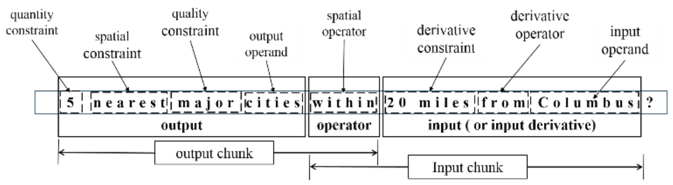


Figure 5. Spatial role labelling in chunks

As readers notice in the input and output chunk parser definitions, we include an <IN> tag at the beginning of a input tag sequence except for the last rule, and at the end of each output tag sequence. This is because without an <IN> tag, input and output chunker are simply both noun phrases and therefore can't be differentiated.

V. Machine learning model

A. Longest common sub tag sequence through dynamic programming

Previous machine learning approaches for question classification rely entirely on n-gram features. This way of extracting features that is significant only on a statistical sense doesn't quite capture the real characteristics of a sentence as we human do. Lots of indiscriminately selected features are also lack of meaning and redundant. Redundant features may significantly increase training time and therefore hamper the performance for classification and tuning a classifier. Hence, we here propose a lightweight classifier using longest common sequence algorithm in dynamic programming along with a voting algorithm for classification verification. Our approach is able to iteratively train our classifiers and update them at a manageable minimal cost.

Our template matching algorithm is based on the dynamic programming algorithm of finding the longest common subsequences (LCS) between two or more input strings [30]. For the general case of LCS problem with an arbitrary number of input sequences, the problem is NP-hard [31]. The worst case running time of a na ïve algorithm for finding the LCS among m sequences is T1 (Fig. 6) where n_i is the length of the first sequence. The complexity for two sequences of n and m elements is T2 and a dynamic programming-based algorithm with memoization table will reduce the complexity to T3.

$$\mathrm{T1} = O\left(2^{n_1}\sum_{i>1} n_i\right)(i = 1, \dots, m)$$

$$\mathrm{T2} = O(2^n m)$$

$$\mathrm{T3} = O(m \times n)$$

Figure 6. Time complexity of LCS algorithm

For LCS-based template matching algorithm, we give the formal definition as the following. Given a finite tag sequence $T=t_1, t_2, \dots$, we define a subsequence T' of T to be any sequence which consists of T with between 0 and m tags deleted. For example, [DT NN] (e.g. the location), [NN IN NNP] (e.g. location of Columbus), and [DT NN IN NNP] (e.g. the location of Columbus) are all subsequences of [DT NN IN NNP]).

As we can see, LCS encodes some latent relations between tag sequences and can be used to measure the similarity between observed lexical sequences. We denote T'<T if T' is a subsequence of T. Given a set $R = \{T1, T2, \dots, Tp\}$ of sequences, we define the Longest Common Subsequence of R (henceforth, LCS(R)) as a longest sequence T such that T < Ti, for i = 1…p.

Clearly, LCS(R) is not unique as there may be multiple sequences satisfying the definition. However, for our problem we are only concerned with the length of LCS(R), and index of the reference sequence, therefore variation in set composition is not an issue.

The algorithm for calculating the LCS between two tag sequences x and y expressed in Fig. 7. x and y are arrays of sequential tags for a reference sentence and a new sentence respectively. We define the stepwise result c[i,j]=LCS(x[1..i],y[1..j]) and the final length result is c[m,n]=LCS(x[1..m],y[1..n]). The peudo code for this algorithm with memoization is in Fig. 8. With a memoization table, we can achieve a time complexicty of T3.

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

Figure 7. LCS using dynamic programming

```
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                     memoization
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]
```

Figure 8. LCS pseudo code

Now, we can use the LCS algorithm to calculate the similarity between a tag sequences of a new question (henceforth, new sequence, or NS) and a reference tag sequence of a template (henceforth, reference sequence, or RS). Fig. 9 is an example of how a new sentence is matched with one of the templates from the proximity buffer category.

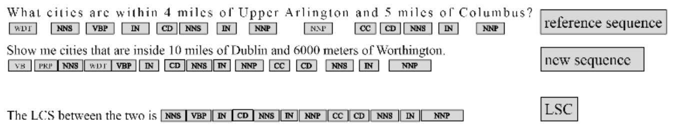

What cities are within 4 miles of Upper Arlington and 5 miles of Columbus?
WDT NNS VBP IN CD NNS IN NNP NNP CC CD NNS IN NNP
reference sequence
Show me cities that are inside 10 miles of Dublin and 6000 meters of Worthington.
VB PRP NNS WDT VBP IN CD NNS IN NNP CC CD NNS IN NNP
new sequence
The LCS between the two is NNS VBP IN CD NNS IN NNP CC CD NNS IN NNP
LSC

Figure 9. Template matching

A double precision similarity metric is also developed to measure the similarity between two sequences. The similarity between the new sequence and the ith reference sequence (across categories) is defined as S_i:

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

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

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

If there is a tie between several templates $\{RS_j : j\in[1,\ldots,n]\}$, all of these templates are kept for further verification using the voting algorithm discussed next. The similarity index for question in Fig. 9 is 0.86. We find 0.5 may be an appropriate threshold for classicization.

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 for concurrent control [32]. Here, we borrow the idea from Maekawa's and develop some new parameters for classification verifier.

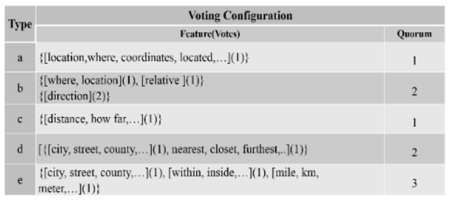
We define a feature set which corresponds to the quorum set in Maekawa's, feature votes which corresponds to grant messages in Maekawa's and quorum which corresponds to the minimum number of permissions in Maekawa's. The reason we find voting algorithm a relevant analogy here is because the nature of question classification using lexical features is a concurrent process. For example, for location question, words like "where, location, located" are all indicators to the question type and therefore a program doesn't need to wait until all evidences are found to proceed. Such flexibility is not available in traditional machine learning classifiers as decision making are sequential. Also, for relative location question, a word like "direction" imposes a same effect as a combination of two words "relative location". Therefore, different features should be granted with different number of votes for determining a class (Table III).

Here, we also define one or more sets of features for each type of questions along with the number of votes for each feature. These features are normally high frequent lower case lexicals as discussed previously in the NLP section. We also include lemmas forms, which are the canonical form of a word, or headword. Due to space limitations, we only include the most frequent features in the table. The feature set, feature votes and quorums are listed in 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 for any number of occurrence of a single feature in the subset. Once a feature in a subset is counted, additional occurrence of features in the same subset will not be counted to prevent double dipping. For example, in the phrase "location of Columbus and location of Dayton", location will only be counted once.

When the voting algorithm executes, the program will first find the category of the template using LCS algorithm, and then create threads to verify the classification by checking whether features in that category satisfy the quorum. The detailed verification step is like this. Each subset will be handled by one tread. If a feature in the subset is found, its votes will be added to the sum of votes of that subset. Once the sum of votes of any subset is greater than or equal to the quorum of that category, a match will fire and all threads can exit. We can then determine the new question is of the verified class. If there is still a tie, we simply choose the type that has the most sample questions to hammer down the decision.

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*

If Wikipedia was the fountain of knowledge for humans, ontologies would be the fountain of knowledge for computer programs. Computational ontologies are encoded as RDFs and OWLs which are W3C (World Wide Web Consortium) standards. One commonly used online ontological database is DBpedia. DBpedia provides a crowd-sourced community platform to extract various kinds of structured information from Wikipedia in 111 languages and combines this information into a huge, cross-domain knowledge base. DBpedia uses RDF as the data model for representing ontological information and make it accessible through SPARQL

SPARQL is a query language to retrieve data from RDF-based database. It has become a standard by the RDF DAWG (Data Access Working Group) of the W3C, and plays a key role to the semantic web. From our experience, query spatial ontologies through SPARQL is not as hassle free as query other domain specific ontologies especially through online endpoints. But we still find it useful for potentially more in-depth usage in geographic question answering in the future.

Besides common sources like Wikipedia, the spatial ontologies on DBpedia are also extracted from the GeoNames. The GeoNames is a gigantic online database of geographic named entities including all countries in the world and places in each county. It includes over 8.3 million toponyms, and is encoded in RDFs and queryable through SPARQL too.

GeoNames ontologies can be used to disambiguate named entities in a sentence. Fig. 10 shows the result of querying Columbus as a city of United States through GeoNames ontologies. This query returns more than 163 records among which Columbus in Ohio and Columbus in Georgia are both listed. We also see Columbus as county of North Carolina which may not be very familiar with most of ordinary people.

It is helpful that GeoNames ontologies rank entries by familiarity so most famous entities are positioned on top of the return. However, for building real systems a disambiguation step is inevitable in spite of such ranking. For example, cooccurrence of entities like "Columbus" and "OH" is a good indicator for disambiguating Columbus as a city from Ohio.

For our current experiments, as all our entities are from the state of Ohio, by default only places from Ohio will be returned from DBpedia.

B. *Operational ontologies*

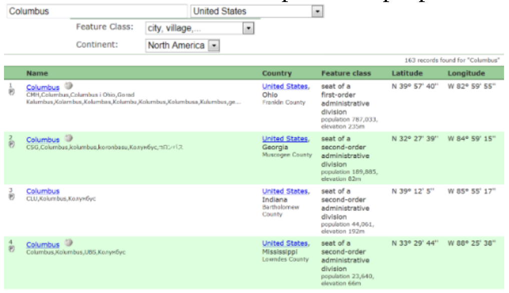
Even with abundance of ontology resources from DBpedia and GeoNames, we are still missing an important array of ontologies, namely the operational ontologies. Spatial operational ontologies are specifications about the definition of spatial operators and operands in a spatial database. It also governs the mapping from natural language terms to operational counterparts in a database.

Operand specification includes definition of entity classes and class relations. For example, Columbus is a City type entity and City is a class. Ohio is a State type entity and State is another class. As a City class is defined as spatially contained in a State, Columbus is therefore automatically contained in Ohio. This spatial containment relationship is encoded as class relation.

Operand specification should also include database schemes which tells a QA system where to look up entries. For example, it should define the mapping of named entities such as cities to the scheme of a relational table. Therefore, once Columbus is resolved as a city the system should automatically know where city information is encoded, either in a table or column, for example.

Operator specification includes spatial relation class and relations between relation classes. For example, both spatial terms “within” and “inside” should be mapped to a relation class called “in”. It then should further associate this relation class with permitted operations. For example, operators like “contain, completely contain, within, completely within” should all be permissible to the relation “in” and operations like “intersect” should return true for the relation “in”.

In the context of a larger project, we are going to formalize all these operational ontologies in RDFs. But for this research, we hard code some of them for experiment purposes.



Columbus | United States

Feature Class: city, village,...

Continent: North America

163 records found for "Columbus"

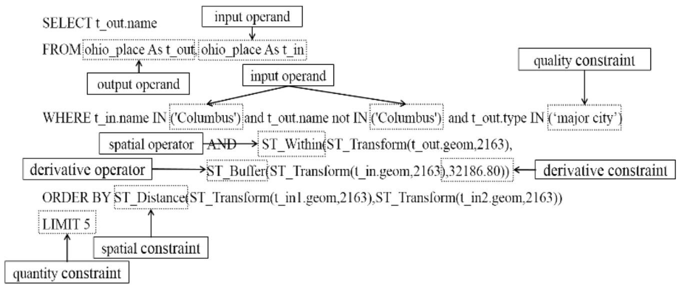
	Name	Country	Feature class	Latitude	Longitude
1	Columbus CMH,Columbus,Columbus i Ohio,Gorad Kalumbus,Kolambus,Kolumbas,Kolumbu,Kolumbus,Kolumbusa,Kulumbus,ge...	United States, Ohio Franklin County	seat of a first-order administrative division population 787,033, elevation 235m	N 39° 57' 40"	W 82° 59' 55"
2	Columbus CSG,Columbus,kolumbus,koronbasu,Колумбус,コロンバス	United States, Georgia Muscogee County	seat of a second-order administrative division population 189,885, elevation 82m	N 32° 27' 39"	W 84° 59' 15"
3	Columbus CLU,Kolumbus,Колумбус	United States, Indiana Bartholomew County	seat of a second-order administrative division population 44,061, elevation 192m	N 39° 12' 5"	W 85° 55' 17"
4	Columbus Columbus,Kolumbus,UBS,Колумбус	United States, Mississippi Lowndes County	seat of a second-order administrative division population 23,640, elevation 66m	N 33° 29' 44"	W 88° 25' 38"

Figure 10. GeoNames entries of Columbus as a U.S. city

VII. SPATIAL MODEL

A. *Spatial query templates*

To answer geographic question using spatial SQL, we create SQL templates for each category and fill in the blanks in the templates using parsed data. Fig. 11 is an example template for proximity buffer query with missing values already filled in. The query is to answer the same question as in Fig. 5.



```
SELECT t_out.name
FROM ohio_place As t_out, ohio_place As t_in
WHERE t_in.name IN ('Columbus') and t_out.name not IN ('Columbus') and t_out.type IN ('major city')
      ~~AND~~ ST_Within(ST_Transform(t_out.geom,2163),
ST_Buffer(ST_Transform(t_in.geom,2163),32186.80))
ORDER BY ST_Distance(ST_Transform(t_in1.geom,2163),ST_Transform(t_in2.geom,2163))
LIMIT 5
```

Figure 11. Spatial SQL for a proximity buffer type of question

B. *Language term to operational term mapping*

Our chunk parser parses out constituents like numbers, adjective, nouns along with their modified spatial entities and relation terms. Our operational ontologies will then map these constituents into spatial constraints, data and operators. These three pieces of information are represented in rectangles in Fig. 11.

VIII. RESULES AND EVALUATION

Our corpus is split them into 7:3 for training and testing respectively. Trick questions are only included in the testing set. Table IV is the result.

Table IV. CLASSIFICATION ACCURACY AND QUESTION ANSWERING ACCURACY

Classification Accuracy					
Type	a	b	c	d	c
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

Question Answering Accuracy and 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

We observe a higher precision and recall for constract cases using classification verification. Without verficiation, trick questions are all assigned to one of the five classes which is incorrect. It is also seen that classificatoin accuracy are higher for location, relative location and distance question while lower for the other two because feature structures for the first three classes are relatively simple.

We also find in a few cases when the word distance are used unnecessarily in various places they can confuse our classifer to the location question. That may explain why recall for relative location question is lower. Classification errors also occur when prepositional phrases are used at the beginning of a sentence such as “Relative to Columbus, where is Dayton”. These sentences only occurs in our test set but unfortunately slip out of our training set.

Moreover, category d and e are similar in lexical structure and can sometimes confuse our system. In a few cases, spatial entity questions can’t be correctly classified if a required feature such as “nearest” is missing. In that case, subjects think the return should be random but our current system gives none.

Overall, our system generates correct answers at a percentage of accuracy over 90%. There are a few cases that causes failure of our query. First is unkown quality constraints such as “capital (city)”. We haven’t encoded such specific knowledge in our database. Second is structure of that cluase which we also haven’t considered either. Last is case errors for example “Columbus” spelled as “columbus” as out POS taggers rely on correct cases to tag a token as proper noun.

Finally, our system delievers a question in a reasonablly short time of less than 3 seconds. The machine that runs our test is an i5 dual core Intel CPU with 4GB ram.

IX. Conclusions

Our proposed synergistic framework is proven to be able to provide effective solutions to answer non-trivial natural language geographic questions. Our dynamic programming-based template matching classifier can classify a sentence in polynomial time based on syntactic features of a sentence. Then, a voting algorithm-based classification verifier checks the classification results using semantic features. A synergy of the two produced a high accuracy for classification.

Evidence are also provided that GIS is an indispensable component to answer in-depth geographic questions as it offers spatial reasoning capabilities which encodes spatial reasoning expertise of humans. With both GIS and spatial ontology implemented, our QA system can produce correct exact answers to most of the questions in selected categories and this can be done within seconds.

References

[1] E. Bick, "A constraint grammar based question answering system for Portuguese," *Progress in Artificial Intelligence,* vol. 2902, pp. 414-418, 2003.

[2] P. R. Comas, J. Turmo, and L. Marquez, "Using Dependency Parsing and Machine Learning for Factoid Question Answering on Spoken Documents," *11th Annual Conference of the International Speech Communication Association 2010 (Interspeech 2010), Vols 1-2,* pp. 1265-1268, 2010.

[3] U. Schafer and B. Kiefer, "Advances in Deep Parsing of Scholarly Paper Content," *Advanced Language Technologies for Digital Libraries,* vol. 6699, pp. 135-153, 2011.

[4] A. Figueroa and J. Atkinson, "Using syntactic distributional patterns for data-driven answer extraction from the web," *MICAI 2006: Advances in Artificial Intelligence, Proceedings,* vol. 4293, pp. 985-995, 2006.

[5] 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.

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

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

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

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

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

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

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

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

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

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

[21] C. Du, J. Xu, J. Zhang, W. Si, B. Liu, and D. Zhang, "Spatial relation query based on geographic ontology," *Sixth International Symposium on Digital Earth: Models, Algorithms, and Virtual Reality,* vol. Proc. SPIE 7840, 78400F, 2010.

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

[28] 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.

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

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

[31] D. Maier, "The complexity of some problems on subsequences and supersequences," *Journal of the ACM (JACM),* vol. 25, pp. 322-336, 1978.

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