Ontology Based Segmentation of Geo-Referenced Queries

Mamoun Abu Helou

Politecnico di Milano, Dipartimento di Elettronica ed Informazione,
V. Ponzio 34/5, 20133 Milano, Italy
abuhelou@elet.polimi.it

Abstract. The last generation of search engines is confronted with complex queries, whose expression goes beyond the capability of the Bag of Word model and requires the systems which understand query sentences. Among these queries, huge importance is taken by geo-referenced queries, i.e. queries whose understanding requires localizing objects of interest, where the user location is the most important parameter. In this paper, we focus on geo-referenced queries and show how natural language analysis can be used to decompose queries into sub-queries and associating them to suitable real-world objects. In this paper we propose a syntactic and semantic approach, which uses syntactic query segmentation techniques and the ontological notion of geographic concepts to produce good query interpretations; an analysis of the method shows its practical viability.

Keywords: Query Segmentation, Query Understanding, Geo-Referenced Query, Multi-Domain Query.

1 Introduction

Search engines perform poorly on complex queries [3]. When a query involves multiple domains and their interconnections, i.e., queries over multiple semantic fields of interest, search engines fail in understanding the query's meaning, also because they try to use all the query information in order to locate one page containing all the results. In this paper, we propose an approach to complex query understanding which focuses on the sub-problem of query segmentation. Such step is essential for decomposing a complex query into sub-queries, and then answering each sub-query independently, as contemplated by Search Computing (SeCo)[2].

However, understanding a natural language (*NL*) query requires the application of syntactical, semantic and conceptual knowledge to resolve the ambiguity that abounds in *NL*. The output desired from a query understanding process must include the objects, properties of objects and relationships among the query objects. In this paper, we focus upon geo-referenced queries, e.g. queries which ask about properties of objects which are placed at specific positions. These queries are very much used in practice, and are the majority of queries which are asked from mobile devices.

This paper is structured as follows. Section 2 presents the background and the preliminaries. Section 3 presents the related work. Section 4 describes the design and implementation. Section 5 presents the experimental results. Finally, we conclude and present some plans for future development.

A. Harth and N. Koch (Eds.): ICWE 2011 Workshops, LNCS 7059, pp. 135–144, 2011.

2 Preliminaries

SeCo aims at the construction of a platform for multi-domain queries across search services. The project is addressing many research problems, including search service engineering and registration, efficient join methods for search services, and flexible query execution engines. SeCo uses the Semantic Resource Framework (SRF) [2] which is a multi-level (conceptual, logical, and physical level) description of data sources for *SeCo* applications. The higher layers provide an abstract semantic description of the services, building on the notions of *Service Marts* and *Connection Patterns*. The lower layers (*service interfaces* and *access patterns*) are concerned with the physical properties of the services. Ideally, every service conceptually belongs to a Service Mart. A Service Mart is structurally defined by means of attributes. Two Service Marts can be connected by a *Connection Pattern*. At the logical level, each Service Mart is associated with one or more access patterns representing the signatures of the service calls. *Access patterns* contain a subset of the attributes of the Service Mart, which are tagged with either I (input), or O (output). Attributes can also be tagged as R (ranking), to denote attributes that are used for ordering result instances. Ranking is particularly important in SeCo, because it allows mastering the combinatory explosion of multi-domain queries typical in Search Computing. For example, a query such as "*Find me a theater in San Francisco showing a romantic movie near Hilton hotel"* requires the user to manually extract and combine the answers from various queries, and this is an intricate and tedious job.

Syntax trees are widely used to preserve the original information conveyed by the query, a syntax tree is an (ordered, rooted) tree that represents the syntactic structure of a string according to some formal grammar. Figure 1 illustrates part of the Stanford Syntactic pares tree [6] of the example query. The parse tree is entire structure, starting from the *ROOT* and ending in the leaf nodes (*find, ..., hotel*). The following abbreviations given by the syntax tree help understanding the part of speech used for

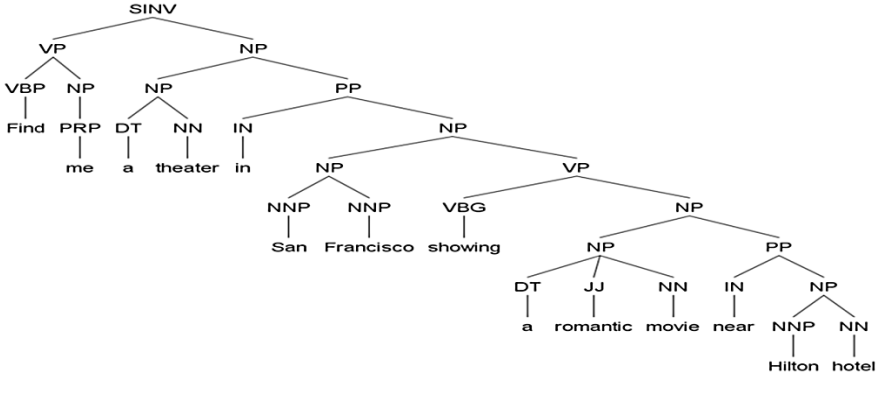

Fig. 1. The syntactic tree of the example query

the query words : *NP* is a noun phrase *(e.g. a theater), NN (e.g. theater) and NNP (e.g. Hilton)* are used for nouns and proper nouns, respectively. *VP (e.g. find me),* and *PP (e.g. near Hilton hotel)* are used for verb, and proposition phrases, respectively. *JJ (e.g. romantic)* is used for adjectives, and *IN (e.g. in)* is used for preposition or subordinating conjunction.

YAGO [5] is a large semantic *KB* which has been automatically built from Wikipedia, GeoNames[1], and WordNet[2], and contains nearly 10 million entities and events, as well as 80 million facts representing general world knowledge. In YAGO, knowledge is represented in the RDFS model. This model can be seen as a directed labeled multi-graph, in which nodes represent entities and edges represent relationships between the entities as illustrated in Figure 2. Furthermore, both the instances (such as *San Francisco*) and concepts, i.e., groups of similar instances (such as *city*), are nodes in the RDF graph. An instance is linked to its concept by the relation *type*. A concept is linked to a more general concept by the relation *subclassOf*. Instance of a sub-concept automatically inherits from super-concept, and can also be generalized as super-concept. Every concept's relation will apply automatically to all its instances (sub-classes) that exist somewhere down in the hierarchy.

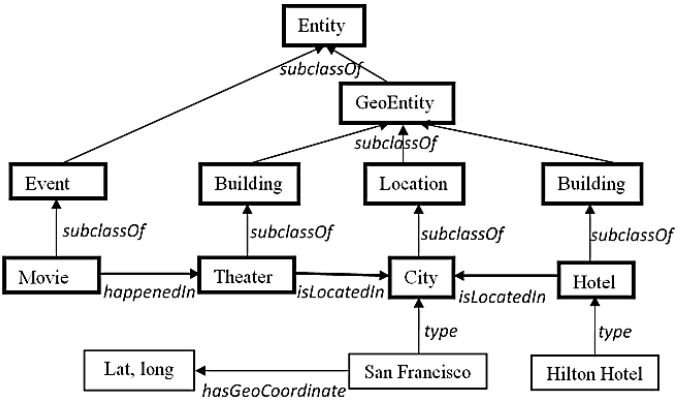

Fig. 2. RDF Model representing the example query based on YAGO

3 Related Work

The research work on Query Segmentation *(QS)* is focused on how to decompose a query into sub-queries. In [4], authors have shown that query segmentation has a positive impact on the retrieval performance. The segmentation process takes a user's query and automatically tries to separate the query words into segments so that each segment maps to a semantic component. Ideally, each segment should map to exactly one "concept" [8]. Recent work in *QS* [1] used a supervised learning method. However, this approach, as all supervised learning methods, requires a significant number of labeled training samples and well designed features to achieve good performance.

[1] http://www.geonames.org/
[2] http://wordnet.princeton.edu/

This makes it hard to adapt in real applications. As an alternative, [8] suggests unsupervised method based on expectation maximization. This approach, as it happens with most unsupervised learning methods, heavily relies on corpus statistics. In some cases, highly frequent patterns with incomplete semantic meaning may be produced. These segmentation algorithms take into account the sequential ordering of words, and do not study non-adjacent terms, therefore these approaches just deal with keyword-based short queries but hardly adapt to long *NL* queries, including complex queries. Also, they do not try to identify named entities or to assign class labels. *NL* Interface (*NLI*) systems to structured data use an underlying ontology to drive the meaning to the queries expressed by a user. [7] utilized pattern matching of a *NL* query to subject-property-object triples in a knowledge base (*KB*), before converting the query to one of SPARQL. [9] employed a named entity recognition engine, and supplemented it with more entity types and relation types to convert *NL* queries into SeRQL expressions. Nevertheless, the focus of *NLI* systems is on efficient porting interfaces between different domains rather on the understanding itself. Besides that, the understanding step should not be mixed up with the answering step. That is, we focus in understanding the search text queries leveraging the *KB* features although there is no matched answer to it in a *KB* of discourse.

4 Methodology

We propose a method with the objective to split a given query into sub-queries by combining the syntactic parsing with the knowledge encoded in a *KB* (ontology) to identify entities and their respective relations in a query. The method views a query as a sequence of entities and relations. The next issue is to determine the relations which link the entities. Valid relations between entities are actually constrained by the *KB*. Therefore, transformation of the *NL* query into a set of ontology concepts (i.e. classes, instances, properties) which is based on the assignment of a proper ontology concept to the query words is carried out first. Then the relations between the extracted entities are identified using the ontology. This is explained below in this section. Therefore, the main focus is to correctly recognize entities and determine their relations as expressed by the query.

Thus, the proposed method can be broken into two phases: *Query Analysis (QA)*, where the key *concepts/"Web objects"* will be extracted from the given query. , Here, the Web objects are a new way of abstraction to reinterpret concept organization in the Web, and go beyond the unstructured organization of Web page. The second phase is *Query Interpretation (QI),* where the possible object properties and entities are identified, and also the relationships among the extracted objects are discovered.

4.1 Query Analysis (*QA*)

The *QA* focuses on the identification of key *concepts* in the query. *QA* decomposes the user query into *concepts*, where each *concept* represents one search objective in a specific domain. The *concept* can either be a *simple concept* which consists of one word (e.g. hotel), or a *complex concept* consisting of multiple words (e.g. train station). *QA* employs two steps.

Step 1*: Morphological Analysis:* In this step we use the Stanford parser to get the syntax tree for the user's query. Shallow parsing is adopted to divide the sentences into a series of words that together compose a grammatical unit, mostly noun phrase (NP), and preposition phrase (PP), and also running a tokenizer, part-of-speech tagger.

Step 2*: Concept Identification:* This step identifies the key *concepts* in the query through a matching process, between the query words (*QW*) and the *KB's* concepts, to extract the query *concept*s "*geo concepts*" (*GC*). A *concept* is a *GC* if it is mapped to one of the *KB* geographical entities. The matching process, first checks if the query word is a *GC*, if not it checks if any synonym of the query word is a *GC*, failing to which it combines the query word with its consecutive word and checks if this composite word is *GC,* using the above 2 steps. Knowing that the geo spatial *concepts* are nouns, only the nouns among the *QW* are examined. For instance, running the *QA* over the example query the *GC* list :{ *theater, hotel*} was identified.

4.2 Query Interpretation (*QI*)

QI tries to extract any possible geo spatial information "*geoEntities"* (i.e. address, and geo spatial entity name). It also extracts the *concept* properties which could be seen as service invocation parameters and filtering criterion. Besides that, it identifies the relation among the extracted parts based on the defined relation in the *KB*. *QI* performs the following steps.

Step 1*: Concept properties extraction.* The adjectives and nouns associated with the *concepts* represent either a property of the entity or a more specific type for that entity than the type expressed by the *concepts* itself (e.g. Hilton hotel). For that, the adjectives and entities name are extracted based on the *NP* which has a *concept*. The *concept's* consecutive nouns are considered as entity name, while the adjectives are filtering criteria. Running step1 over the example query the name property "*Hilton*" was extracted for the *concept "hotel"*.

Step 2*: GeoEntity extraction.* The *geoEntity* (*GE*), i.e. the entity with a permanent physical location on Earth can be described by geographical coordinates, consisting of latitude and longitude. Any geo-spatial entity/*concept* identified in the query should be associated with at least one component of the *address* field (street name, zip/postal code, city, country). The query may also contain the name for a *geo concept* (name of the geo-spatial entity), which would be extracted in (step1) e.g. Eiffel Tower. First candidate *GEs* are collected, we process the syntax tree for this purpose, and assume such entities are the *NPs* which are a child of a *PP*. And also the *GC* will be filtered and checked if any *concept* with its extracted entities names (step 1) would express a *GE*. Then, the *geo validation* process is performed to confirm these candidate entities and check if they are real world geo spatial entities.

The validation process is performed in two ways. Firstly, with the help of the *KB*. As mentioned above, YAGO the adopted *KB* already holds information about geographical entities (i.e. GeoNames entities) which holds the names of geographical entities e.g. cities, streets, monuments etc. Therefore, we are able to identify such entities and also provides the geo-concept under which this *GE* falls. Secondly, via

Google GeoCoder API[3], or similar APIs; the test succeeds if we are able to retrieve the address components of the candidate entity. Later, the extracted entities are processed to identify the address components by finding the best match between the name of the candidate entity and the *GeoCoder* results components. For this comparison we use *Levenshtein distance metrics, i*f such distance is less than a certain threshold. The Levenshtein distance between two strings is the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character. And also the relation of the extracted *GE* components are defined based on the hierarchal representation of the address components (i.e. street is located in a city which is located in a country). Running step 2 over the example query, the chunk *"San Francisco"* was recognized and mapped to the ontology concept "*city"*.

Step 3*: Relation identification.* To achieve the best possible query interpretation, we retrieve and analyze the potential relations between the identified *concepts* and entities, based on the defined relations in the *KB*. These relations are very important as they add descriptions to the *concept,* and define their behavior by adding rules and constraints. To resolve these relations the following steps are employed.

1. *Candidate relations extraction*. The possible relations for *concepts* will be extracted from the *KB.* Distinction is made between the relation's *Domain* and *Range* concepts. The query words, which have not been recognized in the extraction phases before (i.e. as a *concept* or *GE)* will be searched to match the extracted relation's *Domain/Range* concepts. For instance, the word "*movie"* is an instance of the *event* class which is a *Range* concept of the *happendIn* relation.
2. *Filtering the Improper relations*. The identified *concepts* and entities might have more than one possible relation. Similarly to [9], the candidate relations are filtered based on the property position in the hierarchy of concepts and properties. Initially, all possible matches between the extracted *concepts* and relation's *Domain/Range* concepts is performed, and then they are ranked based on the following factors: (i) on the sub property relation in the ontology, i.e. the property which placed at deeper levels in the property hierarchy has higher score, (ii) the position of the domain and range classes of the property, that is, a relation with more specific domain and ranges are ranked higher, and (iii) if any ambiguity raise up (i.e. the concept could be mapped to 2 or more concepts) the closet concepts will be related. For instance, *Location,* the sub-class of *GeoEntity*, is a *Range* concept of *isLocatedIn* which is a sub property of *placedIn* that has the *GeoEntity* as *Range* concept. Thus, the *isLocatedIn* is preferred instead of the *placedIn* to relate the location concepts or its sub classes (e.g. city, county).

	Domain Concepts	Relation	Range Concepts
1	Person	wasBornIn, deidIn, livesIn	Location **(city)**
2	GeoEntity (**structure**, **location**)	placedIn (**isLocatedIn),…**	Location **(city)**
3	Event (**movie**)	**happenedIn**	GeoEntity (**structure** , **location**)

Fig. 3. The example query potential relations

[3] http://code.google.com/apis/maps/

Figure 3 shows part of the possible relations and their *Domain*/*Range* concepts for the example query. The bold *concepts* are the one recognized in the query. The first relation will be discarded since the concept *person* was not found among the query words. The other relations are kept and resolved to {movie *happenedIn* theater, theater *isLocatedIn* San Francisco, hotel *isLocatedIn* San Francisco}.

Additionally, a spatial nearness relation which maintains the context of the geo- referenced query is handled based on existing keywords (e.g. near, close to,…etc). Pattern matching is carried based on the patterns in Figure 4 , where C_i and $\aleph$ represent the *concept* and *nearness* keywords, respectively. Else a conjunctive (default) connection will be used by relating the closest *concepts* by walking through the syntax tree. Once step 3 is carried out, the relations among the *concepts* for the example query was defined, as follows :{ theater *near* hotel}.

Figure 5 shows the result of running the system over the example query as a directed graph, where the square nodes are the *concepts* in the user's query. The ellipses are the *GEs*, and the concept's properties which will serve as the services invocation parameter. The graph edges are the relations among these nodes that *SeCo* engine [2] will utilize as a filtering and join criteria; each concept node with its attached properties would be recognized as a sub query which will be mapped to a Web service. In SeCo each sub query ideally should be mapped to the appropriate access pattern of a Service mart by a Mapping tool (this would be a feature we looking to do in the future). The following represent an access pattern (AP) for a movie, theater, and hotel respectively.

Movie (Title^{O}, Director^{O}, Score^{R}, Year^{O}, Genres.Genre^{I}, Openings.City^{I},Openings.Date^{I}, Actor.Name^{O})
Theatre (Name^{O}, Address^{I}, City^{I}, Country^{I}, Address^{O}, City^{O},Country^{O}, Distance^{R}, Movie.Title^{O})
Hotel (Name^{I}, Address^{I}, City^{I}, Country^{I}, Addess^{O}, City^{O}, Country^{O}, Distance^{R}, Rating^{R})

The Movie AP filters the movies by time (e.g., whose opening date in US is recent enough) and genre (e.g. romantic movies) and then extracting them ranked by their quality score. The theater AP offers a list of movie theatres with the related films ordered w.r.t. the distance from a given location. Theatre AP is connected to Movie AP via a connection pattern "Shows" using a join on titles attribute. Once the theatres have been decided, then we look for a near Hotel. The hotel AP offers a list of hotel ordered w.r.t the distance from a given address (the theater address) and filtered based on the name of the hotel (Hilton). At the end the user should have and order list w.r.t the distance between the theaters which are showing a romantic movies and the hotels which called Hilton in san Francisco.

5 Experiments

The RestQueries dataset provided by Mooney's group[4] was used in the experiments, consisting of 251 queries about restaurants. Out of the 251 queries 13 were redundant and removed; the rest were manually annotated with the *GC* class, *GE* address

[4] http://www.cs.utexas.edu/ users/ml/nldata/restquery.html

components (street, city, administrative area, country), and also the relation among the *GC* and *GE* as well as the *GE* relations and the concept properties (adjective, name) relations. The experiment is designed to measure the capability of the proposed system to extract the geo spatial concepts "*concepts*", the geo spatial entities components "*geoEntities*", the concepts *properties*, and also the *relations* among aforementioned parts, the relations are; Concept-to-Concept "*cc*", Concept-to-geoEntity "*ce*", geoEntity-to-geoEntity "*ee*", and Concept-to-Propriety "*cp*".

Query Pattern	Concepts spatial relation
$c_1\ \mathcal{N}\ c_2$	$c_1\ \mathcal{N}\ c_2$
$c_1\ \mathcal{N}\ c_2$ *and* c_3	$c_1\ \mathcal{N}\ c_2$ *and* $c_1\ \mathcal{N}\ c_3$
c_1 *and* $c_2\ \mathcal{N}\ c_3$	$c_1\ \mathcal{N}\ c_3$ *and* $c_2\ \mathcal{N}\ c_3$
c_1 *and* $c_2\ \mathcal{N}$	$c_1\ \mathcal{N}\ c_2$
c_1 *and* c_2	c_1 *and* c_2

Fig. 4. Spatial nearness patterns, where C_i and N represent the concept and spatial nearness relation, respectively

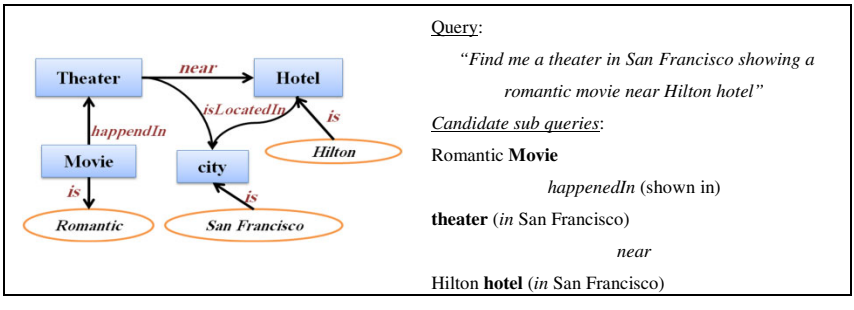

Fig. 5. The result of processing the example query

The correctness of the system was measured based on *Recall* and *Precision*. *Recall* is defined as the ratio between the numbers of correctly extracted parts by the system (true positive "TP") to the total number of manually tagged parts in the dataset (TP and false negative "FN" which been miss extracted). *Precision* is the ratio between the numbers of correctly extracted parts (TP) to the total number of the extracted parts using the system (TP and false positive "FP" which been extra/wrongly extracted). Figure 6 reports the results of running the system over the RestQueries. The *Precision* and the *Recall* were recorded for each extracted part. The experiment data and the result are available at ([5]).

The system was able to extract 297 concepts, then 101 out of these concepts have been filtered in (*QI step2)*, for example, "mountain view" is *GE*. Thus 100%, 196 concepts, have been all correctly extracted. 91.4% of the *GE* address components

[5] http://home.dei.polimi.it/abuhelou/data.htm

were correctly extracted. The reason behind the incorrect and the missed extraction is due to syntax tree, and also the *geo validation* process. For example, *"what are some good places for ice cream on blanding ave in alameda"*, "blanding ave" was tagged with verb phrase which cause the system to miss such entity. And the *geo validation* process was unable to recognize "fairgrounds dr", in the query *"where is a good american restaurant on fairground dr in sunyvale"*, as a street in the city "sunnyvale". 69.1% of the concepts properties was correctly extracted, again the syntax tree affect the extraction process. For instance, the adjective properties e.g. *italian*, was tagged as nouns, which caused 37 missed adjectives as well as 37 extra entity name. Furthermore, the syntax tree split some *NPs* into *NP* and *VP* which cause the system again to wrongly extract part of this name, which also will be considered as an extra extraction , for instance, the nouns "ice, and ave" were extra extracted and the name "ice cream" and the street "balnding ave" was missed as well as their relations. The overall correct extracted relations were 85%. However, the relation extraction was directly affected by the abovementioned extracted parts, the main factor was the property extraction.

An important consideration is that, the queries in this dataset mainly consist of only one geo-spatial concept. As a next step creation of a dataset consisting of more complex queries, having multiple geo-spatial concepts, attributes and relations is anticipated to test the approach more effectively.

	Over all				GeoEntities			
Extracted Parts	Concepts	GeoEntities	Properties	Relations	Street	City	Adm. A.	Country
TP	196	290	291	545	46	167	56	0
FP	0	4	130	96	0	9	0	13
FN	0	7	58	71	20	0	8	0
Precision	100%	98.6%	69.1%	85.0%	100%	94.9%	100%	0.0
Recall	100%	97.6%	83.4%	88.5%	69.7%	100%	87.5%	-

	Concept classes				Relations			Concept Properties	
Extracted Parts	Restaurant	Café	Bakery	place	ce	ee	cp	Adjectives	Names
TP	138	4	20	34	245	53	247	238	53
FP	0	0	0	0	4	0	92	10	120
FN	0	0	0	0	7	13	51	53	5
Precision	100%	100%	100%	100%	98.4%	100%	72.9%	96.0%	30.6%
Recall	100%	100%	100%	100%	97.2%	80.3%	82.9%	81.8%	91.4%

Fig. 6. The results of running the RestQueries dataset

6 Conclusions

This paper presents an approach for understanding natural language queries using geo-localizations by splitting long queries into sub-queries and understanding the role of each word in each sentence, specifically by extracting objects with their properties and identifying the geographic relationship between them. The precision and recall of the method are sufficiently high to warrant its use for geo-referenced queries. Future plans include improving and extending the method, by addressing not only geo-localized queries, but also general compound queries; we aim again at combining the syntactic method for query decomposition to other semantic methods and heuristics,

using general-purpose ontological knowledge. In this way, it will be possible to understand if the method "scales" to arbitrary query decomposition, or instead its good performance descends from the extensive use of geo-localizations concepts. In future, the objective is to test the proposed approach for more complex queries. Furthermore, a stronger integration of Semantic Resource Framework with the proposed approach is anticipated to improve the results.

Acknowledgements. This research is part of the Search Computing (SeCo) project, funded by the European Research Council (ERC), under the 2008 Call for "IDEAS Advanced Grants", a program dedicated to the support of frontier research.

References

1. Bergsma, S., Wang, Q.I.: Learning noun phrase query segmentation. In: EMNLP-CoNLL 2007, pp. 819–826 (2007)
2. Ceri, S., Brambilla, M. (eds.): Search Computing. LNCS, vol. 5950. Springer, Heidelberg (2010)
3. Chen, Y., Zhang, Y.-Q.: A query substitution – search result refinement approach for long query web searches. In: WI-IAT, pp. 245–251 (2009)
4. Guo, J., Xu, G., Li, H., Cheng, X.: A unified and discriminative model for query refinement. In: ACM SIGIR Conference on R&D in IR, pp. 379–386 (2008)
5. Hoffart, J., et al.: YAGO2: Exploring and Querying World Knowledge in Time, Space, Context, and Many Languages. In: WWW 2011, Hyderabad, India (2011)
6. Klein, D., Manning, C.D.: Fast exact inference with a factored model for natural language parsing. In: Becker, S., Thrun, S., Obermayer, K. (eds.) Advances in NIPS, pp. 3–10. MIT Press (2002)
7. Kaufmann, E., Bernstein, A., Fischer, L.: NLP-Reduce: A "Nave" but Domain Independent Natural Language Interface for Querying Ontologies. In: Demo-Paper at the 4th European Semantic Web Conference, pp. 1–2 (2007)
8. Tan, B., Peng, F.: Unsupervised query segmentation using generative language models and Wikipedia. In: WWW 2008, pp. 347–356. ACM (2008)
9. Tablan, V., Damljanovic, D., Bontcheva, K.: A Natural Language Query Interface to Structured Information. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS(LNAI), vol. 5021, pp. 361–375. Springer, Heidelberg (2008)