Natural Language Queries over Heterogeneous Linked Data Graphs: A Distributional-Compositional Semantics Approach

Andre Freitas
Insight Centre for Data Analytics
National University of Ireland, Galway
IDA Business Park, Lower Dangan, Galway, Ireland
andre.freitas@deri.org

Edward Curry
Insight Centre for Data Analytics
National University of Ireland, Galway
IDA Business Park, Lower Dangan, Galway, Ireland
ed.curry@deri.org

ABSTRACT

The demand to access large amounts of heterogeneous structured data is emerging as a trend for many users and applications. However, the effort involved in querying heterogeneous and distributed third-party databases can create major barriers for data consumers. At the core of this problem is the *semantic gap* between the way users express their information needs and the representation of the data. This work aims to provide a natural language interface and an associated semantic index to support an increased level of vocabulary independency for queries over Linked Data/Semantic Web datasets, using a *distributional-compositional semantics* approach. Distributional semantics focuses on the automatic construction of a semantic model based on the statistical distribution of co-occurring words in large-scale texts. The proposed query model targets the following features: (i) a principled semantic approximation approach with low adaptation effort (independent from manually created resources such as ontologies, thesauri or dictionaries), (ii) comprehensive semantic matching supported by the inclusion of large volumes of distributional (unstructured) commonsense knowledge into the semantic approximation process and (iii) expressive natural language queries. The approach is evaluated using natural language queries on an open domain dataset and achieved **avg. recall=0.81**, **mean avg. precision=0.62** and **mean reciprocal rank=0.49**.

Author Keywords

Natural Language Interface; Question Answering; Semantic Interface; Semantic Search; Distributional Semantics; Linked Data; Semantic Web; Databases

ACM Classification Keywords

H.2.5 [Heterogeneous Databases]; H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval

IUI'14, February 24–27, 2014, Haifa, Israel.

ACM 978-1-4503-2184-6/14/02..$15.00.
http://dx.doi.org/10.1145/2557500.2557534

INTRODUCTION

The demand to access large amounts of heterogeneous structured data is emerging as a trend for many users and applications on the Web [1]. Google Knowledge Graph[1] is a recent example of the benefits of enabling the use of large-scale structured data resources may bring to applications. Additionally, during the last years, Linked Data emerged as a standard for publishing structured data on the Web, playing a fundamental role in enabling the next generation of applications driven by rich Web data.

However, the effort involved in querying heterogeneous and distributed third-party Linked Data sources on the Web creates barriers for data consumers. In order to query datasets, users need to discover the datasets of interest, understand the structure and vocabularies used in these datasets, and then finally formulate the query using the syntax of a structured query language (such as SPARQL or SQL). Ideally users should be able to express their information needs without being aware of the dataset vocabulary (or 'schema'), delegating the query formulation process to a query engine.

Structured query mechanisms for datasets allow *expressive queries* at the expense of *usability*: the semantic matching process is manually done by data consumers. On the other side of the usability spectrum, information retrieval (IR) approaches allow users to search using intuitive keyword-based interfaces. In this case, the high usability comes at the expense of query expressivity and effectiveness: traditional vector space (VSM) models for IR typically do not deliver expressive and semantic queries over structured data. At the core of this *usability-expressivity trade-off* is the *semantic/vocabulary gap* between the way users express their information needs and the way structured data is represented. To address the semantic gap it is necessary to provide a semantic model which supports an effective semantic matching between users' information needs and the data representation. Additionally, from an interface perspective, natural language interfaces (NLI), i.e. query interfaces where users can express their information needs using natural language, can also support users to have more freedom and efficiency for querying large and heterogeneous data sources [7].

[1] http://googleblog.blogspot.ie/2012/05/introducing-knowledge-graph-things-not.html, 2012.

This paper proposes a natural language interface (NLI) approach for Linked Data targeting a greater level of *vocabulary-independency* (VoI). To cope with the semantic matching over greater levels of *lexical* and *abstraction* variations between *query* and *data*, a *distributional semantic model* is used. The *distributional semantic model* component is complemented by a *compositional semantic model* which allows the alignment of the sequence of query terms to dataset elements, respecting syntactic constraints in the queries and in the datasets. The compositional model, which is tightly coupled with the distributional model, allows the definition of expressive query capabilities required for a NLI system. The utility of the approach can be extended to any dataset which can be transformed into an *Entity-Attribute-Value* (EAV) representation abstraction.

The contributions of this paper are: (i) a NLI approach for Linked Data (LD) based on a distributional-compositional semantic model which supports greater levels of vocabulary independency, (ii) the implementation of the distributional-compositional model as a semantic inverted index, (iii) the implementation of the proposed system as a research prototype and (iv) an extensive evaluation of the proposed approach.

The Vocabulary Problem for Linked Data

The structural, categorical and lexical elements used in the representation of a proposition (triple) in a Linked Dataset depends largely on factors such as the set of usage intents for the dataset and the individual perspective of the dataset designer. Figure 1 depicts an example of the *vocabulary gap* between a user query and alternative dataset representations for triples providing answers for the query. In (a) *'United States'* composes with *'Presidents'* to form a class, while in (b) *'United States* is an instance associated with the property *'president'*. The vocabulary information related to the query term *'daughter'* is given by the property *'child'* in (a) and *'fatherOf'* in (b). The semantic gap introduced by possible conceptualizations of the reality into a database defines the *vocabulary problem for databases*.

The vocabulary problem for databases is a fundamental and practical concern for many users and information systems, as database schemas grow in size and semantic heterogeneity [7] and the time to build structured queries grows in proportion to the schema size. Addressing the vocabulary problem for databases, however, depends on the definition of a comprehensive semantic matching approach between user information needs and structured data. However, the construction of comprehensive semantic matching solutions is largely dependent on the availability of large-scale commonsense knowledge bases.

Proposed Solution

This work introduces a distributional-compositional semantic model which is used as the central element for the construction of a vocabulary-independent Natural Language Interface (NLI) for Linked Data. Figure 1 depicts the high-level workflow behind the proposed NLI approach which maps to the outline of the paper. The first step (1) in the proposed approach is the construction of a *distributional semantic model* based on the extraction of co-occurrence patterns from large corpora, which defines a distributional semantic vector space. The distributional semantic vector space uses concept vectors to semantically represent data and queries, by mapping datasets elements and query terms to concepts in the distributional space. Once the space is built, the RDF graph data is embedded into the space (step 2), defining the $\tau - Space$, a structured distributional semantic vector space. The alignment between structured data and the distributional model allows the use of the *large-scale commonsense information* embedded in the distributional model (extracted from text) to be used in the *semantic matching/approximation* process. An introduction to distributional semantics and the construction of the $\tau - Space$ is described in section $\tau - Space$. In order to support high performance search and querying, the $\tau - Space$ is mapped into an inverted index, which is described in the section *Query Approach*.

After the data is indexed into the $\tau - Space$, it is ready to be queried. The query processing starts with the analysis of the natural language query, from which a set of *query features* and a *semi-structured query representation* is extracted (step 3). The query analysis is described in section *Query Analysis*. After the query is analyzed, a *query processing plan* is generated, which maps the set of features and the semi-structured query into a set of search, navigation and transformation *operations* (step 5) over the data graph embedded in the $\tau - Space$. These operations define the semantic matching between the query and the data, using the distributional semantic information. This corresponds to the *compositional* model associated to the distributional model. The query processing approach and the operations over the $\tau - Space$ are described in the *Query Processing* section. The approach is evaluated under a large open domain dataset in the *Evaluation* section, followed by the analysis of related work and conclusions.

DISTRIBUTIONAL SEMANTIC MODEL

Distributional Semantics

Distributional semantics is defined upon the assumption that the context surrounding a given word in a text provides important information about its meaning [5]. Distributional semantics focuses on the *automatic* construction of a semantic model based on the statistical distribution of word co-occurrence in texts, allowing the creation of an associational and quantitative model which captures the degree of semantic relatedness between words. Distributional semantic models are represented by Vector Space Models (VSMs), where the meaning of a word is represented by a weighted vector which captures the associational pattern with other words in the corpora. In this work a *distributional semantic model* is used as a core element to address the query-dataset vocabulary gap.

τ-Space

The τ-Space [3] is a *distributional structured vector space model* which allows the semantic (concept-based) indexing of labelled graphs. The distributional model under the scope of

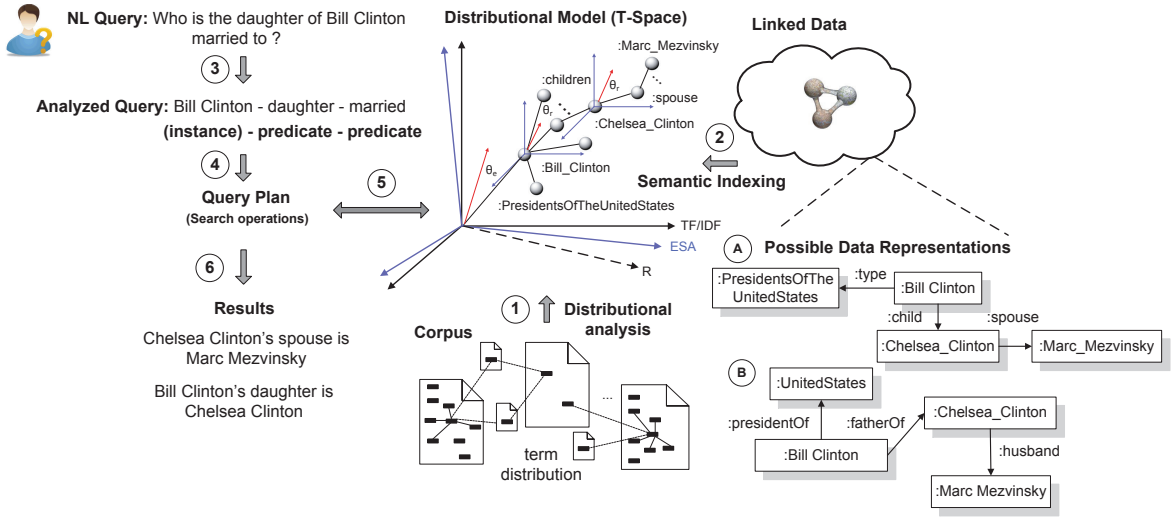

Figure 1. Example of the vocabulary gap between query and data representation.

this work is defined by the Explicit Semantic Analysis (ESA) [2] model. ESA defines a *concept coordinate basis* where the RDF(S) graph labels can be resolved into a high-dimensional concept space. Figure 1(2) depicts the $\tau - Space$ embedding an example RDF graph containing triples with instances, classes and properties. By its construction, the ESA coordinate basis can be transformed to an TF/IDF *term coordinate basis* (Figure 1) [3, 2]. Essentially, the $\tau - Space$ allows the representation of labelled graph data such as RDF with a distributional semantic grounding, using the background commonsense knowledge present in the reference corpora to support semantic search operations.

Construction

The graph data model has a signature $\Sigma = (P, E)$ formed by a pair of finite set of symbols used to represent predicates $p \in P$ and instances $e \in E$ in the graph G. It can be assumed that both elements in P and E are represented using meaningful descriptors (symbols present in the reference corpus). In this context, a predicate represent both properties (binary predicates) and classes (unary predicates). Each element in the signature Σ_G is represented as a vector in a distributional space. The semantics of G is defined by the vectors in the distributional space.

The $\tau - Space$ *coordinate system* is built from a text collection $\mathbb{C}$. The set $Term = \{k_1, \cdots, k_t\}$, of all terms available in $\mathbb{C}$ is used to define the basis $Term_{base} = \{\overrightarrow{\mathbf{k}}_1, \cdots, \overrightarrow{\mathbf{k}}_t\}$ of unit vectors that spans the *term vector space* VS^{Term}.

The set of all distributional concepts $Concept = \{c_1, \cdots, c_t\}$ are extracted from a reference corpus and each concept $c_i \in Concept$ is mapped to an identifier which represents the co-occurrence pattern in the corpus. Each identifier c_i defines a set which tracks the context where a term k_t occurred. This set is used to construct the basis $Concept_{base} = \{\overrightarrow{\mathbf{c}}_1, \cdots, \overrightarrow{\mathbf{c}}_t\}$ of vectors that spans the *distributional vector space* VS^{Dist}.

Thus, the set of contexts where a term occurs define the concept vectors associated with the term, which is a representation of its meaning based on the reference corpora. Each concept vector is weighted according to the term distribution in the corpus, allowing the concept vector space coordinate basis to be defined in terms of a term vector space coordinate basis, where each dimension maps to a word in the corpus. In order to obtain an approach that supports an approximative semantic model, the relational (labelled graph) model is linked to the distributional model, so that the distributional model could enrich and ground the semantics of the relational model.

The first step is to build the τ-*Space concept space* based on the reference corpus. The second step is to translate the elements of the signature $\Sigma = (P, E)$ of the graph G to elements VS^{Dist}. The vector representation of P, under the VS^{Dist} is defined by:

$$\overrightarrow{\mathbf{P}}_{VS^{Dist}} = \{\overrightarrow{\mathbf{p}} : \overrightarrow{\mathbf{p}} = \sum_{i=1}^{t} v_i^p \overrightarrow{\mathbf{c}}_i, \text{for each } p \in P\} \quad (1)$$

and the vector representation of E in VS^{Term} is defined by:

$$\overrightarrow{\mathbf{E}}_{VS^{Term}} = \{\overrightarrow{\mathbf{e}} : \overrightarrow{\mathbf{e}} = \sum_{i=1}^{t} w_i^e \overrightarrow{\mathbf{k}}_i, \text{for each } e \in E\} \quad (2)$$

where w_i^e and v_i^p are defined by a co-occurrence weighting scheme[2].

The third step refers to the translation of triple from G into the $\tau - Space$. As each predicate and instance term has a vector representation, we can define the vector representation of a triple r in the concept vector space by the following definition.

[2]for example, the term-frequency/inverse document frequency(TF/IDF).

Definition (Relational Vector): Let $\overrightarrow{\mathbf{p}}$, $\overrightarrow{\mathbf{e_1}}$ and $\overrightarrow{\mathbf{e_2}}$ be the vector representations, respectively, of p, e_1 and e_2. A triple vector representation (denoted by $\overrightarrow{\mathbf{r}}$) is defined by: $(\overrightarrow{\mathbf{p}} - \overrightarrow{\mathbf{e_1}})$ if $p(e_1)$; $(\overrightarrow{\mathbf{p}} - \overrightarrow{\mathbf{e_1}}, \overrightarrow{\mathbf{e_2}} - \overrightarrow{\mathbf{p}})$ if $p(e_1, e_2)$.

The vocabulary-independent query model uses the *distributional semantic relatedness search* as a native (and primitive) semantic approximation/equivalence operation between query terms and data elements in the $\tau - Space$. This primitive operation is coordinated with data navigation and transformation steps over the data graph, which defines the compositional model, i.e. progressively matches query structures to dataset structures.

VOCABULARY-INDEPENDENT QUERY APPROACH

Outline

To facilitate the understanding of the principles behind the query approach we start with the description of the query analysis and processing for the query example and then we follow to the description of the generalised approach.

For the example query *'Who is the daughter of Bill Clinton married to?'*, the query analysis starts with the *part-of-speech* (POS) tagging of the natural language query terms and by determining the *dependency structure* of the query. The query analysis consists in transforming the natural language query into a *partial ordered dependency structure* (PODS), a triple-like representation of the query associated with a set of *query features* (Figure 2).

With the PODS and the query features, the query processing approach starts by resolving the *core (pivot) entity* in the query (in this case *Bill Clinton*) to the corresponding database entity (*dbpedia: Bill_Clinton*) (Figure 2). The pivot determination is dependent on heuristics which take into account the query features and targets the element which is *less vague or ambiguous*, and consequently presents a higher probability of a correct matching (covered in the *Query Analysis* section).

After *Bill Clinton* is resolved, the subspace of the entity *dbpedia:Bill_Clinton* is selected, constraining the search space to elements associated with *dbpedia:Bill_Clinton*, and the next term in the PODS (*'daughter'*) is used as a query term for a distributional semantic search over the neighboring elements of *dbpedia:Bill_Clinton*. The distributional semantic search is equivalent to computing the *distributional semantic relatedness* between the query term (*'daughter'*) and all predicates associated with *dbpedia:Bill_Clinton* (*dbprop:religion*, *dbprop:child*, *dbprop:almaMater*, etc). The semantic equivalence between *'daughter'* and *dbprop:child* is determined by using the corpus-based distributional commonsense information (the words *'daughter'* and *'child'* occur in similar contexts). A *threshold* filters out unrelated relations. After the alignment between *'daughter'* and *dbprop:child* is done, the query processing *navigates to* the entity associated with the *dbprop:child* relation (*dbpedia:Chelsea_Clinton*) and the next query term (*'married'*) is taken. At this point the entity *dbpedia:Chelsea_Clinton* defines the search subspace (relations associated with *dbpedia:Chelsea_Clinton*) and the semantic search for predicates which are semantically related to *'married'* is done. The query term *'married'* is aligned

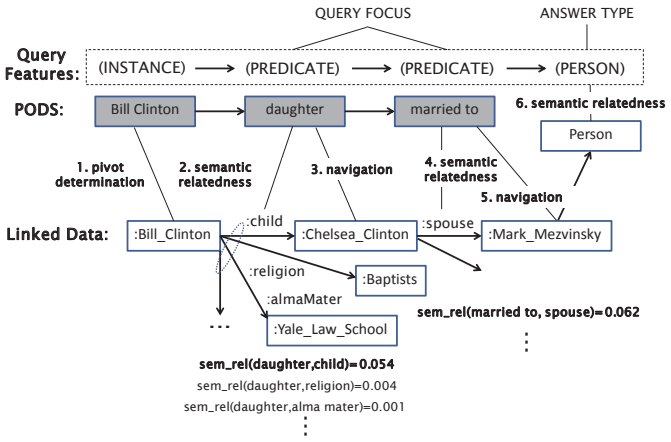

Figure 2. Query processing steps for the query example.

to *dbprop:spouse* and the answer to the query is found: the entity *dbpedia:Mark_Mezvinsky* (Figure 2).

Semantic Matching

The query processing approach has the objective of providing a mapping $m(Q, G)$ between the query terms $\langle q_0 ... q_n \rangle$ $\forall q_i \in Q$ and P, E elements in the graph G. This mapping is named an *interpretation* of the query Q under the graph G. In the interpretation process it is central to minimize the impact of *ambiguity*, *vagueness* and *synonymy*, which are central phenomena for the vocabulary problem for databases.

RDF(S) defines a semantic representation model which typically maps specific lexical categories to specific types of entities. Below we examine the relation between lexical categories and the representation model of RDF.

- **Instance:** Instances typically represent *named entities* such as people, places, organizations, events and are associated with proper nouns. Named entities are less bound to *vagueness* and *synonymy*. Compared to other entity types (property, relationship, class) it also presents a lower incidence of *ambiguity*. From a semantic matching perspective, instances are less bound to vocabulary variation (lower vocabulary gap) and as a consequence are more likely to match using string/term similarity approaches (under VS^{Term}).

- **Property, Class:** Represent predications, categories, relations and states. Typically map to nouns, adjectives, verbs and adverbs. Nouns, adjectives, verbs and adverbs are more bound to synonymy, vagueness, ambiguity. From a corpora perspective, predications tend to occurr in a larger number of contexts. From a semantic matching perspective, properties and classes are more prone to vocabulary variation (higher vocabulary gap) and are typically dependent on more sophisticated semantic matching approaches (distributional search under VS^{Dist}).

Query Analisys

The query analysis process consists in recognizing and classifying *entities* and *operations* in the query and also in mapping the natural language query into a *PODS* (triple-like format) and into a set of *query features* (Figure 2). The query analysis operations are:

Entity detection and classification (instances, classes, complex classes): The pre-processing phase starts by determining the part-of-speech (POS) of the query terms. *POS tags pattern rules* are used to determine entity candidates' types: *instances*, *classes* and *properties*. Examples of the POS tag-based entity detection and classification mapping rules are as follows: $NNP+ \rightarrow$ Instance; $\{RB * \ JJ*\}$ $NN(S)*$ $\{IN \ NNP\}* \rightarrow$ Class OR Property; $BE* \ VB$ $\{IN \ NN\}* \rightarrow$ Property; $\{CD+ \ OR \ (NN(S*)+ \ AND \ isAfterAPropertyTerm)\} \rightarrow$ Value.

Operation detection: *POS tags* and *keyword patterns* are used in the detection of operations and associated parameters in the query. While the lexical and structural variation for dataset elements is large, the vocabulary for typical database operations can be enumerated in a knowledge base of lexical expressions of operations Op.

Triple-pattern ordering: Natural language queries are parsed using a *dependency parser*. The dependency structures are reduced to a set of *partial ordered dependency structures (PODS)* by applying two sets of operations: (i) the removal of stopwords and their associated dependencies (ii) the re-ordering of the dependencies based on the core entity position in the query (where the core entity becomes the first query term and the topological relations given by the dependencies are preserved). For the example query the PODS is *Bill Clinton - daughter - married to*. The triple-pattern processing rules are described in details in [4].

Query classification: Classifies the query according to the following *query features*: (i) *queries with instances references*, (ii) *queries with classes/complex classes references*[3], (iii) *queries with operators references*, (iv) *queries with constraint composition* (*property composition*, *conjunction & disjunction operators*). The set of features represent database primitives in the *data representation level* (instances, properties, classes), *operational level* (e.g. aggregation, ordering) and *structural/compositional level* (conjuction, disjunction, property composition). The query features for the example query are shown in Figure 2.

Query Approach over the $\tau - Space$

Definition (Semantic Relatedness): A *semantic relatedness function* $sr : VS^{dist} \times VS^{dist} \rightarrow [0, 1]$ is defined as $sr(\overrightarrow{\mathbf{p_1}}, \overrightarrow{\mathbf{p_2}}) = \cos(\theta) = \overrightarrow{\mathbf{p_1}}.\overrightarrow{\mathbf{p_2}}/||\overrightarrow{\mathbf{p_1}}||||\overrightarrow{\mathbf{p_2}}||$. A threshold $\eta \in [0, 1]$ could be used to establish the semantic relatedness between the two vectors: $sr(\overrightarrow{\mathbf{p_1}}, \overrightarrow{\mathbf{p_2}}) > \eta$. The experimental threshold η is based on the semantic differential approach for ESA proposed in Freitas et al. [6].

[3]complex classes are classes which contain more than two words

The *distributional semantic relatedness measure* is used to establish an approximate semantic equivalence between query-dataset elements in the context of a query matching step. The first element to be resolved in the PODS, called *the semantic pivot*, in general is a term which represents the most specific element in the query. The semantic pivot, as the more constraining element in the query, helps to reduce the search space, since just the elements in the graph associated with the pivot at a given iteration are candidates for the semantic matching. The query sequence is embedded in the vector space VS^{dist}, allowing its identification with the following sequence of vectors $< \overrightarrow{\mathbf{q'}}_0, \overrightarrow{\mathbf{q'}}_1, \cdots, \overrightarrow{\mathbf{q'}}_n >$.

Definition: Given a query q, its instances and predicates, denoted by $q_0, q_1, ..., q_n$ are ordered in a sequence $< q'_0, q'_1, \cdots, q'_n >$ using a heuristic measure of specificity $h_{specificity}$ from the most specific to the less specific, that is, $\forall i \in [0, n], h_{specificity}(q'_i) \geq h_{specificity}(q'_{i+1})$. The specificity can be computed by taking into account lexical categories (e.g. $f_{spec}(proper\_noun) > f_{spec}(noun) > f_{spec}(adjective) > f_{spec}(adverb)$) in combination with a corpus-based measure of specificity, in this case inverse document frequency (IDF) using Wikipedia as a corpus.

In the first iteration, $\overrightarrow{\mathbf{q'}}_0 \in VS^{dist}$, the vector representation of the pivot q'_0 can be resolved to a vector $\overrightarrow{\mathbf{e}}_0$. The entity e_0 defines a vector subspace which can be explored by the next query term (which spans the relations associated with the entity e_0). The second query term q'_1 can be matched with one or more predicates associated with e_0, for example p_0, considering that the *distributional semantic relatedness measure*, $sr(\overrightarrow{\mathbf{q'}}_1, \overrightarrow{\mathbf{p}}_0) \geq \eta$, where η is a semantic relatedness threshold. The entities associated with p_0 (for example e_1) are used as new semantic pivots.

At each iteration of the querying process, a set of semantic pivots are defined and are used to navigate to other points in the vector space. This navigation corresponds to the reconciliation process between the semantic intent defined by the query and the semantic intent expressed in the dataset G. The reconciliation process can be defined as the sequence of vectors $< (\overrightarrow{\mathbf{q'}}_1 - \overrightarrow{\mathbf{p}}_1), (\overrightarrow{\mathbf{q'}}_2 - \overrightarrow{\mathbf{p}}_2), \cdots, (\overrightarrow{\mathbf{q'}}_n - \overrightarrow{\mathbf{p}}_n) >$. The proposed approximate querying process can also be represented geometrically as the vectors $< (\overrightarrow{\mathbf{e}}_0 - \overrightarrow{\mathbf{p}}_0), (\overrightarrow{\mathbf{p}}_0 - \overrightarrow{\mathbf{e}}_1), ..., (\overrightarrow{\mathbf{p}}_{n-1} - \overrightarrow{\mathbf{e}}_n) >$ over the $\tau - Space$, which geometrically represents the process of finding the answer in the graph.

Query Processing

After the *Query Analysis*, the PODS and the query features are sent to the *Query Planner*, which generates the *query processing plan*. A query processing plan involves the application of a sequence $\langle op_0, ..., op_n \rangle$ of search, navigation and transformation operations over the τ-Space. The primitive operations for the *Query Planner* are:

(i) Search operations:

Consists of keyword and distributional search operations over the graph G.

- **Instance search (VS^{Term}):** Due to its low level of vagueness, ambiguity and synonymy and typically large number of instances, the instance search approach does not use the distributional semantic model. For a query term q^I over the term space VS^{Term}, the *ranking function* $s(\mathbf{q^I}, \mathbf{i_j})$ is given by a combination of the *dice coefficient* between the returned URI label and the query term $sim_{dice}(\mathbf{q^I}, \mathbf{i_j})$, the *node cardinality* (number of associated properties) $n(i_j)$.

- **Class search (VS^{Dist}):** Differently from instances, classes are more bound to synonym, vagueness and ambiguity (larger vocabulary gap), being more sensitive to vocabulary variation. The class search operation is defined by the computation of the semantic relatedness between the class candidate query term q^C and the class entities in the $\tau - Space$: $sr(\overrightarrow{\mathbf{q^C}}, \overrightarrow{\mathbf{c}})$.

- **Property search (VS^{Dist}):** Consists in the search operation where a set of URIs define a set of subspaces associated with instances $\mathbf{i}_j$. The property candidate query term q^P is used as an input for a distributional semantic search over the relations associated with the instance subspace. The search is defined by $sr(\overrightarrow{\mathbf{q^P}}, \overrightarrow{\mathbf{p}})$.

(ii) Graph navigation & transformation operations

- **Graph navigation:** Graph navigation elements provide the core structural compositional operations for the query processing over instances (i), properties (p), classes (c) in the graph. There are three main graph navigation elements:

  1. *Navigation (Predicate composition):* Consists in a predicate composition that defines a path query. The predicate composition is determined for a path of predicates connected through a common instance. Geometrically, the predicate composition is defined by a sequence of translations in the VS^{Dist}. This operation maps to the $(i - p_0 - v_0)(v_0 - p_1 - v_1)...(v_n - p_{n+1} - v_{n+1})$ or $(i - p_0 - v_0)(v_0 - p_1 - v_1)...(v_n - rdf : type - c)$ graph patterns, where v_i represents a variable.

  2. *Extensional class expansion (instance listing for a class):* Consists in expanding the set of instances $\mathbf{i}_j$ associated with a class $\mathbf{c}$ through the *rdf:type* predicate. This operation maps to the $(c - rdf : type - v_n)$ triple pattern, where v_n defines a set of instances associated with the class c.

  3. *Star-Shaped property composition:* Consists in the composition of triple patterns in a disjunctive $(i_0 - p_0 - v) \vee (i_1 - p_1 - v) \vee ... \vee (i_n - p_n - v)$ or conjunctive form $(i_0 - p_0 - v) \wedge (i_1 - p_1 - v) \wedge ... \wedge (i_n - p_n - v)$.

  The application of the compositional constraints is done as a graph navigation over the $\tau - Space$. The query processing algorithm works as a *semantic best-effort* query system, where the algorithm tries to maximize the amount of semantic constraints which are matched, but eventually can return approximate or incomplete results. The search operations and the constraints application are done in a inverted index, aiming at performance and scalability. This approach can be contrasted with NLI approaches which try to satisfy all constraints in a single SPARQL query.

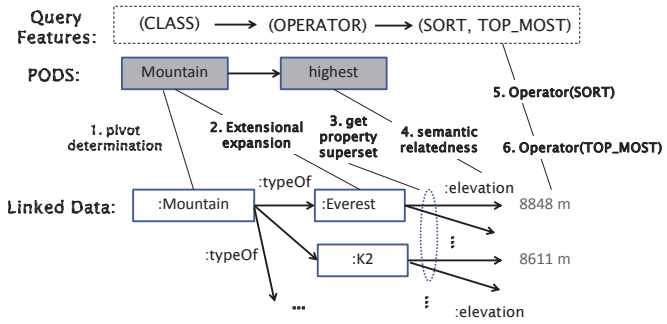

Figure 3. Execution of a query processing plan for the query '*What is the highest mountain ?*'

- **Data transformation operations:** Consists in the application of *functions* for filtering triples or mapping triples to the real domain.

  1. **Aggregation operators:** Maps a set of triples or entities from VS^{Dist} or VS^{Term} into the $\mathbb{R}$ domain, based on an enumerable set of functional operators Op (e.g. *count*, *average*, etc).
  2. **Ordering operators:** Defines a sequence for a set of triples and entities based on an ordering criteria (*ascending*, *descending*).
  3. **Conditional operators:** Filters a set of triples and entities based on a *conditional expression* (e.g. $>$, $<$, etc).
  4. **User feedback operators:** Filters a set of triples based on the user input for a set of instances, classes and properties. This operation aims at allowing users to cope with possible errors in the term and distributional search operations over the $\tau - Space$, by allowing them to select from a list matching the search criteria, a set of valid instances, classes and properties. The *user feedback dialog* define just a filtering function, where users can select from a reduced list of options (maximum 5 elements) in case there is ambiguity in the term/distributional search process.

The *query planning algorithm* (Algorithm I) orchestrates the *search & graph navigation & transformation* operations defined above.

A query plan defines multiple operations over the index. Figure 2 shows an example of a set of operations for a query with an *instance* as a pivot entity, while Figure 3 shows the execution of a query plan for a second example query ('*What is the highest mountain ?*'), which is a query with a *class* as a pivot entity.

Algorithm 1 Distributional-compositional query planning algorithm

$Q(V_Q, E_Q, Op)$: VoI query graph patterns
$G(V_G, E_G)$: Indexed LD graph
$A(V_A, E_A, P_A)$: answer graph and post-processed answer
i : set of instances URIs
c : set of classes URIs
p : related properties URIs
q : query term
$initialize(A)$
for all $q \in V_Q$ **do**
  if $(isCoreEntity(q))$ **then**
    $i \leftarrow searchInstances(q)$
    $c \leftarrow searchClasses(q)$
  end if
  if $(isAmbiguous(i, c))$ **then**
    $i, c \leftarrow disambiguatePivotEntity(i, c)$
  end if
  if $(pivotEntiyIsClass)$ **then**
    $i \leftarrow extensionalExpansion(c)$
  end if
  $p \leftarrow searchProperties(i, q)$
  if $(hasOperations(Q))$ **then**
    $p \leftarrow searchOperations(i, Op)$
  end if
  if $(isAmbiguous(p))$ **then**
    $p \leftarrow disambiguateProperty(p)$
    $triples \leftarrow selectByPivotAndProperty(i, p)$
  end if
  $i, c \leftarrow navigateTo(triples)$
  $V_A, E_A \leftarrow triples$
  $P_A \leftarrow applyOperation(triples, Op)$
end for

ARCHITECTURE

The high-level workflow and main components for the query approach are given in Figure 4. In the first phase (*query preprocessing*), the natural language query analysis process is done by the *Interpreter* component. The second phase consists in the *query processing approach* which defines a sequence of search and data transformation operations over the RDF graph embedded in the τ-Space, based on the query plan which is defined by the query features. The *Query Planner* generates the sequence of operations (the *query processing plan*) over the data graph on the semantic inverted index. The query processing plan is sent to the *Query Processor* which initialy executes the *search operations* part of the query plan over the *Distributional Search* component. The query plan also includes the application of a set of *graph navigation & transformation operations* which are implemented on the *Operators* component. The result of search operations can be disambiguated using the *Disambiguation* component for *pivot entities* and *predicates*.

Index & System Implementation

The distributional-compositional index is implemented over the *Lucene* 3.5 IR framework. The core *index* structure consists of three indexes: the *graph index* for mapping the graph topology (triples), the *entity index* (for instances and classes) and the *predicate/property index*. While *uri* stores the element URI, the field *terms/stemmed terms* covers the contents of the parsed and stemmed URIs and the *distributional concept vector* field indexes the ESA weighted concept vector. The distributional ESA concept vector is serialized as a Lucene *payload* (byte array). For elements in the entity index, only classes have a distributional field. Figure 4 depicts the distributional index structure. The index structure allows its natural parallelization: the subspaces defined by entities can serve as partition identifiers. Each entity subspace can also be partitioned and distributed given that the distributional reference corpus for each index partition is defined.

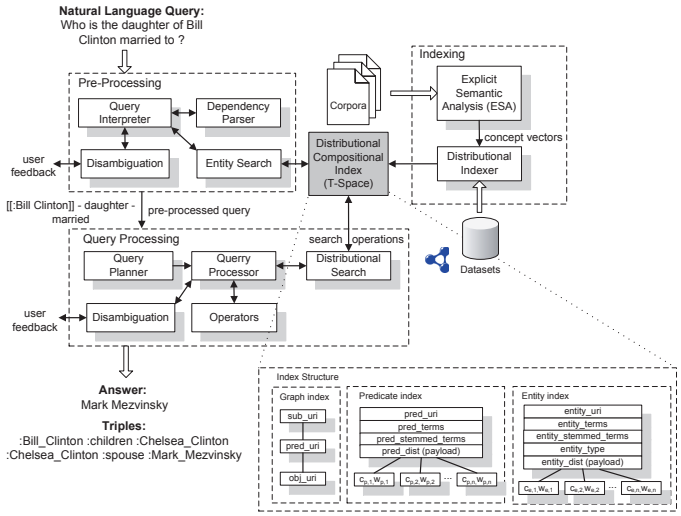

Figure 4. High-level components diagram of the vocabulary-independent query approach and distributional inverted index structure.

The query processing mechanism is implemented in the *Treo* NLI system following the components diagram (Figure 4). The system is implemented in Java. The *core* component contains the Lucene-based $\tau - Space$ implementation which can be used in other semantic search scenarios, while the *NLI* component contains the module for analyzing natural language queries. The *DS* component contains a distributional semantics infrastructure which implements ESA.

Figures 5 and 6 show the query interface for different query types. For the example queries we can observe the *semantic best-effort* characteristic of the approach, where other highly related elements are returned by the distributional matching. A video of a running prototype can be found online[4].

EVALUATION

The QA approach is evaluated under an open domain question answering over Linked Data scenario, using unconstrained natural language queries. The query processing approach was evaluated using the Question Answering over Linked Data 2011 test collection[5]. The dataset contains 102 natural language queries over DBpedia 3.7 and YAGO. The experiments were executed on an intel core i5 computer with 8GB of RAM. Table 1 shows the distribution of query features in the query set and the dataset statistics. In addition to the query features, the test collection was analysed in relation

[4] **http://bit.ly/1c36LGD**
[5] QALD-1, **http://www.sc.cit-ec.uni-bielefeld.de/qald-1**

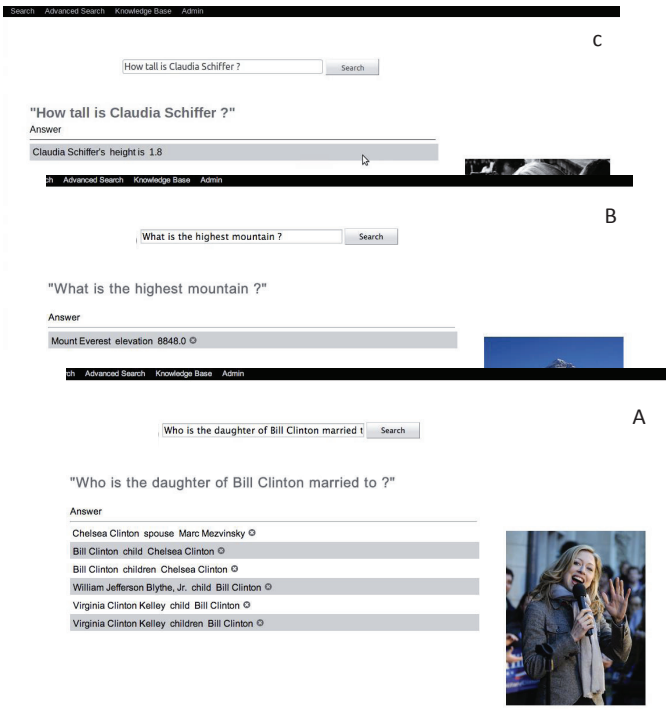

Figure 5. Screenshots of the query interface for queries returning triples.

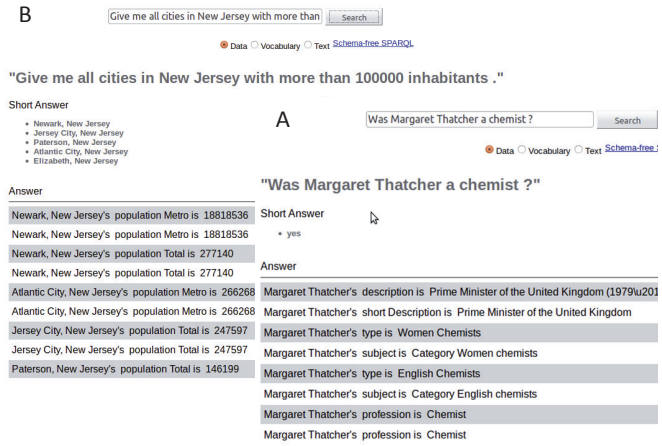

Figure 6. Screenshots of the query interface for queries returning post-processed answers.

to the presence of vocabulary gaps: 41% of the properties were aligned to different terms expressed in the queries, and only 17% of the properties had identical lexical expressions. For classes 30% had a completely different lexical expression, while 11% had an identical lexical expression. The test collection also expresses a large number of distinct query patterns. The reader is referred to [6] for the analysis of the query patterns.

The average evaluation measures are provided in the Tables 2 and 5. The first category of measurements evaluates the

[6] **http://bit.ly/1c36LGD**

Query Features	**%**
Contains instance reference	0.63
Contains class reference	0.12
Contains complex class reference	0.10
Contains operator reference	0.15
Contains constraint composition	0.84
Dataset Feature	**value**
# of predicates	45,767
# of classes	5,556,492
# of classes	9,434,677
dataset size	17GB

Table 1. Statistics for the test collection features.

Measure Type	**Value**
Mean Avg. Precision	0.62
Avg. Recall	0.81
Avg. F-Measure	0.70
Avg. MRR	0.49
% of queries answered	80%

Table 2. Relevance analysis for the QALD 2011 queries.

answer relevance using *mean avg. precision*, *avg. recall*, *mean reciprocal rank* (mrr) and the *% of answered queries* (fully and partially (recall $>$ 0.20) answered). **80%** of the queries were answered using the distributional-compositional index. The **0.81** recall confirms the hypothesis that the distributional-compositional search provides a comprehensive semantic matching mechanism. The mean avg. precision=**0.62** and mrr=**0.49** confirms the hypothesis that the approach provides an effective approximate (semantic best-effort) NLI mechanism, also returning a limited list of unrelated results. The approach was tested under different combinations of query patterns (14 based on entity types, 57 distinct patterns based on lexical categories), showing a medium-high coverage in terms of **query expressivity**.

Table 3 shows the comparison between the distributional approach with three baseline systems. The system outperforms the existing approaches in recall and % of answered queries, showing equivalent precision to the best performing system. Analyzing the query features related to the queries with f-measure $<$ 0.1 it is possible to observe that most of the queries which were not answered by PowerAqua have aggregations and comparisons (53% - 9 queries) and/or reference to classes (70% - 12 queries). For Freya, the same pattern could be observed: queries with aggregations and comparisons account for 50% (7 queries) of the queries with f-measure $<$ 0.1, while queries with reference to classes account for 64% (9 queries). Comparatively, the proposed approach is able to cope with queries containing references to aggregations and comparisons and reference to classes (accounting for 40% on the queries which were not answered - 2 queries). This difference can be explained by the construction of a comprehensive query planning algorithm, which provides a mechanism to detect core query features and map them into a query execution plan (which in the context of this work, defines the compositional/interpretation model).

The second category of measurements in Table 5 evaluates individually the core search components of the approach: *instance/class(pivot) search* and *predicate search*. Queries with instances as semantic pivots have higher precision and recall

System	Avg. R	MAP	% answered queries
Treo	**0.79**	**0.63**	**79%**
PowerAqua	0.54	0.63	48%
FREyA	0.48	0.52	54%
Unger et al.	0.63	0.61	-

Table 3. Comparison with existing systems for the QALD 2011 test subset.

Measure	value
Avg. query execution time (ms)	8,530
Avg. entity search time (ms)	3,495
Avg. predicate search time (ms)	3,223
Avg. number of search operations per query	2.70
Avg. index insert time per triple (ms)	5.35
Avg. index size per triple (bytes)	250
Dataset adaptation effort (minutes)	0.00
Dataset specific semantic enrichment effort per query (secs)	0.00
Dataset specific semantic enrichment effort (minutes)	0.00

Table 4. Temporal and size measures of the distributional-compositional semantic index for the NLI approach.

compared with queries with classes pivots. The individual performance of these two components minimizes the number of *user-feedback* for disambiguation over the semantic inverted index. To support an effective user feedback dialog mechanism, the set of returned results should have high mrr and precision. From a user-interaction perspective, an average mrr higher than 0.33 (where the target result is ranked third on the list) provides a low impact disambiguation mechanism. The measured average mrr=**0.91** for *entity search* and **0.76** for *predicate search* components provide a low interaction cost disambiguation mechanism. Both entity and predicate search have a high recall value (**1.0** and **0.95** respectively). Compared with queries with instances as pivots, queries containing classes as pivots have a significantly higher number of entity disambiguation operations. The evaluation shows that the semantic matching copes with the *ability to handle lexical variation* (including non-taxonomic matchings and alignments from different POS). Most queries do not require user disambiguation. The average number of user disambiguation operations per query is **0.14** for entities and **0.05** for predicates. The system can stand on its own query processing approach, without the disambiguation functionality. Comparatively, Freya [9] has an average of 3.65 user feedback operations per query, while the proposed approach has 0.20 disambiguation operations per query.

In addition to the evaluation of the results quality, the index is evaluated in relation to its temporal performance and size (Table 4). The **8,530 ms** average query execution time supports an interactive query mechanism. This average value for the query execution time is increased by the influence of a small number of queries which have large answer sets. For queries with small answer sets, the average query execution time is less than 2,000 ms. Additionally, the approach supports a *minimum dataset adaptation effort*, neither requiring data transformations nor a dataset specific manual semantic enrichment.

RELATED WORK

PowerAqua [8] is a question answering (QA) system which uses PowerMap, a hybrid matching algorithm comprising terminology-level and structural schema matching techniques with the assistance of large scale ontological or lexical resources. In addition to the ontology structure, PowerMap uses WordNet-based similarity approaches as a semantic approximation strategy. Unger et al. [11] presents a QA approach that relies on a deep linguistic analysis which generates a SPARQL template with slots that are filled with URIs. In order to fill these slots, potential entities are identified using string similarity and natural language patterns extracted from structured data and text documents. The final result is given by a ranking of the remaining query candidates. Exploring user interaction techniques, FREyA [9] is a QA system which employs feedback and clarification dialogs to resolve ambiguities and improve the domain lexicon with the help of users. User feedback is used to enrich the semantic matching process by allowing manual query-vocabulary mappings. Yahya et al. [12] describes an approach for translating natural language questions into structured SPARQL queries. The approach uses an integer linear program to coordinate the solution of various disambiguation tasks jointly, including the segmentation of questions into phrases, the mapping of phrases to entities and the construction of SPARQL triple patterns. In the evaluation of NLI/QA systems, usually the effort involved in the adaptation, in the semantic enrichment of the dataset and the user interaction in the question-answering process is not measured, bringing additional barriers to the comparability of the approaches. Additionally, temporal performance measurements are not prioritized. This work addresses these methodological issues.

Herzig & Tran [10] propose an approach for searching heterogeneous Web datasets using a structured seed query that matches to the vocabulary of one of the datasets. They introduce the entity relevance model which is used for matching and ranking results from external datasets and for performing data integration on the fly. Novacek et al. [13] describe a distributional approach applied to Semantic Web Data targeting the description of a tensor-based model for RDF data and its evaluation on entity consolidation. Freitas et al. [3] propose an initial analysis of a distributional structured semantic space (τ-Space). The work presented in [3] had the following limitations: (i) low query expressivity - focus on IR instead of QA, (ii) lack of a more extensive evaluation, (iii) no extensive scalability/performance evaluation - lack of a robust implementation of an inverted index.

Comparatively, this work focuses on improving query expressivity while keeping query flexibility, by introducing a compositional model based on the analysis of query features. The compositional model is used to define the query planning algorithm over the distributional vector space model (which supports a flexible semantic matching mechanism). A comparative analysis with existing QA systems over Linked Data shows the improvement of query expressivity reflected by the introduction of the compositional model. Another relevant characteristic of the approach (compared to existing approaches) is the fact that it addresses each of the structural

Type	Measure	all queries	w/ instances	w/ classes	w/ complex classes	w/ operations	w/ const. comp.
Query Processing	Mean Avg. Precision	**0.62**	0.65	0.77	0.46	0.88	0.63
	Avg. Recall	**0.81**	0.93	0.76	0.67	1.00	0.87
	MRR	**0.49**	0.59	0.44	0.19	0.92	0.56
	% of queries answered	**0.80**	0.94	0.80	1.00	0.75	0.82
	% of queries fully answered	**0.62**	0.81	0.40	0.30	0.75	0.70
	% of queries partially answered	**0.21**	0.13	0.40	0.70	0.00	0.12
Entity Search	Avg. Entity Precision	0.47	0.49	0.56	0.27	0.36	0.49
	Avg. Entity Recall	1.00	1.00	1.00	1.00	1.00	1.00
	Entity MRR	0.91	0.96	0.73	0.82	1.00	0.90
	% of entity queries fully answered	0.88	1.00	1.00	1.00	0.75	0.88
	Avg. # of entity disamb. operations per query	0.14	0.06	0.40	0.30	0.25	0.12
Predicate Search	Avg. Predicate Precision	0.45	0.36	0.18	0.52	0.43	0.42
	Avg. Predicate Recall	0.95	0.98	0.67	1.00	1.00	0.95
	Predicate MRR	0.76	0.81	0.30	0.40	0.71	0.83
	% of predicate queries fully answered	0.65	0.90	0.60	0.00	0.75	0.74
	Avg. # of predicate disamb. operations per query	0.05	0.06	0.20	0.00	0.25	0.05

Table 5. Evaluation of the query processing mechanism results using natural language queries. Measures are collected for the full query mechanism and its core subcomponents: entity search and predicate search. The measures are categorized according to the query features.

query constraints at a time (instead of generating a single SPARQL query), supporting a semantic approximation process. Compared with FREyA, the proposed approach relies 10x less on user feedback, and it can be used without user feedback. The construction of a semantic inverted index supports an interactive query execution time. The use of a distributional semantic model supports a low maintenance semantic matching mechanism, which can be automatically built from corpora, with higher vocabulary coverage. The independency of manually created linguistic resources or rich ontologies brings the potential for higher transportability across other languages or domains.

CONCLUSIONS & FUTURE WORK

This paper proposes and evaluates the suitability of the $\tau - Space$ distributional-compositional model applied to the construction of a question answering system for Linked Data. The contributions of this work are: (i) the definition of a NLI approach for Linked Data based on a distributional-compositional VSM, focusing on an additional level of vocabulary independency, (ii) the formulation and implementation of the distributional-compositional model as a semantic inverted index, and (iii) an extensive evaluation of the proposed index and query processing mechanism. The proposed approach was evaluated using the QALD 2011 dataset over DBpedia achieving an **avg. recall= 0.81**, **mean avg. precision=0.62** and **mrr=0.49**, outperforming existing systems in recall and query coverage. The final distributional-compositional semantic model is defined by a set of operations over a vector space model which preserves the dataset structure at the same time that supports semantically approximate queries. Future work will concentrate on the investigation of the approach under domain specific scenarios and on the verification of the impact of the use of distributional models with more constraining context windows.

Acknowledgments.
This work has been funded by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2).

REFERENCES

1. Heath, T. and Bizer, C. Linked Data: Evolving the Web into a Global Data Space (1st edition). Synthesis Lectures on the Semantic Web, (2011), 1-136.
2. Gabrilovich, E. and Markovitch, S. Computing semantic relatedness using Wikipedia-based explicit semantic analysis, in *Proc. Intl. Joint Conf. On Artificial Intelligence*, (2007), pp. 1606-1611.
3. Freitas, A., Curry, E., Oliveira, J. G., O'Riain, S. A Distributional Structured Semantic Space for Querying RDF Graph Data. *Intl. Journal of Semantic Computing (IJSC)*, (2012), vol. 5, no. 4, pp. 433-462.
4. Freitas, A., Oliveira, J.G., O'Riain, S., Curry, E. and Pereira da Silva, J.C. Querying Linked Data using Semantic Relatedness: A Vocabulary Independent Approach, in *Proc. of the 16th Intl. Conf. on Applications of Natural Language to Information Systems*, NLDB, (2011), vol. 6716, pp. 40-51.
5. Turney, P. D. and Pantel, P. From Frequency to Meaning: Vector Space Models of Semantics, *Journal of Artificial Intelligence Research*, (2010), vol. 37, pp. 141-188.
6. Freitas, A., Curry, E. and O'Riain, S. A Distributional Approach for Terminological Semantic Search on the Linked Data Web, in *Proc. of the 27th ACM Symposium On Applied Computing (SAC)*, (2012).
7. Freitas, A., Curry, E., Oliveira, J. G. and O'Riain, S. Querying Heterogeneous Datasets on the Linked Data Web: Challenges, Approaches and Trends. *IEEE Internet Computing, Special Issue on Internet-Scale Data*, (2012).
8. Lopez, V., Motta, E., Uren, V. PowerAqua: Fishing the Semantic Web. The Semantic Web: Research and Applications, Lecture Notes in Computer Science (2006), vol. 4011, pp. 393-410.
9. Damljanovic, D., Agatonovic, M. and Cunningham, H. FREyA: An Interactive Way of Querying Linked Data Using Natural Language, in *Proc. 1st Workshop on Question Answering over Linked Data*,(ESWC), (2011).
10. Herzig, D. M. and Tran, T. Heterogeneous Web Data Search Using Relevance-based On The Fly Data Integration, in *Proc. of 21st Intl. World Wide Web Conference* (WWW), (2012).
11. Unger, C., Bühmann, L., Lehmann, J., Ngonga Ngomo, A.-C., Gerber, D. and Cimiano, P. Template-based Question Answering over RDF Data, in *Proc. of 21st Int. World Wide Web Conference*, (2012), pp. 639-648.
12. Yahya, M., Berberich, K., Elbassuoni, S., Ramanath, M., Tresp, V. and Weikum, G. Natural Language Questions for the Web of Data, EMNLP, (2012), pp. 379-390.
13. Novacek, V. Handschuh, S. and Decker, S. Getting the Meaning Right: A Complementary Distributional Layer for the Web Semantics, in *Proc. of the Intl. Semantic Web Conference*, (2011), pp. 504-519.