Question Answering over Linked Data Using First-order Logic[*]

Shizhu He, Kang Liu, Yuanzhe Zhang, Liheng Xu and Jun Zhao
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China
{shizhu.he, kliu, yzzhang, lhxu, jzhao}@nlpr.ia.ac.cn

Abstract

Question Answering over Linked Data (QALD) aims to evaluate a question answering system over structured data, the key objective of which is to translate questions posed using natural language into structured queries. This technique can help common users to directly access open-structured knowledge on the Web and, accordingly, has attracted much attention. To this end, we propose a novel method using first-order logic. We formulate the knowledge for resolving the ambiguities in the main three steps of QALD (phrase detection, phrase-to-semantic-item mapping and semantic item grouping) as first-order logic clauses in a Markov Logic Network. All clauses can then produce interacted effects in a unified framework and can jointly resolve all ambiguities. Moreover, our method adopts a pattern-learning strategy for semantic item grouping. In this way, our method can cover more text expressions and answer more questions than previous methods using manually designed patterns. The experimental results using open benchmarks demonstrate the effectiveness of the proposed method.

1 Introduction

With the rapid development of the Web of Data, many RDF datasets have been published as Linked Data (Bizer et al., 2009), such as DBpedia (Auer et al., 2007), Freebase (Bollacker et al., 2008) and YAGO (Suchanek et al., 2007). The growing amount of Linked Data contains a wealth of knowledge, including entities, classes and relations. Moreover, these linked data usually have complex structures and are highly heterogeneous. As a result, there are gaps for users regarding access. Although a few experts can write queries using structured languages (such as SPARQL) based on their needs, this skill cannot be easily utilized by common users (Christina and Freitas, 2014). Thus, providing user-friendly, simple interfaces to access these linked data becomes increasingly more urgent.

Because of this, question answering over linked data (QALD) (Walter et al., 2012) has recently received much interest, and most studies on this topic have focused on translating natural language questions into structured queries (Freitas and Curry, 2014; Yahya et al., 2012; Unger et al., 2012; Shekarpour et al., 2013; Yahya et al., 2013; Bao et al., 2014; Zou et al., 2014). For example, with respect to the question

> *"Which software has been developed by organizations founded in California, USA?"*,

the aim is to automatically convert this utterance into an SPARQL query that contains the following *subject-property-object (SPO)* triple format: **⟨?url rdf:type dbo:Software, ?url dbo:developer ?x1, ?x1 rdf:type dbo:Company, ?x1 dbo:foundationPlace dbr:California⟩**[1].

To fulfill this objective, existing systems (Lopez et al., 2006; Unger et al., 2012; Yahya et al., 2012; Zou et al., 2014) usually adopt a pipeline framework that contains four major steps: 1) decomposing the question and detecting phrases, 2) mapping the detected phrases into semantic items of Linked Data, 3) grouping the mapped semantic items into semantic triples, and 4) generating the correct SPARQL query.

However, completing these four steps and constructing such a structured query is not easy. The first three steps mentioned above are subject to the

[*] Shizhu He and Kang Liu have equal contribution to this work.

[1] The prefixes in semantic items indicate the source of their vocabularies.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1092–1103,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

problem of ambiguity, which is the major challenge in QALD. Using the question mentioned above as an example, we can choose *California* or *California, USA* when detecting phrases, the phrase *California* can be mapped to the entity **California_State** or **California_Film**, and the class **Software** (mapped from the phrase *software*) can be matched with the first argument of the relation **producer** or **developer** (these two relations can be mapped from the phrase *developed*). Previous methods (Lopez et al., 2006; Lehmann et al., 2012; Freitas and Curry, 2014) have usually performed disambiguation at each step only, and the subsequent step was performed based on the disambiguation results in the previous step(s). However, we argue that the three steps mentioned above have mutual effects. In the previous example, the phrase *founded in (verb)* can be mapped to the entities (**Founding_of_Rome** and **Founder_(company)**), classes (**Company** and **Department**) or relations (**foundedBy** and **foundationPlace**). If we know that the phrase *California* can refer to the entity **California_State**, and which can be the second argument of the relation **foundationPlace**, together with a *verb* phrase being more likely to be mapped to *Relation*, we should map the phrase *founded in* to **foundationPlace** in this question. **Thus, we aim to determine if joint disambiguation is better than individual disambiguation.** (**Question One**)

In addition, previous systems usually employed manually designed patterns to extract predicate-argument structures that are used to guide the disambiguation process in the three steps mentioned above (Yahya et al., 2012; Unger et al., 2012; Zou et al., 2014). For example, (Yahya et al., 2012) used only three dependency patterns to group the mapped semantic items into semantic triples. Nevertheless, these three manually designed patterns miss many cases because of the diversity of the question expressions. We gathered statistics on 144 questions and found that the macro-average F1 and micro-average F1 of the three patterns[2] used in (Yahya et al., 2012) are only 62.8 and 66.2%, respectively. Furthermore, these specially designed patterns may not be valid with variations in domains or languages. **Therefore, another important question arises: can we automatically learn rules or patterns to achieve the same objective?** (**Question Two**)

Focusing on the two problems mentioned above, this paper proposes a novel algorithm based on a learning framework, Markov Logic Networks (MLNs) (Richardson and Domingos, 2006), to learn a joint model for constructing structured queries from natural language utterances. MLN is a statistical relational learning framework that combines first-order logic and Markov networks. The appealing property of MLN is that it is readily interpretable by humans and that it is a natural framework for performing joint learning. We formulate the knowledge for resolving the ambiguities in the main three steps of QALD (phrase detection, phrase-to-semantic-item mapping and semantic item grouping) as first-order logic clauses in an MLN. In the framework of MLN, all clauses will produce interacted effects that jointly resolve all problems into a unified process. In this way, the result in each step can be globally optimized. Moreover, in contrast to previous methods, we adopt a learning strategy to automatically learn the patterns for semantic item grouping. We design several meta patterns as opposed to the specific patterns. In addition, these meta patterns are formulated as the first-order logic formulas in the MLN. The specific patterns can be generated by these meta patterns based on the training data. The model will learn the weights of each clause to determine the most effective patterns for semantic triple construction. In this way, with little effort, our approach can cover more semantic expressions and answer more questions than previous methods, which depend on manually designed patterns.

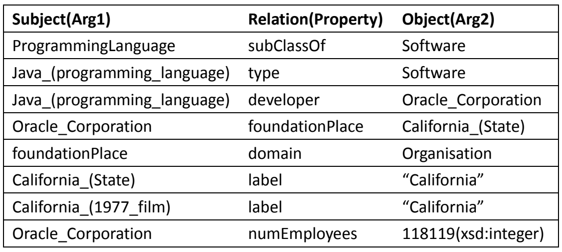
We evaluate the proposed method using several benchmarks (QALD-1, QALD-3, QALD-4). The experimental results demonstrate the advantage of the joint disambiguation process mentioned above. They also prove that our approach, employing MLN to automatically learn the patterns of semantic triple grouping, is effective. Our system can answer more questions and obtain better performance than the traditional methods based on manually designed heuristic rules.

2 Background

2.1 Linked Data Sources

Linked Data consist of many relational data, which are usually inter-linked as subject-property-object (*SPO*) triple statements (such as using the **owl:sameAs** relation). In this paper, we mainly use

[2] They are 1) verbs and their arguments, 2) adjectives and their arguments and 3) propositionally modified tokens and objects of prepositions.



Subject(Arg1)	Relation(Property)	Object(Arg2)
ProgrammingLanguage	subClassOf	Software
Java_(programming_language)	type	Software
Java_(programming_language)	developer	Oracle_Corporation
Oracle_Corporation	foundationPlace	California_(State)
foundationPlace	domain	Organisation
California_(State)	label	"California"
California_(1977_film)	label	"California"
Oracle_Corporation	numEmployees	118119(xsd:integer)

Figure 1: Sample knowledge base facts.

DBpedia[3] and some classes from Yago[4]. These knowledge bases (*KBs*) are composed of many ontological and instance statements, and all statements are expressed by *SPO* triple facts. Figure 1 shows some triple fact samples from DBpedia.

Each fact is composed of three **semantic items**. A semantic item can be an entity (**California_(State)**, **Oracle_Corporation**, etc.), a class (**Software**, **Organisation**, etc.) or a relation (called a property or predicate in some occasions). Some entities are literals including strings, numbers and dates (**118119(xsd:integer)**, etc.). Relations contain standard Semantic Web relations (**subClassOf**, **type**, **domain** and **label**) and ontological relations (**developer**, **foundationPlace** and **numEmployees**).

2.2 Task Statement

Given a knowledge base (*KB*), our objective is to translate a natural language question q_{NL} into a formal language query q_{FL} that targets the semantic vocabularies given by the *KB*, and the query q_{FL} should capture the user information needs expressed by q_{NL}.

Following (Yahya et al., 2012), we focus on the factoid questions, and the answers to such questions are an entity or a set of entities. We ignore the questions that need the aggregation[5] (max/min, etc.) and negation operations. That is, we generate queries that consist of a plentiful number of triple patterns, which are multiple conjunctions of *SPO* search conditions.

3 Framework

Figure 2 shows the entire framework of our system for translating a question into a formal SPARQL query. The first three steps address the input question through 1) Phrase Detection (detecting possible phrases), 2) Phrase Mapping (mapping all phrase candidates to the corresponding semantic items), and 3) Feature Extraction (extracting the linguistic features and semantic item features from the question and the Linked Data, respectively). As a result, a space of candidates is constructed, including possible phrases, mapped semantic items and the possible argument match relations among them. Next, the fourth step (Inference) formulates the joint disambiguation as a generalized inference task. We employ rich features and constraints (including hard and soft constraints) to infer a joint decision through an MLN. Finally, with the inference results, we can construct a semantic item query graph and generate an executable SPARQL query. In the following subsection, we demonstrate each step in detail.

1) Phrase detection. In this step, we detect phrases (sequences of tokens) that probably indicate semantic items in the *KB*. We do not use a named entity recognizer (NER) because of its low coverage. We perform testing on two commonly used question corpora, QALD-3 and free917[6], using the Stanford NER tool[7]. The results demonstrate that only 51.5 and 23.8% of the NEs are correctly recognized, respectively. To avoid missing useful phrases, we retain all n-grams as phrase candidates, and then use some rules to filter them. The rules include the following: the span length must be less than 4 (accepting that all contiguous tokens are capitalizations), the POS tag of the start token must be *jj*, *nn*, *rb* and *vb*, all contiguous capitalization tokens must not be split, etc. For instance, *software*, *developed by*, *organizations*, *founded in* and *California* are detected in the example of the first section.

2) Phrase mapping. After the phrases are detected, each phrase can be mapped to the corresponding semantic item in *KB* (entity, class and relation). For example, *software* is mapped to **dbo:Software**, **dbo:developer**, etc., and *California* is mapped to **dbr:California**, **dbr:California_(wine)**, etc. For different types of semantic items, we use different techniques. For mapping phrases to entities, considering that the entities in DBpedia and Wikipedia are consistent, we employ anchor, redirection and disambiguation information from Wikipedia. For mapping phrases to classes, considering that classes have lexical variation, especially synonyms, e.g., **dbo:Film** can be mapped

[3] http://dbpedia.org/

[4] http://www.mpi-inf.mpg.de/yago-naga/yago/

[5] We can address the *count* query questions, which will be explained in Section 3.

[6] http://www.cis.temple.edu/∼yates/open-sem-parsing/index.html

[7] http://nlp.stanford.edu/software/CRF-NER.shtml

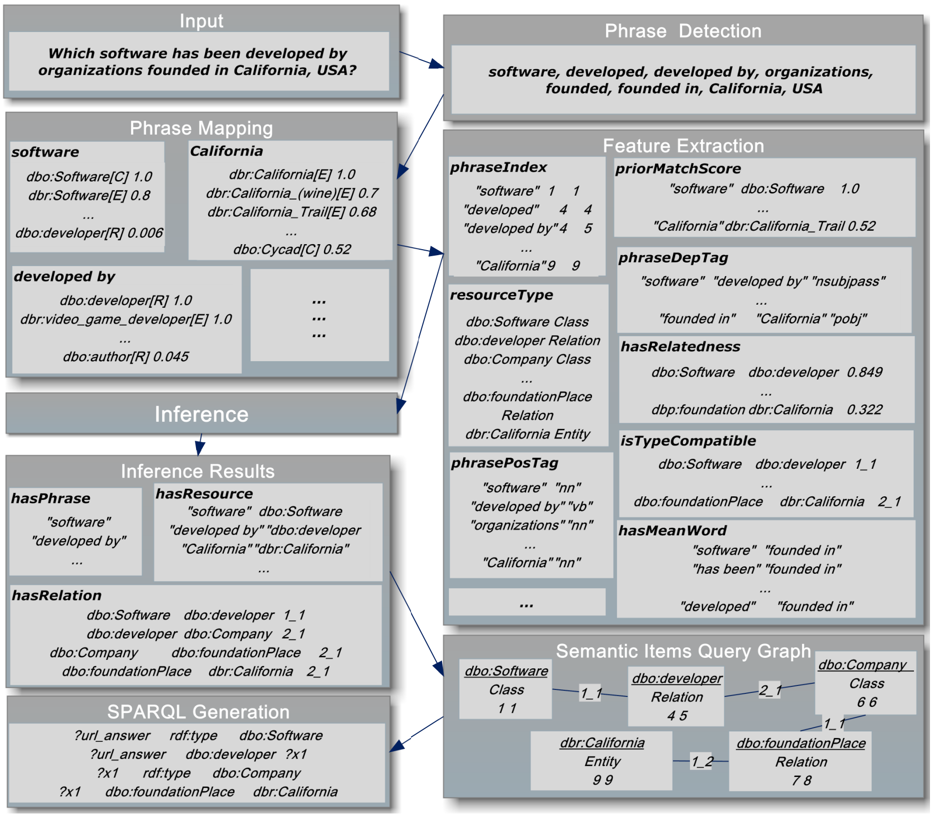

Input
Which software has been developed by organizations founded in California, USA?
Phrase Detection
software, developed, developed by, organizations, founded, founded in, California, USA
Phrase Mapping
software
dbo:Software[C] 1.0
dbr:Software[E] 0.8
...
dbo:developer[R] 0.006
California
dbr:California[E] 1.0
dbr:California_(wine)[E] 0.7
dbr:California_Trail[E] 0.68
...
dbo:Cycad[C] 0.52
developed by
dbo:developer[R] 1.0
dbr:video_game_developer[E] 1.0
...
dbo:author[R] 0.045
...
...
...
Feature Extraction
phraseIndex
"software" 1 1
"developed" 4 4
"developed by" 4 5
...
"California" 9 9
priorMatchScore
"software" dbo:Software 1.0
...
"California" dbr:California_Trail 0.52
phraseDepTag
"software" "developed by" "nsubjpass"
...
"founded in" "California" "pobj"
resourceType
dbo:Software Class
dbo:developer Relation
dbo:Company Class
...
dbo:foundationPlace Relation
dbr:California Entity
hasRelatedness
dbo:Software dbo:developer 0.849
...
dbp:foundation dbr:California 0.322
isTypeCompatible
dbo:Software dbo:developer 1_1
...
dbo:foundationPlace dbr:California 2_1
phrasePosTag
"software" "nn"
"developed by" "vb"
"organizations" "nn"
...
"California" "nn"
hasMeanWord
"software" "founded in"
"has been" "founded in"
...
"developed" "founded in"
...
Inference
Inference Results
hasPhrase
"software"
"developed by"
...
hasResource
"software" dbo:Software
"developed by" "dbo:developer
"California" "dbr:California"
...
hasRelation
dbo:Software dbo:developer 1_1
dbo:developer dbo:Company 2_1
dbo:Company dbo:foundationPlace 2_1
dbo:foundationPlace dbr:California 2_1
Semantic Items Query Graph
dbo:Software Class 1 1
1_1
dbo:developer Relation 4 5
2_1
dbo:Company Class 6 6
1_1
dbr:California Entity 9 9
1_2
dbo:foundationPlace Relation 7 8
SPARQL Generation
?url_answer rdf:type dbo:Software
?url_answer dbo:developer ?x1
?x1 rdf:type dbo:Company
?x1 dbo:foundationPlace dbr:California

Figure 2: Framework of our system.

from *film*, *movie* and *show*, we compute the similarity between the phrase and the class in the *KB* with the *word2vec* tool[8]. The *word2vec* tool computes fixed-length vector representations of words with a recurrent-neural-network based language model (Mikolov et al., 2010). The similarity scoring methods are introduced in Section 4.2. Then, the top-N most similar classes for each phrase are returned. For mapping phrases to relations, we employ the resources from PATTY (Nakashole et al., 2012) and ReVerb (Fader et al., 2011). Specifically, we first compute the associations between the ontological relations in DBpedia and the relation patterns in PATTY and ReVerb through instance alignments as in (Berant et al., 2013). Next, if a detected phrase is matched to some relation pattern, the corresponding ontological relations in DBpedia will be returned as a candidate. This step only generates candidates for every possible mapping, and the decision of the best selection will be performed in the next step.

3) Feature extraction and joint inference. There exist ambiguities in phrase detection and in mapping phrases to semantic items. This step focuses on addressing these ambiguities and determining the argument match relations among the mapped semantic items. This is the core component of our system, and it performs disambiguation in a unified manner. First, feature extraction is performed to prepare a rich number of features from the input question and from the *KB*. Next, the disambiguation is performed in a joint fashion with a Markov Logic Network. Detailed information will be presented in Section 4.

4) Semantic item query graph construction. Based on the inference results, we construct a query graph. The vertices contain the following: the detected phrase, the token span indexes of the phrases, the mapped semantic items and their types. The edge indicates the argument match relation between two semantic items. For example, we use *1_2* to indicate that the first argument of an item matches the second argument of another item[9]. The right bottom in Figure 2 shows an example of this.

5) Query generation. The SPARQL queries require the grouped triples of semantic items. Thus, in this step, we convert a query graph into multiple joined semantic triples. Three interconnected semantic items, whereby it must

[8] https://code.google.com/p/word2vec/

[9] The other marks will be introduced in Section 4.2.

be ensured that the middle item is a ***relation***, are converted into a semantic triple (multiple joined facts containing variables). For example, the query graph ⟦**dbo:Book[Class]** $\underset{\longleftrightarrow}{1\_2}$ **dbo:author[Relation]** $\underset{\longleftrightarrow}{1\_1}$ **dbr:Danielle_Steel[Entity]**⟧ is converted into ⟨**?x rdf:type dbo:Book, dbr:Danielle dbo:author ?x**⟩, and ⟦**dbo:populationTotal[Relation]** $\underset{\longleftrightarrow}{1\_2}$ **dbo:capital[Relation]** $\underset{\longleftrightarrow}{1\_1}$ **dbr:Australia[Entity]**⟧[10] is converted into ⟨**?x1 dbo:populationTotal ?answer, ?x1 dbo:capital dbr:Australia**⟩. If the query graph only contains one vertex that indicates a class ***ClassURI***, we generate ⟨**?x rdf:type ClassURI**⟩. If the query graph only contains two connected vertexes, we append a variable to bind the missing match argument of the semantic item.

The final SPARQL query is constructed by joining the semantic item triples based on the corresponding SPARQL template. We divide the questions into three types: **Yes/No**, **Normal** and **Number**. Yes/No questions use the ***ASK WHERE*** template. Normal questions use the ***SELECT ?url WHERE*** template. Number questions first use the normal question template, and if they cannot obtain a correct answer (a valid numeric value), we use the ***SELECT COUNT(?url) WHERE*** template to generate a query again. For instance, we construct the SPARQL query ***SELECT(?url) WHERE***{ ***?url rdf:type dbo:Software. ?url dbo:developer ?x1. ?x1 rdf:type dbo:Company. ?x1 dbo:foundationPlace dbr:California.***} for this example.

4 Joint Disambiguation with MLN

In this section, we present our method for question answering over linked data using a Markov Logic Network (MLN). In the following subsections, we first briefly describe the MLN. Then, we present the predicates and the first-order logic formulas used in the model.

4.1 Markov Logic Networks

Markov logic networks combine Markov networks with first-order logic in a probabilistic framework (Richardson and Domingos, 2006). An MLN $\mathcal{M}$ consists of several weighted formulas $\{(\phi_i, w_i)\}_i$, where ϕ_i is a first order formula and w_i is the penalty (the formula's weight). In contrast to the first-order logic, whereby a formula represents a hard constraint, these logic formulas are relaxed and can be violated with penalties in the MLN. Each formula ϕ_i consists of a set of first-order predicates, logical connectors and variables. These weighted formulas define a probability distribution over a possible world. Let $\mathbf{y}$ denote a possible world. Then $p(\mathbf{y})$ is defined as follows:

$$p(\mathbf{y}) = \frac{1}{Z} exp \left(\sum_{(\phi_i, w_i) \in \mathcal{M}} w_i \sum_{\mathbf{c} \in C^{n_{\phi_i}}} f_{\mathbf{c}}^{\phi_i}(\mathbf{y}) \right),$$

where each c is a binding of the free variables in ϕ_i to constants; $f_{\mathbf{c}}^{\phi_i}$ is a binary feature function that returns 1 if the ground formula that we obtain through replacing the free variables in ϕ_i with the constants in $\mathbf{c}$ under the given possible world $\mathbf{y}$ is true and is 0 otherwise; and $C^{n_{\phi_i}}$ is the set of all possible bindings for the free variables in ϕ_i. Z is a normalized constant. The Markov network corresponds to this distribution, where nodes represent ground atoms and factors represent ground formulas.

4.2 Predicates

In the MLN, we design several predicates to resolve the ambiguities in phrase detection, mapping phrases to semantic items and semantic item grouping. Specifically, we design a hidden predicate *hasPhrase(i)* to indicate that the *i*-th candidate phrase has been chosen. The predicate *hasResource(i,j)* indicates that the *i*-th phrase is mapped to the *j*-th semantic item. The predicate *hasRelation(j,k,rr)* indicates that the *j*-th semantic item and the *k*-th semantic item should be grouped together with the argument-match-type *rr*. Note that we define four argument match types between two semantic items: *1_1*, *1_2*, *2_1* and *2_2*. Here, the argument match type *t_s* denotes that the *t*-th argument of the first semantic item corresponds to the *s*-th argument of the second semantic item[11]. The detailed illustration is shown in Table 1.

Type	Example	Question
1_1	*dbo:height 1_1 dbr:Michael_Jordan*	*How tall is Michael Jordan?*
1_2	*dbo:River 1_2 dbo:crosses*	*Which river does the Brooklyn Bridge cross?*
2_1	*dbo:creator 2_1 dbr:Walt_Disney*	*Which television shows were created by Walt Disney?*
2_2	*dbo:birthPlace 2_2 dbo:capital*	*Which actors were born in the capital of American?*

Table 1: Examples of the argument match types.

[10] This corresponds to the question "*How many people live in the capital of Australia?*"

[11] The *2*-nd argument is corresponding to the object argument of the relation, and the *1*-st argument is corresponding with the subject argument of the relation and the entity (including the class) itself.

Describing the attributes of phrases and relation between two phrases	
phraseIndex(p, i, j)	The start and end position of phrase *p* in question.
phrasePosTag(p, pt)	The POS tag of the head word in phrase *p*.
phraseDepTag(p, q, dt)	The dependency path tags between phrase *p* and *q*.
phraseDepOne (p, q)	If there is only one tag in the dependency path, the predicate is true.
hasMeanWord (p, q)	If there is any one meaning word in the dependency path of two phrases, the predicate is true.
Describing the attributes of semantic item and the mappings between phrases and semantic items	
resourceType(r, rt)	The type of semantic item r. Types of semantic items include *Entity*, *Class* and *Relation*
priorMatchScore(p, r, s)	The prior score of phrase *p* mapping to semantic item *r*.
Describing the attributes of relation between two semantic items in a knowledge base	
hasRelatedness(p, q, s)	The semantic coherence of semantic items.
isTypeCompatible(p, q, rr)	If the semantic items *p* are type-compatible with the semantic items *q*, the predicate is true.
hasQueryResult(s, p, o, rr1, rr2)	If the triple pattern consisting of semantic items *s*, *p*, *o* and argument-match-types *rr1* and *rr2* have query results, the predicate is true.

Table 2: Descriptions of observed predicates.

Moreover, we define a set of observed predicates to describe the properties of phrases, semantic items, relations between phrases and relations between semantic items. The observed predicates and descriptions are shown in Table 2.

Previous methods usually designed some heuristic patterns to group semantic items, which usually employed a human-designed syntactic path between two phrases to determine their relations. In contrast, we collect all the tokens in the dependency path between two phrases as possible patterns. The predicates *phraseDepTag* and *hasMeanWord* are designed to indicate the possible patterns. Note that if these tokens only contain POS tags *dt*|*in*|*wdt*|*to*|*cc*|*ex*|*pos*|*wp* or *stop words*, the value of the predicate *hasMeanWord* is false; otherwise, it is true. In this way, our system is expected to cover more question expressions. Moreover, the SPARQL endpoint is used to verify the type compatibility of two semantic items and if one triple pattern can obtain query results.

The predicate *hasRelatedness* needs to compute the coherence score between two semantic items. Following (Yahya et al., 2012), we use the Jaccard coefficient (Jaccard, 1908) based on the inlinks between two semantic items.

The predicate *priorMatchScore* assigns a prior score when mapping a phrase to a semantic item. We use different methods to compute this score according to different semantic item types. For entities, we use a normalized score based on the frequencies of a phrase referring to an entity. For classes and relations, we use different methods. We first define the following three similarity metrics: a) s_1: The Levenshtein distance score (Navarro, 2001) between the labels of the semantic item and the phrase; b) s_2: The word embedding (Mikolov et al., 2010) score, which measures the similarity between two phrases and is the maximum cosine value of the words' word embeddings between two phrases; and c) s_3: the instance overlap score, which is computed using the Jaccard coefficient of the instance overlap. All scores are normalized to produce a comparable scores in the interval of (0, 1). The final prior scores for mapping phrases to classes and relations are $\gamma s_1 + (1-\gamma)s_2$ and $\alpha s_1 + \beta s_2 + (1-\alpha-\beta)s_3$, respectively. The parameters are set to empirical values[12].

4.3 Formulas

According to these predicates, we design several first-order logic formulas for joint disambiguation. As mentioned in the first section, these formulas represent the meta patterns. The concrete patterns can be generated through these meta patterns with training data. Specifically, we use two types of formulas for the joint decisions: *Boolean* and *Weighted* formulas. *Boolean* formulas are hard constraints, which must be satisfied by all of the ground atoms in the final inference results. *Weighted* formulas are soft constraints, which can be violated with some penalties.

4.3.1 Boolean Formulas (Hard Constraints)

Table 3 lists the Boolean formulas used in this work. The "_" notation in the formulas indicates an arbitrary constant. The "$|f|$" notation expresses the number of true grounded atoms in the formula f. These formulas express the following constraints:

hf1: If a phrase is chosen, then it must have a mapped semantic item;

hf2: If a semantic item is chosen, then its mapped phrase must be chosen;

hf3: A phrase can be mapped to at most one semantic item;

hf4: If the phrase is not chosen, then its mapped

[12] Set γ to 0.6 for Class and set α and β to 0.3 and 0.3 for Relation, respectively.

hf1	$hasPhrase(p) \Rightarrow hasResource(p, \_)$
hf2	$hasResource(p, \_) \Rightarrow hasPhrase(p)$
hf3	$\lvert hasResource(p, \_)\rvert \leq 1$
hf4	$!hasPhrase(p) \Rightarrow !hasResource(p, r)$
hf5	$hasResource(\_, r) \Rightarrow hasRelation(r, \_, \_) \vee hasRelation(\_, r, \_)$
hf6	$\lvert hasRelation(r1, r2, \_)\rvert \leq 1$
hf7	$hasRelation(r1, r2, \_) \Rightarrow hasResource(\_, r1) \wedge hasResource(\_, r2)$
hf8	$phraseIndex(p1, s1, e1) \wedge phraseIndex(p2, s2, e2) \wedge overlap(s1, e1, s2, e2) \wedge hasPhrase(p1) \Rightarrow !hasPhrase(p2)$
hf9	$resourceType(r, \text{“}Entity\text{”}) \Rightarrow !hasRelation(r, \_, \text{“2\_1”}) \wedge !hasRelation(r, \_, \text{“2\_2”})$
hf10	$resourceType(r, \text{“}Entity\text{”}) \Rightarrow !hasRelation(\_, r, \text{“2\_1”}) \wedge !hasRelation(r, \_, \text{“2\_2”})$
hf11	$resourceType(r, \text{“}Class\text{”}) \Rightarrow !hasRelation(r, \_, \text{“2\_1”}) \wedge !hasRelation(r, \_, \text{“2\_2”})$
hf12	$resourceType(r, \text{“}Class\text{”}) \Rightarrow !hasRelation(\_, r, \text{“2\_1”}) \wedge !hasRelation(r, \_, \text{“2\_2”})$
hf13	$!isTypeCompatible(r1, r2, rr) \Rightarrow !hasRelation(r1, r2, rr)$

Table 3: Descriptions of Boolean formulas.

sf1	$priorMatchScore(p, r, s) \Rightarrow hasPhrase(p)$
sf2	$priorMatchScore(p, r, s) \Rightarrow hasResource(p)$
sf3	$phrasePosTag(p, pt+) \wedge resourceType(r, rt+) \Rightarrow hasResource(p, r)$
sf4	$phraseDepTag(p1, p2, dp+) \wedge hasResource(p1, r1) \wedge hasResource(p2, r2) \Rightarrow hasRelation(r1, r2, rr+)$
sf5	$phraseDepTag(p1, p2, dp+) \wedge hasResource(p1, r1) \wedge hasResource(p2, r2) \wedge !hasMeanWord(p1, p2) \Rightarrow hasRelation(r1, r2, rr+)$
sf6	$phraseDepTag(p1, p2, dp+) \wedge hasResource(p1, r1) \wedge hasResource(p2, r2) \wedge phraseDepOne(p1, p2) \Rightarrow hasRelation(r1, r2, rr+)$
sf7	$hasRelatedness(r1, r2, s) \wedge hasResource(\_, r1) \wedge hasResource(\_, r2) \Rightarrow hasRelation(r1, r2, \_)$
sf8	$hasQueryResult(r1, r2, r3, rr1, rr2) \Rightarrow hasRelation(r1, r2, rr1) \wedge hasRelation(r2, r3, rr2)$

Table 4: Descriptions of weighted formulas.

semantic item should not be chosen;
hf5: If a semantic item is chosen, then it should have at least one argument match relation with other semantic items;
hf6: Two semantic items have at most one argument match relation;
hf7: If an argument match relation for two semantic items is chosen, then they must be chosen;
hf8: Each of two chosen phrases must not overlap;
hf9**, **hf10**, **hf11**, **hf12: The semantic item with type *Entity* and *Class* should not have a second argument that matches with others;
hf13: The chosen argument match relation for two sematic items must be type compatible.

4.3.2 Weighted Formulas (Soft Constraints)

Table 4 lists the weighted formulas used in this work. The "+" notation in the formulas indicates that each constant of the logic variable should be weighted separately. Those formulas express the following properties in joint decisions:
sf1**, **sf2: The larger the score of the phrase mapping to a semantic item, the more likely the corresponding phrase and semantic item should been chosen;
sf3: There are some associations between the POS tags of phase and the types of mapped semantic items;
sf4**, **sf5**, **sf6: There are some associations between the dependency tags in the dependency pattern path of two phases and the types of argument match relations of two mapped semantic items;
sh7: The larger the relatedness of two semantic items, the more likely they have an argument match relation;
sf8: If the triple pattern has query results, these semantic items should have corresponding argument match relations.

5 Experiments

5.1 Dataset & Evaluation Metrics

We use the following three collections of questions from the QALD[13] task for question answering over linked data: QALD-1, QALD-3 and QALD-4. The generated SPARQL queries are evaluated on Linked Data from DBpedia and YAGO using a Virtuoso engine[14]. A typical example question from the QALD benchmark is *"Which books written by Kerouac were published by Viking Press?"*. As mentioned in Section 2.2, our system is not designed to answer questions that contain numbers, date comparisons and aggregation operations such as *group by* or *order by*. Therefore, we remove these types of questions and retain 110 questions from the QALD-4 training set for generating the specific formulas and for training their weights in MLN. We test our system using 37, 75 and 26 questions from the training set of QALD-1[15], and the testing set of QALD-3 and QALD-4 respectively. We use #T, #Q and #A to indicate the total

[13] www.sc.cit-ec.uni-bielefeld.de/qald/

[14] https://github.com/openlink/virtuoso-opensource

[15] We use the training set because we try to make a fair comparison with (Yahya et al., 2012).

number of questions in the testing set, the number of questions we could address and the number of questions answered correct, respectively. We select Precision ($P = \frac{\#A}{\#Q}$), Recall ($R = \frac{\#A}{\#T}$), and F1-score ($F1 = \frac{2 \cdot P \cdot R}{P+R}$) as the evaluation metrics. To assess the effectiveness of the disambiguation process in the MLN, we computed the overall quality measures by precision and recall with the manually obtained results.

5.2 Experimental Configurations

The Stanford dependency parser (De Marneffe et al., 2006) is used for extracting features from the dependency parse trees. We use the toolkit *thebeast*[16] to learn the weights of the formulas and to perform the MAP inference. The inference algorithm uses a cutting plane approach. In addition, for the parameter learning, we set all initial weights to zero and use an online learning algorithm with MIRA update rules to update the weights of the formulas. The number of iterations for the training and testing are set to 10 and 200, respectively.

5.3 Results and Discussion

5.3.1 The Effect of Joint Learning

To demonstrate the advantages of our joint learning, we design a pipeline system for comparison, which independently performs phrase detection, phrase mapping, and semantic item grouping by removing the unrelated formulas in MLN. For example, the formulas[17] related to the predicates *hasResource* and *hasRelation* are removed when detecting phrases in questions.

Table 5 shows the results, where **Joint** denotes the proposed method with joint inference and **Pipeline** denotes the compared method performing each step independently. We perform a comparison with the question answering results of QALD (QA), and comparisons at each of the following steps: PD (phrase detection), PM (phrase mapping) and MG (mapped semantic items grouping). From the results, we observe that our method answers over half of the questions. Moreover, our joint model based on MLN can obtain better performance in question answering compared to the pipeline system. We also observe that Joint exhibits better performance than Pipeline in most steps, except for MG in QALD-3. We believe this is because the three tasks (phrase detection, phrase mapping, and semantic item grouping) are connected with each other. Each step can provide useful information for the other two tasks. Therefore, performing joint inference can effectively improve the performance. Finally, we observe that the former task usually produces better results than the subsequent tasks (phrase detection exhibits a better performance than phrase mapping, and phrase mapping exhibits a better performance than semantic item grouping). The main reason is that the latter subtask is more complex than the former task. The decisions of the latter subtask strongly rely on the former results even though they have interacted effects.

5.3.2 The Effect of Pattern Learning

Table 6 shows a comparison of our system with *DEANNA* (Yahya et al., 2012), which is based on a joint disambiguation model but which employs hand-written patterns in its system. Because *DEANNA* only reports its results of the QALD-1 dataset, we do not show the results for QALD-3 and QALD-4 for equity. From the results, we can see that our system solved more questions and exhibited a better performance than did *DEANNA*. One of the greatest strengths of our system is that the learning system can address more questions than hand-written pattern rules.

System	#T	#Q	#A	P	R	F1
DEANNA (Yahya et al., 2012)	50	27	13	0.48	0.26	0.33
Ours	50	37	20	0.54	0.4	0.46

Table 6: Comparisons with *DEANNA* using the QALD-1 test questions.

Compared to the ILP (Integer Linear Programming) used in (Yahya et al., 2012) for joint disambiguation, we argue that there are two major differences to our method. 1) Our method is a data-driven approach that can learn effective patterns or rules for the task. Therefore, it exhibits more robustness and adaptability for various *KBs*. 2) We design several meta rules in MLN as opposed to specific ones. The specific rules can be generated by these meta rules based on the training data. By contrast, the traditional approach using ILP needs to set specific rules in advance, which requires more intensive labor than our approach.

To further illustrate the effectiveness of our pattern-learning strategy, we show the weights of the learned patterns corresponding to formula *sf3* in the MLN, as shown in Table 7. From the table,

[16] http://code.google.com/p/thebeast

[17] including entire formulas, excluding *hf8* and *sf1*

Benchmark	PD			PM			MG			QA					
	P	R	F1	P	R	F1	P	R	F1	#T	#Q	#A	P	R	F1
QALD-1(Joint)	0.93	0.981	**0.955**	0.895	0.944	**0.919**	0.703	0.813	**0.754**	50	37	20	0.54	0.4	**0.46**
QALD-1(Pipeine)	0.921	0.972	0.946	0.868	0.917	0.892	0.585	0.859	0.696	50	34	17	0.5	0.34	0.41
QALD-3(Joint)	0.941	0.941	**0.941**	0.878	0.918	**0.898**	0.636	0.798	0.708	99	75	45	0.6	0.46	**0.52**
QALD-3(Pipeline)	0.912	0.912	0.912	0.829	0.867	0.848	0.677	0.789	**0.729**	99	75	42	0.56	0.42	0.48
QALD-4(Joint)	0.947	0.978	**0.963**	0.937	0.967	**0.952**	0.776	0.865	**0.817**	50	26	15	0.58	0.3	**0.4**
QALD-4(Pipeline)	0.937	0.967	0.952	0.905	0.935	0.920	0.683	0.827	0.748	50	24	13	0.54	0.26	0.35

Table 5: The performance of joint learning on three benchmark datasets.

we can see that *nn*[18] is more likely mapped to *Entity*[19] than to *Class* and *Relation*, and *vb* is most likely mapped to *Relation*. This proves that our model can learn effective and reasonable patterns for QALD.

POS tag of Phrase	type of mapped Item	Weight
nn	*Entity*	2.11
nn	*Class*	0.243
nn	*Relation*	0.335
vb	*Relation*	0.517
wp	*Class*	0.143
wr	*Class*	0.025

Table 7: Sample weights of formulas, corresponding with formula *sf3*.

5.3.3 Comparison to the state of the art

To illustrate the effectiveness of the proposed method, we perform comparisons to the state-of-the-art methods. Table 8 shows the results using QALD-3 and QALD-4. These systems are the participants in the QALD evaluation campaigns. From the results, we can see that our system outperforms most systems at a competitive performance. They further prove the effectiveness of the proposed method.

Test set	System	#T	#Q	#A	P	R	F1
QALD-3	CASIA (He et al., 2013)	99	52	29	0.56	0.3	0.38
	Scalewelis (Joris and Ferré, 2013)	99	70	32	0.46	0.32	0.38
	RTV (Cristina et al., 2013)	99	55	30	0.55	0.3	0.39
	Intui2 (Corina, 2013)	99	99	28	0.28	28	0.28
	SWIP (Pradel et al., 2013)	99	21	15	0.71	0.15	0.25
	Ours	99	75	45	0.6	0.46	0.52
QALD-4[20]	gAnswer	50	25	16	0.64	0.32	0.43
	Intui3	50	33	10	0.30	0.2	0.24
	ISOFT	50	50	10	0.2	0.2	0.2
	RO FII	50	50	6	0.12	0.12	0.12
	Ours	50	26	15	0.58	0.3	0.4

Table 8: Comparisons with state-of-the-art systems using the QALD benchmark.

5.3.4 The Effect of Different Formulas

To determine which formulas are more useful for QALD, we evaluate the performance of the proposed method with different predicate sets. We subtract one weighted formula from the original sets at a time, except retaining the first two formulas *sf1* and *sf2* for basic inference. Because of space limitations, only the results using QALD-3 testing set are shown in Table 9.

From the results, we can observe that removing some formulas can boost the performance on some single tasks, but employing all formulas can produce the best performance. This illustrates that solely resolving the steps in QALD (phrase detection, phrase mapping, semantic items grouping) can obtain local results, and that making joint inference is necessary and useful.

6 Related Work

Our proposed method is related to two lines of work: *Question Answering over Knowledge bases* and *Markov Logic Networks*.

Question answering over knowledge bases has attracted a substantial amount of interest over a long period of time. The initial attempts included BaseBall (Green Jr et al., 1961) and Lunar (Woods, 1977). However, these systems were mostly limited to closed domains due to a lack of knowledge resources. With the rapid development of structured data, such as DBpedia, Freebase and Yago, the need for providing user-friendly interface to these data has become increasingly urgent. Keyword (Elbassuoni and Blanco, 2011) and semantic (Pound et al., 2010) searches are limited to their ability to specify the relations among the different keywords.

The open topic progress has also been pushed by the QALD evaluation campaigns (Walter et al., 2012). Lopez et al. (2011) gave a comprehensive survey in this research area. The authors developed the *PowerAqua* system (Lopez et al., 2006) to

[18] The POS tag of the head word in the phrase

[19] The type of semantic item

[20] Because the QALD-4 conference does not start until after submission, we have no citation for the state-of-the-art systems in QALD-4. The results can be found at http://greententacle.techfak.uni-bielefeld.de/ cunger/qald.

Formulas	PD			PM			MG			Avg		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
All Formulas	**0.941**	**0.941**	**0.941**	0.878	**0.918**	**0.898**	0.636	0.798	0.708	0.839	**0.901**	**0.869**
-sf3	0.931	0.927	0.929	0.877	0.913	0.895	0.637	**0.816**	0.715	0.834	0.897	0.864
-sf4	0.926	0.917	0.922	0.852	0.883	0.867	0.63	0.763	0.69	0.824	0.87	0.846
-sf5	0.931	0.927	0.929	0.873	0.908	0.89	0.633	**0.816**	0.713	0.831	0.895	0.862
-sf6	0.922	0.922	0.922	0.844	0.883	0.863	**0.702**	0.746	**0.723**	**0.842**	0.868	0.855
-sf7	0.931	0.917	0.924	**0.881**	0.908	0.894	0.621	0.763	0.685	0.833	0.88	0.856
-sf8	0.927	0.927	0.927	0.868	0.908	0.888	0.639	0.807	0.713	0.83	0.893	0.861

Table 9: Performance comparisons of different weighted formulas evaluated using the QALD-3 question set.

answer questions on large, heterogeneous datasets. For questions containing quantifiers, comparatives or superlatives, Unger et al. (2012) translated *NL* to *FL* using several SPARQL templates and using a set of heuristic rules mapping phrases to semantic items. The system most similar to ours is *DEANNA* (Yahya et al., 2012). However, *DEANNA* extracts predicate-argument structures from the questions using three hand-written patterns. Our system jointly learns these mappings and extractions completely from scratch.

Recently, the Semantic Parsing (SP) community targeted this problem from limited domains (Tang and Mooney, 2001; Liang et al., 2013) to open domains (Cai and Yates, 2013; Berant et al., 2013). The methods in semantic parsing answer questions by first converting natural language utterances into meaningful representations (e.g., the lambda calculus) and subsequently executing the formal logical forms over *KBs*. Compared to deriving the complete logical representation, our method aims to parse a question into a limited logic form with the semantic item query, which we believe is more appropriate for answering factoid questions.

Markov Logic Networks have been widely used in NLP tasks. Huang (2012) applied MLN to compress sentences by formulating the task as a word/phrase deletion problem. Fahrni and Strube (2012) jointly disambiguated and clustered concepts using MLN. MLN has also been used in coreference resolution (Song et al., 2012). For the task of identifying subjective text segments and of extracting their corresponding explanations from product reviews, Zhang et al. (2013) modeled these segments with MLN. To discover logical knowledge for deep question answering, Liu (2012) used MLN to resolve the inconsistencies of multiple knowledge bases.

Meza-Ruiz and Riedel (2009) employed MLN for Semantic Role Labeling (SRL). They jointly performed the following tasks for a sentence: predicate identification, frame disambiguation, argument identification and argument classification. The semantic analysis of SRL solely rested on the lexical level, but our analysis focuses on the knowledge-base level and aims to obtain an executable query and to support natural language inference.

7 Conclusions and Future Work

For the task of QALD, we present a joint learning framework for phrase detection, phrase mapping and semantic item grouping. The novelty of our method lies in the fact that we perform joint inference and pattern learning for all subtasks in QALD using first-order logic. Our experimental results demonstrate the effectiveness of the proposed method.

In the future, we plan to address the following limitations that still exist in the current system: a) numerous hand-labeled data are required for training the MLN, and we could use a latent form of semantic item query graphs (Liang et al., 2013); b) more robust solutions can be developed to find the implicit relations in questions; c) our system can be scaled up to large-scale open-domain knowledge bases (Fader et al., 2013; Yao and Van Durme, 2014); and d) the learning system has the advantage of being easily adapted to new settings, and we plan to extend it to other domains and languages (Liang and Potts, 2014).

Acknowledgments

The authors are grateful to the anonymous reviewers for their constructive comments. This work was sponsored by the National Basic Research Program of China (No. 2014CB340503) and the National Natural Science Foundation of China (No. 61202329, 61272332), CCF-Tencent Open Fund. This work was also supported in part by Noahs Ark Lab of Huawei Tech. Ltm.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *ACL*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

Christian Bizer, Tom Heath, and Tim Berners-Lee. 2009. Linked data-the story so far. *International journal on semantic web and information systems*, 5(3):1–22.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *ACL*.

Unger Christina and Andr Freitas. 2014. Question answering over linked data: Challenges, approaches, trends. In *ESWC*.

Dima Corina. 2013. Intui2: A prototype system for question answering over linked data. In *Work. Multilingual Question Answering over Linked Data (QALD-3)*.

Giannone Cristina, Bellomaria Valentina, and Basili Roberto. 2013. A hmm-based approach to question answering against linked data. In *Work. Multilingual Question Answering over Linked Data (QALD-3)*.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *LREC*.

Shady Elbassuoni and Roi Blanco. 2011. Keyword search over rdf graphs. In *CIKM*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *EMNLP*.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *ACL*.

Angela Fahrni and Michael Strube. 2012. Jointly disambiguating and clustering concepts and entities with markov logic. In *COLING*.

Andre Freitas and Edward Curry. 2014. Natural language queries over heterogeneous linked data graphs: A distributional-compositional semantics approach. In *IUI*.

Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. 1961. Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pages 219–224. ACM.

Shizhu He, Shulin Liu, Yubo Chen, Guangyou Zhou, Kang Liu, and Jun Zhao. 2013. Casia@qald-3: A question answering system over linked data. In *Work. Multilingual Question Answering over Linked Data (QALD-3)*.

Minlie Huang, Xing Shi, Feng Jin, and Xiaoyan Zhu. 2012. Using first-order logic to compress sentences. In *AAAI*.

Paul. Jaccard. 1908. Nouvelles recherches sur la distribution florale. *Bulletin de la Sociète Vaudense des Sciences Naturelles*, 44:223–270.

Guyonvarc'H Joris and Sébastien Ferré. 2013. Scalewelis: a scalable query-based faceted search system on top of sparql endpoints. In *Work. Multilingual Question Answering over Linked Data (QALD-3)*.

Jens Lehmann, Tim Furche, Giovanni Grasso, Axel-Cyrille Ngonga Ngomo, Christian Schallhart, Andrew Sellers, Christina Unger, Lorenz Bühmann, Daniel Gerber, Konrad Höffner, et al. 2012. Deqa: deep web extraction for question answering. In *ISWC*.

Percy Liang and Christopher Potts. 2014. Bringing machine learning and compositional semantics together. *Annual Reviews of Linguistics (to appear)*.

Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Zhao Liu, Xipeng Qiu, Ling Cao, and Xuanjing Huang. 2012. Discovering logical knowledge for deep question answering. In *CIKM*.

Vanessa Lopez, Enrico Motta, and Victoria Uren. 2006. Poweraqua: Fishing the semantic web. In *The Semantic Web: research and applications*, pages 393–410. Springer.

Vanessa Lopez, Victoria Uren, Marta Sabou, and Enrico Motta. 2011. Is question answering fit for the semantic web?: a survey. *Semantic Web*, 2(2):125–155.

Ivan Meza-Ruiz and Sebastian Riedel. 2009. Jointly identifying predicates, arguments and senses using markov logic. In *NAACL*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. Patty: a taxonomy of relational patterns with semantic types. In *EMNLP*.

Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88.

Jeffrey Pound, Ihab F Ilyas, and Grant Weddell. 2010. Expressive and flexible access to web-extracted data: a keyword-based structured query language. In *SIGMOD*.

C Pradel, G Peyet, O Haemmerlé, and N Hernandez. 2013. Swip at qald-3: results, criticisms and lesson learned (working notes). In *Work. Multilingual Question Answering over Linked Data (QALD-3)*.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning*, 62(1-2):107–136.

Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer. 2013. Question answering on interlinked data. In *WWW*.

Yang Song, Jing Jiang, Wayne Xin Zhao, Sujian Li, and Houfeng Wang. 2012. Joint learning for coreference resolution with markov logic. In *EMNLP*.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW*.

Lappoon R. Tang and Raymond J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th European Conference on Machine Learning*, pages 466–477.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over rdf data. In *WWW*.

Sebastian Walter, Christina Unger, Philipp Cimiano, and Daniel Bär. 2012. Evaluation of a layered approach to question answering over linked data. In *The Semantic Web–ISWC 2012*, pages 362–374. Springer.

William A Woods. 1977. Lunar rocks in natural english: Explorations in natural language question answering. In *Linguistic structures processing*, pages 521–569.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. 2012. Natural language questions for the web of data. In *EMNLP*.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. 2013. Robust question answering over the web of linked data. In *CIKM*.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *ACL*.

Qi Zhang, Jin Qian, Huan Chen, Jihua Kang, and Xuanjing Huang. 2013. Discourse level explanatory relation extraction from product reviews using first-order logic. In *ACL*.

Lei Zou, Ruizhe Huang, Haixun WangZou, Jeffrey Xu Yu, Wenqiang He, and Dongyan Zhao. 2014. Natural language question answering over rdf — a graph data driven approach. In *SIGMOD*.