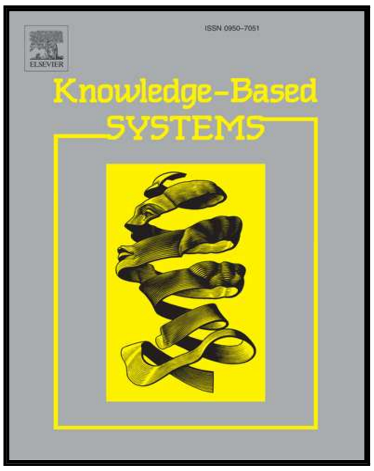

Grounding Proposition Stores for Question Answering over Linked Data

Bernardo Cabaleiro, Anselmo Peñas, Suresh Manandhar

PII: S0950-7051(17)30192-2
DOI: 10.1016/j.knosys.2017.04.016
Reference: KNOSYS 3896

To appear in: *Knowledge-Based Systems*

Received date: 17 June 2016
Revised date: 25 April 2017
Accepted date: 25 April 2017

Highlights

- Grounding natural language is required for question answering but the contribution remains unmeasured.
- We provide a standalone evaluation and propose a method to ground propositions into a knowledge base.
- Results show how grounding accounts for 78.6
- Simple lexical expansion can improve the results from 0.8

Grounding Proposition Stores for Question Answering over Linked Data

Bernardo Cabaleiro[a], Anselmo Peñas[a], Suresh Manandhar[b]

[a]*NLP & IR Group at UNED*
bcabaleiro,anselmo@lsi.uned.es
C/ Juan del Rosal, 16 28040 Madrid Spain
+34 91 398 8407 - +34 91 398 7750
[b]*Department of Computer Science at University of York*
suresh.manandhar@york.ac.uk
Heslington, York, YO10 5GH, United Kingdom
+44 (0)1904 325677

Abstract

Grounding natural language utterances into semantic representations is crucial for tasks such as question answering and knowledge base population. However, the importance of the lexicons that are central to this mapping remains unmeasured because question answering systems are evaluated as end-to-end systems.

This article proposes a methodology to enable a standalone evaluation of grounding natural language propositions into semantic relations by fixing all the components of a question answering system other than the lexicon itself. Thus, we can explore different configurations trying to conclude which are the ones that contribute better to improve overall system performance.

Our experiments show that grounding accounts with close to 80% of the system performance without training, whereas training supposes a relative improvement of 7.6%. Finally we show how lexical expansion using external linguistic resources can consistently improve the results from 0.8% up to 2.5%.

Keywords: Question Answering, Semantic Parsing, Linked Data, Grounding

1. Introduction

Linked Data refers to *a set of best practices for publishing and connecting structured data on the Web* [1]. It establishes the bases for the Web of Data, an effort from the community of web users to create large amounts structured, machine-friendly knowledge, preserving the structure and semantics of the relations between elements. Although there are plenty of linked data databases (e.g. Freebase [2], DBPedia [3] or Yago2 [4]), common web users lack of the necessary know-how to use them.

Question Answering (QA) can be viewed as one human friendly method for accessing linked data since it alleviates the need to learn query languages such as SPARQL. QA systems typically employ semantic parsing to map natural language into a predicate-argument meaning representation. The map can easily be translated into knowledge base query languages.

We define grounding as the procedure for expressing natural language in terms of the target knowledge base language. More specifically, the task is to map an unbounded number of expressions (natural language) into a small set of entities and properties (linked data). For example the constructions *What does John do for a living?*, *What is John's profession?*, and *Who is John?* are be mapped to the same property {`John - Profession - X`}.

Grounding provides two key benefits. On the one hand, it alleviates the problem of logic form annotation by providing data for indirect supervision [5]. Secondly, if the logic forms share the same vocabulary with the target knowledge base the querying step becomes trivial.

Semantic Parsing methods require a lexicon to enable the mapping between text and the labels of the knowledge base. A *lexicon* captures and ranks the candidate mappings between predicates in natural language and properties in the linked data database. For instance, solving the previous example would require an entry *living* $\rightarrow$ `profession`. However building these lexicons is not trivial and the contribution to the full system remains unmeasured because the final score is given by the complete system and involves other processes, e.g. choosing the appropriate entry of the lexicon.

Recent work proposes a method to build a lexicon by acquiring knowledge from large text corpora [6]. This process relies on distant supervision to build a lexicon that then is used to fed a semantic parser. Our goal is to study the contribution of this process of knowledge acquisition on closing the gap between natural language and linked data properties. Specifically, it is unclear which syntactic structures should be aligned and what is the impact of each one.

We use our methods of representation and acquisition to transform natural language utterances into logic forms composed by a set of propositions, which are triples with the form `<argument 1 - predicate - argument 2>`. A propositions is mapped into a linked data triple {`argument 1 - property - argument 2`} to build a grounded proposition, which is a proposition expressed with the linked data vocabulary. We build a lexicon that we denote Grounded Proposition Store (henceforth, GPS) by grounding a large number of propositions automatically extracted from text. Finally, we combine the GPS with the method proposed in [6] to create a scenario where grounding can be evaluated in isolation to study how different grounding configurations affect semantic parsing. Figure 1 shows how the utterance *"Carrie Fisher is the actress who played Princess Leia"* is transformed to a logic form composed by two propositions and then grounded into linked data properties. We explain this process and some related concepts in Section 4.

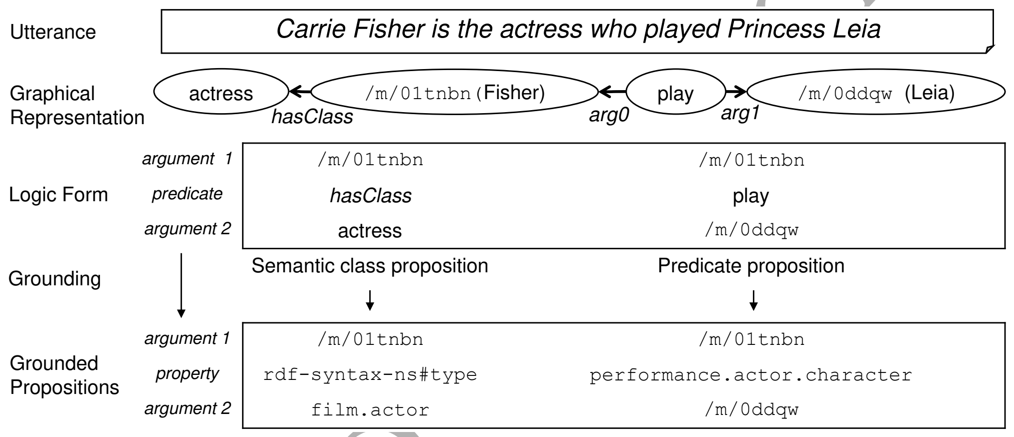

Figure 1: Example of acquisition of a grounded proposition. For simplicity, we represent the property `performance.actor.character` as a single triplet. In Freebase, this property is expressed with two triplets related by an intermediate entity.

We structure our research around the following research questions. In the context of a Semantic Parser trained using raw text for distant supervision:

- *What are the methodological steps to build a GPS?*
- *What is the impact of the GPS when used to fed a semantic parser for question answering?*
- *What linguistic phenomena (syntactic-semantic relations) should be considered in the knowledge acquisition step?*
- *Are external linguistic resources useful for enriching the GPS?*

This article is structured as follows: In Section 2 we motivate the choice of distant supervision using raw text for QA over LD. Section 3 details the architecture of the semantic parser, Section 4 studies the grounding step and presents our approach to build a GPS. In Section 5 we evaluate the effect of GPS in QA over LD and we present the results in Section 6. We finish with some conclusions in Section 7 and propose some future work in Section 8.

2. Semantic Parsing over Linked Data

Early works on semantic parsing for question answering were done on domains with controlled language and small predefined domains such as baseball [7] and geography [8]. However, these approaches cannot be scaled to general-domain knowledge bases.

As semantic parsers scaled to answer a wider range of queries, several problems arise. Firstly, systems have to deal with the lexical variability of the utterances, a problem that grows as domains become less restricted. Secondly, knowledge bases become bigger and richer, so the potential to give wrong answers increases.

Finally, dealing with the variability of knowledge bases also introduces additional challenges since semantic parsers have to adapt to different structures and vocabularies. Currently, many efforts point to linked data databases like DB-Pedia or Freebase as a source of general domain knowledge. The main reason is that they are a compromise solution between the high quality data that provide the hand-labelled databases and the extension of the automatically generated databases. Linked data databases are often structured in triples that denote relations between two entities, which are named properties. Properties are labelled with a name close to natural language. For example, an instance of the database may be {`John - profession - teacher`}, although these labels are arbitrary and, in fact, properties are defined extensively by their members.

Early approaches were too dependent on hand-labelled logic forms [8, 9, 10], and hence were unable to scale up. More recent work aims to alleviate the supervision problem by using forms of distant supervision, i.e. observation of system behaviour [11], conversations from dialog systems [12], schema matching [13], questions [5] and question-answer pairs [14, 15, 16, 17, 18].

GraphParser [6] is a method for distant supervision that hypothesizes that a natural language predicate found in a text expresses a Freebase property. The idea is to identify pairs of entities connected through a predicate in a large document collection and look for the Freebase properties that connect both entities.

For example, given the sentence $s =$ *Cameron is the director of Titanic* one of the properties in Freebase between $e_1 =$ `Cameron` and $e_2 =$ `Titanic` is $r =$ `film.directed_by`. Thus, we assume that $\{\texttt{e}_1\ \texttt{-}\ \texttt{r}\ \texttt{-}\ \texttt{e}_2\}$ corresponds to the natural language expression s.

Distant supervision provides a noisy method to learn weights for each predicate-property pairs. For this purpose, the starting point is to take as prior the frequencies observed in a large text collection to build a lexicon and use it to feed the learning process.

GraphParser tackles this task by pairing reified logic forms derived from a Combinatory Categorial Grammar (CCG) parser [19] with Freebase properties. Each logic form corresponds in turn to a predicate-argument relation. Instead, we show how to obtain similar logic forms from a standard dependency parser. Dependency trees are transformed into graphs, which are then used to extract propositions. Then, propositions are aligned with Freebase to produce a new lexicon.

This setting allows us to measure the effect that different configurations of our Propositions Stores produce on semantic parsing when they are grounded to build the lexicon the system requires.

3. System Architecture

In this section we revise the architecture of the Question Answering system. The system is divided in three main layers: Text Processing, Learning and Inference. Figure 2 illustrates the architecture diagram.

3.1. Text Processing

The purpose of text processing is to structure each document into a machine-friendly representation. This includes both sentences from the document collection and questions for the test. Questions are further analysed to extract the focus. The main tasks are the following:

1. Entity Linking: The entity linking component maps natural language entities to their canonical form in the linked data database, which is often done with a software tuned for the target database.
   For our experimentation we use the ClueWeb09 Corpus [20], which is already automatically tagged with Freebase entities [21]. This is an automatic process whose authors estimate that has between 80-85% precision and 70-85% recall.
   Performance on this step has a double impact: On the one hand, unlinked entities are lost for the grounding step. This is a small problem, because the

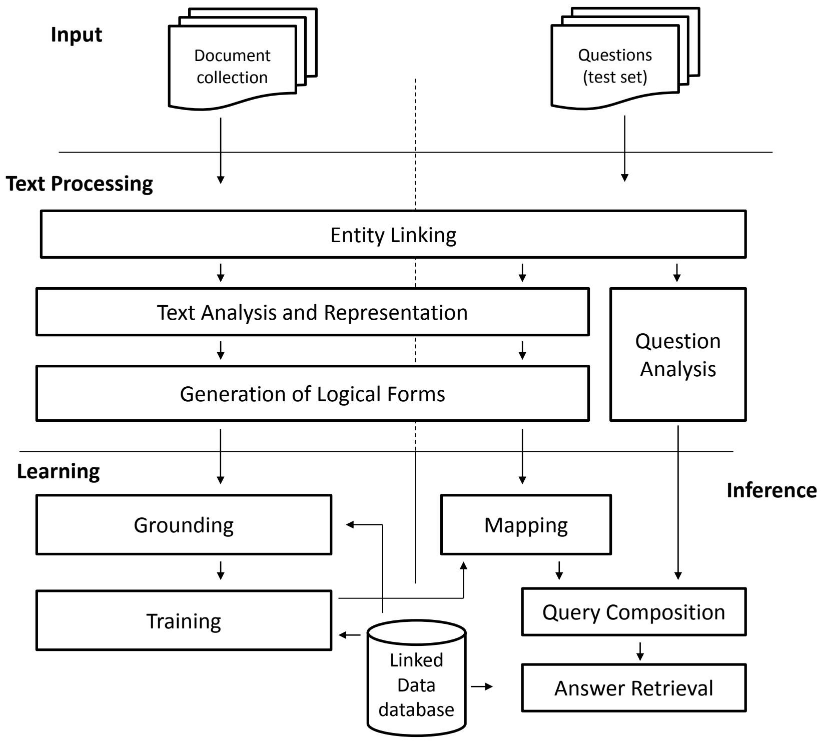

Figure 2: Architecture of the Question Answering system.

goal of the grounding step is to collect a wide sample of linguistic phenomena, not to ground every entity. On the other hand, mistakes on this phase introduce noise on the system, which we cannot detect on posterior steps.

2. Text Analysis and Representation: In this step, the system takes a sentence annotated with entities and produces a structured representation. It is expected that the structured representation is closer to the meaning of the sentence, and therefore mapping it to a property should be easier. GraphParser uses in this step a CCG parser. In our case, we rely on dependency parsing to create a graph representation.
3. Generation of Logic Forms: This step is devoted to flatten the graphical representation. In GraphParser, this step is equivalent to generate a set of predicates denoted as ungrounded graphs. In our approach we obtain neo-

davidsonian reified logic forms from dependency trees, and from them we select predicate structures in the form of propositions. Section 4 gives more details about this processing.

4. Question Analysis: In question analysis, the main goal is to find the question focus, which is the part of the question that, if replaced by the answer, makes the question a single statement. For example, in the question *Who is the director of Titanic?*, the focus is given by *Who*, as it can be replaced by *David Cameron* to produce the affirmative statement *David Cameron is the director of Titanic*.
   Moreover, special operators like `count` and `argmax` are created by searching for special keywords like *How many* and *most* respectively.

3.2. Learning

The learning layer creates a model that evaluates a proposition to produce the most probable grounded proposition.

1. Grounding: In Grounding we take the pre-processed documents in order to build the GPS. In GraphParser, this step corresponds to the building of the alignment lexicon. We explain this step in detail in Section 4.
2. Training: We use the Maximum Weighted Graph (MWG) of GraphParser [6] to replace each new proposition with the highest weighted grounded proposition given by the GPS.
   The default configuration of GraphParser for training is far more complex. It uses a Structured Perceptron that employs several kinds of features from the alignment between logic forms and properties. Features weights are tuned using denotation as a form of distant supervision. In GraphParser, training pushes the results from 36.5% to 39.3%.

3.3. Inference

The inference layer applies the model generated in the learning layer to the questions of the test set in order to generate an answer. Our configuration follows [6]. This process is subdivided in three components: Mapping, Query Composition and Answer Retrieval.

1. Mapping: Mapping is devoted to adapt the vocabulary and structure of the logic forms of the text processing step into the logic forms grounded on Freebase. It uses the model created in the learning step to decide which the most promising grounded proposition is.

2. Query Composition: The query composition step combines the information given by question analysis with the logic forms in order to create an executable SPARQL query. As grounded propositions are expressed with Freebase vocabulary, the composition is straightforward. Besides, it adds extra metadata about prefixes, domains and optional language filters.
3. Answer Retrieval: The last step is to retrieve the answer given the SPARQL query. We use Virtuoso[1] as an open source, free-available server to allocate the database and enable querying.

4. Grounding

Despite the need of a full QA system, our goal is to get insights on the effects of the grounding step in overall system performance. The challenge is to build a map between natural language utterances and the properties in a linked data database and measure the effect on the QA task.

In our case, we turn natural language utterances into propositions in the acquisition step and then we build a map from propositions to properties, which is the Grounded Proposition Store. The generation of GPSs is divided into three steps: Proposition Store Building, Proposition Store Grounding and Lexical Expansion.

4.1. Proposition Store Building

Sentences from the ClueWeb09 Corpus are processed with Stanford CoreNLP [22] to obtain dependency trees which are also annotated with part-of-speech and coreferences [23]. We collapse multi-words nodes such as named entities into single nodes. Coreferences are also used to replace pronouns with the correspondent named entity. We depart from the Stanford syntactic dependencies [24], and then we perform a naive semantic role labelling to normalize subjects, direct objects, indirect objects, copulatives, genitives and class-instance relations with a new set of semantic dependencies (See Table 1). This is an automatic process that relies on a predefined set of patterns. As a matter of example, we show some patterns in Table 2. We aim for two advantages: First it allows us to define a simple set of patterns to extract propositions, and second, this normalization reduces the sparsity of the extraction.

Then, we extract a set of propositions applying patterns based on semantic dependencies. A proposition is composed by a predicate with two arguments,

[1]`http://virtuoso.openlinksw.com/`

Role	Semantic dependency
Subject	subject
Direct object	dobject
Indirect object	iobject
Copulative	is
Genitive	has
Semantic class	hasClass
Prepositions	prep

Table 1: Semantic dependencies introduced by the semantic role labelling.

Input	Output
$ne(N_1), N_1 \longrightarrow nn \longrightarrow N_2$	$N_1 \longrightarrow hasClass \longrightarrow N_2$
$ne(N_2), N_1 \longrightarrow poss \longrightarrow N_2$	$N_2 \longrightarrow has \longrightarrow N_1$
$V \longrightarrow subj \longrightarrow N$	$V \longrightarrow subject \longrightarrow N$
$V \longrightarrow agent \longrightarrow N$	$V \longrightarrow subject \longrightarrow N$
$N_1 \longrightarrow nsubj \longrightarrow N_2$	$N_1 \longrightarrow is \longrightarrow N_2$

Table 2: Examples of patterns for naive semantic role labelling. N represent nouns and V are verbs. $ne(N)$ means that the noun N is a named entity.

and is denoted as `<arg1 - predicate - arg2>`. We distinguish two kinds of propositions: Semantic class propositions, where the predicate denotes a type relationship (See Table 3), and predicate propositions, where the predicate denotes any other relationship (See Table 4).

Pattern	Example	Proposition
NhasClassN	*Beatty scored a double-win by casting Madonna as chanteuse Breathless Mahoney.*	`<Madonna - hasClass - chanteuse>`
NisN	*War of the Worlds is a movie with Tom Cruise.*	`<War of the Worlds - is - movie>`

Table 3: Syntactic patterns used to extract semantic classes from the graphical representation.

4.2. Proposition Store Grounding

We ground propositions in the following manner:

Pattern	Example	Proposition
NhasClassN+NPN	*Robby Benson, a surprising choice for The Beast, is excellent.*	`<Robby Benson - choice:for - The Beast>`
NisN+NPN	*War of the Worlds is a movie with Tom Cruise.*	`<War of the Worlds - movie:with - Tom Cruise>`
NhasN	*Likewise, Jackman's Drover is a surprising bore.*	`<Jackman - has - Drover>`
NhasN+NPN	*Main Hoon Na is a Bollywood's film of 2004 starring Shahrukh and Sushmita Sen.*	`<Main Hoon - film:of - 2004>`
NPN	*Frank Welker was the voice of Megatron.*	`<voice - voice:of - Megatron>`
NNV	*Nichols and Koenig played Uhura and Chekhov, respectively.*	`<Nichols - play - Koening>`
NVN	*Four Rooms was released by Miramax in December, 1995.*	`<Miramax - release - Four Rooms>`
NVPN	*Don Knotts won five Emmys as Barney Fife.*	`<Don Knotts - win - Barney Five>`
VNN	*The Incredible Hulk also starring Liv Tyler, Tim Roth and William Hurt.*	`<Liv Tyler - star - Tim Roth>`
VNPN	*Tarantino is scheduled to begin shooting Death Proof in Austin in August.*	`<Death Proof - shoot:in - Austin>`
VPNPN	*Under Berg, Hancock was filmed in Los Angeles.*	`<Los Angeles - film:in:under - Berg>`

Table 4: Syntactic patterns used to extract propositions from the graphical representation. NVNPN is equivalent to Subject - Verb - Direct Object -Indirect Object

1. Select sentences with two or more entities present in Freebase.
2. Extract the set of propositions from the sentence that involve the entities present in Freebase.
3. Retrieve all possible types and properties from Freebase that link the entities found.
4. Pair propositions and retrieved properties to build the mapping lexicon. Semantic classes are mapped to types, and predicate propositions are mapped to properties.
5. Compute the join probability of each pairing between a predicate r with a property p as a way to rank the most probable properties for a given predicate. The joint probability is calculated as:

$$p(r,p) = \sum_{(arg1,arg2)\in ARG} p(p \mid arg1, arg2) \cdot p(arg1, arg2 \mid r) \cdot p(r) \quad (1)$$

where $p(p \mid arg1, arg2)$ is estimated as $\frac{1}{|P|_{arg1,arg2}}$ being $|P|_{arg1,arg2}$ the number of distinct properties retrieved for a pair of arguments $arg1$ and $arg2$ that belong to the set of all arguments ARG if p is retrieved, and 0 otherwise, $p(arg1, arg2 \mid r)$ is estimated as $\frac{\#r(arg1,arg2)}{\#r}$ where $\#r(arg1, arg2)$ corresponds to the number of times where a proposition `<arg1 - r - arg2>` is derived from the corpus and $\#r$ is the number of times that the predicate r is derived from the corpus, and $p(r)$ is estimated as $\frac{\#r}{|R|}$ where $|R|$ is the total number of propositions.

The result is a GPS with a total of 3,416 semantic classes aligned to an average of 40.52 types, plus 10,799 predicates aligned to an average of 3.82 properties. Tables 5 and 6 show some of the most frequent pairs extracted for semantic classes and predicates respectively.

Semantic class	Type	$p(r,p)$
	`organization.organization_founder`	5.96E-4
president	`business.board_member`	4.23E-4
	`people.person`	3.43E-4
	`people.person`	6.63E-4
son	`people.deceased_person`	2.84E-4
	`people.family_member`	5.20E-5
	`organization.organization_founder`	1.68E-3
founder	`business.board_member`	1.54E-4
	`people.person`	1.37E-4

Table 5: Highest probability pairings between semantic classes and types in Freebase.

4.3. Lexical Expansion

Alignment through examples is very sensitive to lexical variability. For instance, we may not find the verb film in a proposition like `<Cameron - film - Titanic>`. However, if instead we have found the verb *direct* like in `<Cameron - direct - Titanic>` we could easily expand our lexicon by replacing the

Predicate	Property Argument 1	Property Argument 2	$p(r,p)$
	`person.place_of_birth.1`	`person.place_of_birth.2`	0.014
bear	`place_lived.person`	`place_lived.location`	0.008
	`person.nationality.1`	`person.nationality.2`	0.006
	`person.nationality.1`	`person.nationality.2`	0.005
has	`employment_tenure.person`	`employment_tenure.company`	0.004
	`organization.geographic_scope.1`	`organization.geographic_scope.2`	0.001
	`deceased_person.place_of_death.1`	`deceased_person.place_of_death.2`	0.007
die:in	`deceased_person.date_of_death.1`	`deceased_person.date_of_death.2`	0.001
	`person.date_of_birth.1`	`person.date_of_birth.2`	0.001

Table 6: Higher probability pairings between predicates and properties. We removed the domains of the properties for readability.

verb *direct* with the related verb *film*. We explore the use of synonyms in WordNet [25], a hand-made lexical database. Lexical expansion through external resources has proven to be useful in Semantic Parsing both with WordNet [26] and other sources [14].

The lexical expansion process takes a predicate paired with the properties and obtains every synonym given by WordNet, including the original word. Then, the final weight of each predicate is divided among the number of synonyms retrieved. We compute the joint probability of each pairing between a synonym s with a property p, that is:

$$p'(s,p) = \sum_r p(s,r,p) = \sum_r p(s \mid r,p) \cdot p(r,p) \tag{2}$$

where $p(s \mid r,p)$ is estimated as $\frac{1}{|S|_r}$ being $|S|_r$ the number of synonyms retrieved for the predicate r and $p(r,p)$ is calculated as in Equation 1.

The resulting expanded GPS extends the predicates up to 72,130, with an average of 5.42 properties paired.

4.4. Working example

We illustrate the full process through a working example. Consider the sentence *Spurlock is the creator of the film Supersize Me*, extracted from the ClueWeb09 Corpus. Our method is decomposed in the following steps:

1. Proposition Store Building
   (a) Select a sentence s: The working example is selected because it contains two Freebase entities, *Spurlock* and *Supersize Me*. Entities are

annotated with their Freebase id, which is `m.035sc2` and `m.022prxf` respectively.

(b) Transform the sentence into a graph g_s: Figure 3 shows the resulting graph.

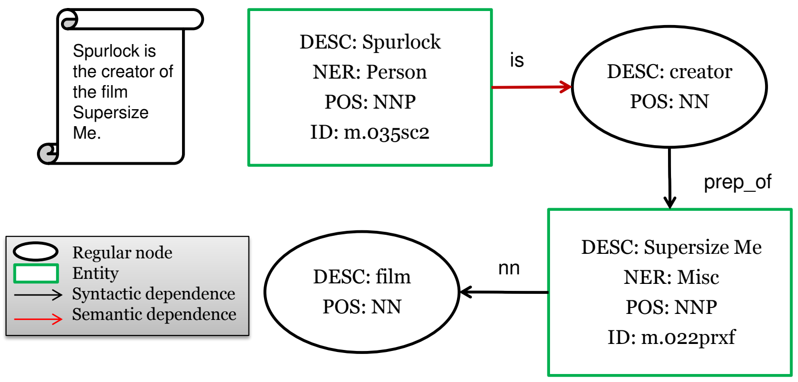

Figure 3: Graph extracted from the sentence *Spurlock is the creator of the film Supersize Me.*

(c) Extract propositions `<arg1-r-arg2>`: We flatten the graph representation applying syntactic patterns to extract propositions, composed by a predicate r and a pair of arguments $arg1, arg2$. Table 7 shows the patterns found in the working example and the resulting propositions.

Pattern	Proposition
NPN	`<creator - of - m.022prxf>`
NisN	`<m.035sc2 - is -creator>`
NisN+NPN	`<m.035sc2 - creator:of - m.022prxf>`

Table 7: Propositions extracted from the sentence *Spurlock is the creator of the film Supersize Me*. Note that entities are replaced with their Freebase id.

2. Grounding Proposition Stores

(a) Ground propositions: A grounded proposition $\{$`arg1 - p - arg2`$\}$ is built by replacing the original predicate r of a proposition with a property p. We search for the properties $P_{arg1,arg2} = p_0, \ldots, p_n$ that connect the entities $arg1, arg2$ in Freebase.

(b) Build final lexicon: Compute the probability $p(r, p)$. Tables 8 and 9 show the result of $p(''creator'', p)$ for semantic class propositions and predicate propositions, respectively. For instance, the probability of the semantic class $''creator''$ with the type `film.writer`, that is:

$$\begin{aligned}
&p(''creator'', ''film.writer'') = p(''film.writer'' \mid m.035sc2)\cdot \\
&\quad\cdot p(m.035sc2 \mid ''creator'') \cdot p(''creator'')+ \\
&\quad+ p(''film.writer'' \mid m.02sbwl) \cdot p(m.02sbwl \mid ''creator'')\cdot \\
&\quad\cdot p(''creator'') + \ldots = \\
&\quad= \frac{1}{8} \cdot \frac{1}{35022} \cdot \frac{35022}{193163} + \frac{1}{4} \cdot \frac{1}{35022} \cdot \frac{35022}{193163} + \ldots = 3.45E-5
\end{aligned}$$

Semantic Class	Type	$p(r, p)$
creator	`organization.organization_founder`	3.45E-5
	`business.board_member`	7.30E-6
	`people.person`	6.68E-6
	`film.writer`	5.90E-6
	`film.director`	4.30E-6
	...	

Table 8: Relevant probabilities for the semantic class *creator*. Probabilities are obtained after processing the whole collection.

Predicate	Property Argument 1	Property Argument 2	$p(r, p)$
creator	`film.written_by.1`	`film.written_by.2`	9.06E-6
	`organization.founders.1`	`organization.founders.2`	7.77E-6
	`employment_tenure.person`	`employment_tenure.company`	7.77E-6
	`film.directed_by.1`	`film.directed_by.2`	6.47E-6
	...		

Table 9: Relevant probabilities for a proposition with the predicate *creator*. We removed the domains of the properties for readability. Probabilities are obtained after processing the whole collection.

3. Lexical Expansion

(a) Create new pairs: Search in WordNet for synonyms of a predicate. For example, for the predicate *create* we find the synonyms *make* and *produce*. Properties that would be paired with the original predicate are paired with the synonyms $S_r = \{s_0 = make, s_1 = produce, s_2 = create\}$.

(b) Compute the probability $p'(s, p)$. For instance, the probability $p'(\text{"}creator\text{"}, p)$ is computed as:

$$p'(\text{"}creator\text{"}, p) = \frac{p(\text{"}make\text{"}, p)}{3} + \frac{p(\text{"}produce\text{"}, p)}{3} + \frac{p(\text{"}create\text{"}, p)}{3}$$

5. Experiment Design

In this section we explain the semantic parsing task and the datasets that we have used both for the creation of the GPS and for testing them for the task of semantic parsing.

Formally, our goal is to learn a function to map an utterance u to a query q over a database D. The database is defined by a schema that contains properties $p \in P$ and entities $e \in E$. Both properties and entities are human-readable strings like `film.directed_by` or `David_Cameron`. The database contains a set of triples $\{e_1 - p - e_2\}$. For each utterance (natural language question) the system gets set of SPARQL queries $Q_u = q_0 \dots q_n$. Each query executed over the database obtains a set of answers $A_q = a_0, \dots, a_m$.

5.1. Implementation

We took advantage of GraphParser to evaluate our approach against the test collections. We use the Maximum Weighted Graph (MWG) configuration, a baseline that replaces each predicate with the property with the highest probability without any further training. With the goal of measuring the contribution of each syntactic pattern, we perform an ablation test where we remove one by one a syntactic pattern in the GPS building process and compare the resulting GPS with the full building process. Moreover, we experiment with the expansion of propositions using WordNet and perform an additional ablation test on the expanded GPS. Then, we try to maximise our results by removing the harmful patterns, and finally, we perform a comparison with an state of the art system.

5.2. Evaluation Measures

Following [14, 15, 6] and many others, the system is evaluated using precision, recall and F1-measure.

$$Precision = \frac{\text{number of correct system answers}}{\text{number of system answers}} \tag{3}$$

$$Recall = \frac{\text{number of correct system answers}}{\text{number of questions}} \tag{4}$$

These measures consider only the first answer of the ranking. An answer of a query is correct if contains exactly the same responses that the gold standard. Partial answers are considered mistakes. Results are ranked according to the F1-measure.

5.3. Dataset

Our dataset is a subset of WebQuestions [15] as defined in [6]. The scope is reduced to three domains: film, business and people. The final dataset is composed by 200 questions devoted for development and 570 questions for testing. Note that, with the MWG configuration, the development dataset is not used.

6. Results

We compare the effect of our lexicon against having no lexicon at all to highlight how determinant is the grounding step. With a GPS, results are pushed 25.3 points with respect to the empty lexicon, showing the high impact that knowledge acquisition has in the task (Table 10). We also show that the baseline proposed in GraphParser is already informed. MWG uses the default lexicon of GraphParser to ground logic forms without any training. Compared with the empty lexicon, results are pushed 28.7 points, which indicates that the lexicon contributes with a 78.6% of the total result of MWG. When compared to our regular GPS, the contribution is similar, with a 76.4% of the total result.

	Prec	Rec	F1	difference
Empty lexicon	8.40	7.30	7.80	
GPS	35.74	30.95	33.14	+25.34 (76.46%)
GraphParser - MWG	39,4	34,0	36,5	+28.70 (78.63%)

Table 10: Comparison between an empty lexicon, the regular GPS and GraphParser's baseline.

Ablation Test: Table 11 shows the results of the ablation test, divided between regular and expanded system. The GPS row corresponds to the full system, while the remaining rows correspond to individual ablations where a single proposition type is removed. We observe that in both cases the F1-measure of the full system is close to the highest result. Moreover, the difference between these systems and most of ablations is small, a range of [-0,31,+0,16] in the regular case and [-0,35,+0,12] for the expanded case. The VNPN, NVN y NVPN patterns are the exception, with higher loses, up to 12.64 points. This means that these structures are essential to acquire knowledge in the context of linked data, as they gather information that is unavailable for other patterns. Considering individual ablations, the best result for the regular case is to ignore hasClass+NPN structures and for the expanded case the best results are obtained by ignoring NNV or VNN structures in the construction of the proposition stores. These results show that these patterns introduce more noise than useful information.

	Regular			Expanded		
	Prec	Rec	F1	Prec	Rec	F1
GPS	35.74	30.95	33.14	38.08	32.99	35.38
-NNV	35.82	31.02	33.22 (+0.08)	**38.20**	**33.10**	**35.50 (+0.12)**
-VNN	35.68	30.89	33.09 (-0.05)	38.16	33.06	35.46 (+0.08)
-NPN	35.46	30.7	32.88 (-0.26)	38.08	32.98	35.38 (0.00)
-NisN	35.48	30.71	32.90 (-0.24)	38.06	32.97	35.36 (-0.02)
-NhasClassN+NPN	**35.90**	**31.10**	**33.30 (+0.16)**	38.04	32.94	35.34 (-0.04)
-NhasN+NPN	35.82	31.02	33.22 (+0.08)	38.04	32.94	35.34 (-0.04)
-NhasN	35.82	31.02	33.22 (+0.08)	38.02	32.93	35.32 (-0.06)
-VPNPN	35.74	30.94	33.14 (0.00)	37.84	32.77	35.15 (-0.23)
-NisN+NPN	35.67	30.46	32.83 (-0.31)	37.8	32.74	35.11 (-0.27)
-NhasClassN	35.48	30.71	32.89 (-0.25)	37.96	33.39	35.03 (-0.35)
-VNPN	28.58	24.69	26.48 (-6.66)	30.92	26.76	28.72 (-6.66)
-NVN	26.74	23.14	24.74 (-8.40)	27.54	23.84	25.54 (-9.84)
-NVPN	23.48	20.38	21.78 (-11.36)	25.10	20.97	22.74 (-12.64)

Table 11: Experimental results for the ablation test. We report the difference between the ablation and the GPS with and without expansion.

Lexical Expansion: Table 12 shows that the expanded GPS consistently outperform the regular GPS both in the full system and in every ablation, with a contribution of 0.8% in the worst case and 2.5% in the best. This confirms that GPS can be effectively expanded using WordNet synonyms in order to reduce the lexical gap, and points out that external resources can be a good complement to distant-supervised methods to acquire knowledge. In other words, the more knowledge we inject into the system the better performance it shows.

	Regular	Expanded	Difference
GPS	33.14	35.38	+2.24
-NPN	32.88	32.98	+2.50
-NisN	32.90	32.97	+2.46
-VNN	33.09	33.06	+2.37
-NNV	33.22	33.10	+2.28
-NisN+NPN	32.83	32.74	+2.28
-VNPN	26.48	26.76	+2.24
-NhasClassN	32.89	33.39	+2.14
-NhasN+NPN	33.22	32.94	+2.12
-NhasN	33.22	32.93	+2.10
-NhasClassN+NPN	33.30	32.94	+2.04
-VPNPN	33.14	32.77	+2.01
-NVPN	21.78	20.97	+0.96
-NVN	24.74	23.84	+0.80

Table 12: Comparison between F1 measure for the Regular and Expanded GPS.

Best Configuration: Table 13 shows the results of the configurations that remove harmful patterns, which are NhasClassN+NPN, NhasN, NhasN+NPN and NNV for the regular configuration and NNV and VNN for the expansion. Results indicate that these patterns can be omitted with a further small boost on the results. More significant, we can achieve the same performance with less patterns, which in turn means less computational cost of building the lexicons.

	Regular			Expanded		
	Prec	Rec	F1	Prec	Rec	F1
GPS	35.74	30.95	33.14	38.08	32.99	35.38
(1)	**35.86**	**31.06**	**33.26 (+0.12)**	**38.22**	**33.12**	**35.52 (+0.14)**
(2)	**35.86**	**31.06**	**33.26 (+0.12)**	38.18	33.08	35.48 (+0.1)

Table 13: Experimental results removing the harmful ablations. (1) Corresponds to NNV and VNN patterns which are harmful for the expanded GPS and (2) corresponds to NhasClassN+NPN, NhasN, NhasN+NPN and NNV, which are harmful for the regular GPS.

Comparative Evaluation: We compare our salient configurations with GraphParser's baseline (MWG) and system with training (GraphParser). Table 14 shows how we achieve similar results, which is promising considering that we limited the acquisition to the sentences used in GraphParser. Our representation relies on syntactic parsing and syntactic rules to process sentences, which is faster than the CCG parser, so there is potential to increase the acquisition by using a larger document collection. Note that GraphParser's default and baseline configuration have 2.8 points of difference, which represents a 7.6% of relative improvement. Again, this highlights the importance of the grounding.

	Prec	Rec	F1
GraphParser	41,9	37,0	39,3
GraphParser - MWG	39,4	34,0	36,5
GPS expanded-(1)	38,22	33,12	35,52
GPS expanded	38,08	32,99	35,38
GPS	35,74	30,95	33,14

Table 14: Comparison between GraphParser and the regular and expanded GPS, plus our best system which is the NNV and VNN ablation of the expanded system (1). MWG refers to GraphParser's baseline configuration where there is no training.

7. Conclusions

Question Answering systems in the state of the art are evaluated as a monolithic system that involves acquisition, learning and querying without measuring the contribution of each component. However, such a system level comparison does not provide any insight into the real contribution of each component, and, in particular, the effect of the amount of knowledge digested in the final result.

We show here that the main component is the lexicon itself (the GPS in our case), so we need better ways of creating and evaluating this resource before addressing the learning and querying steps in deeper and more sophisticated settings.

We have presented both the methodology to generate a GPS linked to a particular knowledge base, and a study evaluating the effect in QA performance that different natural language structures produce when they are considered to build the GPS.

For this reason, we evaluated the construction and grounding of the lexicon (GPS in our case) *per se*, without additional training or wiring to the SPARQL queries generation. This additional evaluation, such as done in [14, 15, 6], is out of the scope of this work. Different methods for training and querying must be evaluated once the GPS is fixed, so we can learn about the effect of different techniques.

We summarize our conclusions to the following research questions:

- *What are the methodological steps to build a GPS?*

  We have developed a method to map propositions into Freebase properties using distant supervision that helps in solving both lexical and structural gaps by finding multiple grammatical structures and lexical realizations of the same query.

  Our method is divided in three steps: (1) Build a proposition store. To do so, we select relevant sentences, transform them into graphs from which we extract propositions. (2) Ground each proposition by pairing them with KB properties considering linked entities, and (3) Compute the global weights of each pairing.

  We consider an optional step, (4) Perform a lexical expansion by creating new pairs and re-evaluate the weights of the lexicon entries.

- *What is the impact of the GPS when used to fed a semantic parser for question answering?*

  Building and grounding the proposition stores is key to the final performance of the semantic parser. A system with an empty alignment lexicon achieves a 7.80% of F1-measure. In baseline systems with lexicon but without training, our experiments show that the lexicon contributes with near 80% of the results, and training only accounts for 7.6% of relative improvement.

- *What linguistic phenomena (syntactic-semantic relations) should be considered in the knowledge acquisition step?*

  We have analysed different linguistic structures that can be included in the GPS and what is the contribution on the final result. For this setting, NVN, NVPN and VNPN patterns have a significant effect in the performance. Our results suggest that extensive coverage of every possible syntactic pattern is not as useful as it may be intuitive. Conversely, systems can dispense with some patterns and reduce the computational cost of building the alignment.

- *Are external linguistic resources useful for enriching the GPS?*

We have shown how to enrich the lexicon using linguistic information from external resources, helping to bridge the lexical gap between utterances and database queries. Enrichment consistently pushes the results in every case, in a range from 0.8% to 2.50%.

8. Future work

This work opens many new research questions. With the learned lessons, we could aim to scale up the system. One option would be to generalize the GPS using bigger knowledge bases, which now are small because entities are required to be linked to Freebase. The hypothesis of *one sense per collocation* defined in [27] could help to automate entity linking.

We leave also for future work refining the distant supervision process. For example, we could follow a similar approach as [28], that try to reduce the noise by jointly modelling instances and labels for relation extraction. This would be equivalent to our problem, where we could model utterances and labels at the same time.

Finally, it would be interesting to study how to improve GPS using distributed representations at word level like Word2Vec [29] or GloVe [30], at relation level [31, 32], or embeddings for QA [33, 34]. Whereas they have proven to be useful for training [35, 36], to the best of our knowledge there is not any effort to enrich the acquisition step.

Acknowledgements

We are grateful to Siva Reddy for sharing GraphParser and the evaluation data. This work has been partially supported by the Spanish Government(MINECO) in the framework of CHIST-ERA program (READERS project), and TUNER project (TIN2015-65308-C5-1-R). Anselmo Peñas received the support of the Spanish Government (grant PRX15/00269, Subprograma Estatal de Movilidad del Plan Estatal de Investigación Cientfica, Técnica y de Innovación 2013-2016).

References

[1] C. Bizer, T. Heath, T. Berners-Lee, Linked data-the story so far, Semantic Services, Interoperability and Web Applications: Emerging Concepts (2009) 205–227.

[2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, ACM, 2008, pp. 1247–1250.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, Dbpedia: A nucleus for a web of open data, Springer, 2007.

[4] J. Hoffart, F. M. Suchanek, K. Berberich, G. Weikum, Yago2: A spatially and temporally enhanced knowledge base from wikipedia, in: Proceedings of the Twenty-Third international joint conference on Artificial Intelligence, AAAI Press, 2013, pp. 3161–3165.

[5] H. Poon, Grounded unsupervised semantic parsing., in: ACL (1), 2013, pp. 933–943.

[6] S. Reddy, M. Lapata, M. Steedman, Large-scale semantic parsing without question-answer pairs, Transactions of the Association for Computational Linguistics 2 (2014) 377–392.

[7] B. F. Green, Jr., A. K. Wolf, C. Chomsky, K. Laughery, Baseball: An automatic question-answerer, in: Papers Presented at the May 9-11, 1961, Western Joint IRE-AIEE-ACM Computer Conference, IRE-AIEE-ACM '61 (Western), ACM, New York, NY, USA, 1961, pp. 219–224. doi:10.1145/1460690.1460714.

[8] J. M. Zelle, R. J. Mooney, Learning to parse database queries using inductive logic programming, in: Proceedings of the National Conference on Artificial Intelligence, 1996, pp. 1050–1055.

[9] L. S. Zettlemoyer, M. Collins, Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars, in: In Proceedings of the 21st Conference on Uncertainty in AI, 2005, pp. 658–666.

[10] S. Hakimov, C. Unger, S. Walter, P. Cimiano, Applying semantic parsing to question answering over linked data: Addressing the lexical gap, in: International Conference on Applications of Natural Language to Information Systems, Springer, 2015, pp. 103–109.

[11] D. L. Chen, R. J. Mooney, Learning to interpret natural language navigation instructions from observations, San Francisco, CA (2011) 859–865.

[12] Y. Artzi, L. Zettlemoyer, Bootstrapping semantic parsers from conversations, in: Proceedings of the conference on empirical methods in natural language processing, Association for Computational Linguistics, 2011, pp. 421–432.

[13] Q. Cai, A. Yates, Large-scale semantic parsing via schema matching and lexicon extension., in: ACL (1), Citeseer, 2013, pp. 423–433.

[14] T. Kwiatkowski, E. Choi, Y. Artzi, L. Zettlemoyer, Scaling semantic parsers with on-the-fly ontology matching.

[15] J. Berant, A. Chou, R. Frostig, P. Liang, Semantic parsing on freebase from question-answer pairs., in: EMNLP, 2013, pp. 1533–1544.

[16] J. Berant, P. Liang, Semantic parsing via paraphrasing, in: Proceedings of ACL, Vol. 7, 2014, p. 92.

[17] A. Fader, L. S. Zettlemoyer, O. Etzioni, Paraphrase-driven learning for open question answering., in: ACL (1), Citeseer, 2013, pp. 1608–1618.

[18] A. Fader, L. Zettlemoyer, O. Etzioni, Open question answering over curated and extracted knowledge bases, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2014, pp. 1156–1165.

[19] S. Clark, J. R. Curran, Parsing the wsj using ccg and log-linear models, in: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics, 2004, p. 103.

[20] J. Callan, M. Hoy, C. Yoo, L. Zhao, Clueweb09 data set (2009).

[21] E. Gabrilovich, M. Ringgaard, A. Subramanya, Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0), Note: http://lemurproject. org/clueweb09/FACC1/Cited by 5.

[22] D. Klein, C. D. Manning, Accurate unlexicalized parsing, in: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1, Association for Computational Linguistics, 2003, pp. 423–430.

[23] H. Lee, Y. Peirsman, A. Chang, N. Chambers, M. Surdeanu, D. Jurafsky, Stanford's multi-pass sieve coreference resolution system at the conll-2011

shared task, in: Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task, Association for Computational Linguistics, 2011, pp. 28–34.

[24] M.-C. De Marneffe, B. MacCartney, C. D. Manning, et al., Generating typed dependency parses from phrase structure parses, in: Proceedings of LREC, Vol. 6, 2006, pp. 449–454.

[25] G. A. Miller, WordNet: a lexical database for English, Communications of the ACM 38 (11) (1995) 39–41.

[26] S. Walter, C. Unger, P. Cimiano, D. Bär, Evaluation of a layered approach to question answering over linked data, in: The Semantic Web–ISWC 2012, Springer, 2012, pp. 362–374.

[27] A. Barrena, E. Agirre, B. Cabaleiro, A. Peñas, A. Soroa, "one entity per discourse" and "one entity per collocation" improve named-entity disambiguation, in: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, Dublin City University and Association for Computational Linguistics, Dublin, Ireland, 2014, pp. 2260–2269.
URL `http://www.aclweb.org/anthology/C14-1213`

[28] M. Surdeanu, J. Tibshirani, R. Nallapati, C. D. Manning, Multi-instance multi-label learning for relation extraction, in: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL '12, Association for Computational Linguistics, Stroudsburg, PA, USA, 2012, pp. 455–465.

[29] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in neural information processing systems, 2013, pp. 3111–3119.

[30] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation., in: EMNLP, Vol. 14, 2014, pp. 1532–1543.

[31] M. Lewis, M. Steedman, Combining distributional and logical semantics, Transactions of the Association for Computational Linguistics 1 (2013) 179–192.

[32] Y. Ji, J. Eisenstein, One vector is not enough: Entity-augmented distributed semantics for discourse relations, Transactions of the Association for Computational Linguistics 3 (2015) 329–344.

[33] G. Zhou, T. He, J. Zhao, P. Hu, Learning continuous word embedding with metadata for question retrieval in community question answering, in: Proceedings of ACL, 2015, pp. 250–259.

[34] G. Zhou, Y. Zhou, T. He, W. Wu, Learning semantic representation with neural networks for community question answering retrieval, Knowledge-Based Systems 93 (2016) 75–83.

[35] A. Kumar, O. Irsoy, J. Su, J. Bradbury, R. English, B. Pierce, P. Ondruska, I. Gulrajani, R. Socher, Ask me anything: Dynamic memory networks for natural language processing, CoRR abs/1506.07285.

[36] Y. Li, P. Clark, Answering elementary science questions by constructing coherent scenes using background knowledge, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 2007–2012.