Natural Language Question Answering and Analytics for Diverse and Interlinked Datasets

Dezhao Song **Frank Schilder** **Charese Smiley**
Research and Development, Thomson Reuters
610 Opperman Drive
Eagan, MN 55123, USA

Chris Brew
Research and Development, Thomson Reuters
1 Mark Square
London, UK
{dezhao.song, frank.schilder, charese.smiley, chris.brew}@thomsonreuters.com

Abstract

Previous systems for natural language questions over complex linked datasets require the user to enter a complete and well-formed question, and present the answers as raw lists of entities. Using a feature-based grammar with a full formal semantics, we have developed a system that is able to support rich autosuggest, and to deliver dynamically generated analytics for each result that it returns.

1 Introduction

In order to retrieve data from a knowledge base (KB), knowledge workers, such as physicians or financial analysts, often face the challenge of having to learn specific query languages (e.g., SQL and SPARQL[1]). However, the fast pace of changing query languages to different types of KBs (e.g., Relational Databases, Triple Stores, NoSQL stores, etc.) makes it difficult for users to keep up with the latest developments of such query languages that allow them to access the data they need for their work. This situation prevents users without extensive computer training from effectively utilizing the available information in the KB. Developing user-friendly natural language interfaces will make it easier for non-technical users to access the information in the KB in an intuitive way.

In this paper, we present a Natural Language Interface that allows users to query the underlying KBs with natural language questions. Unlike previous approaches, instead of asking the users to provide the entire question on their own, our system makes suggestions to help the users to complete their questions. Given a complete question, our system parses it to its First Order Logic (FOL) representation using a grammar derived from interlinked datasets; different translators are developed to further translate the FOL of a query into executable queries, including both SQL and SPARQL. Finally, our system generates dynamic analytics for the result sets in order to help users to gain a better understanding of the data.

2 Related Work

Keyword-based search (Ding et al., 2004; Tummarello et al., 2007; d'Aquin and Motta, 2011) and faceted search (Zhang et al., 2013; Zhang et al., 2014) have been frequently adopted for retrieving information from KBs. However, users have to figure out the most effective queries in order to retrieve relevant information. Furthermore, without appropriate ranking methods, users may be overwhelmed by the information available in the search results.

Early Natural Language Interfaces (NLIs) required a handcrafted interface solution for each database thereby restricting its portability (Green et al., 1961; Hendrix et al., 1978; Woods, 1973). Recent research has focused more on developing open domain systems (Kwiatkowski et al., 2013; Yao and Durme, 2014; Bordes et al., 2014), but there remains a need for specialized NLIs (Minock, 2005). One unique feature of our system is to help users to build a complete question by providing suggestions according to a partial question and a grammar.

Much of prior work translates a natural language question into SPARQL and retrieves answers from a

[1] http://www.w3.org/TR/rdf-sparql-query/

Proceedings of NAACL-HLT 2015, pages 101–105,
Denver, Colorado, May 31 – June 5, 2015.

triple store (Lopez et al., 2005; Unger et al., 2012; Lehmann et al., 2012; Yahya et al., 2013; He et al., 2014); however, SPARQL queries have been criticized to have unsatisfying query response time. In this work, we maintain flexibility by first parsing a question into First Order Logic, which is further translated into both SQL and SPARQL. This enables us to easily adapt to new query languages and allows us to choose the most appropriate query language technology for a given use case.

Finally, to the best of our knowledge, none of existing NLIs provide dynamic analytics for the results. Our system performs descriptive analytics and comparisons on various dimensions of the data, conducts sentiment analysis, and analyzes trends over time in the data. Such analytics would enable users to better conduct further analyses and derive insights from the data. This feature of our system is a clear advantage over other NLI systems that only retrieve a simple result list of documents/entities.

3 Overall Architecture

Figure 1 shows the overall architecture of our proposed NLI system. Users can input their questions

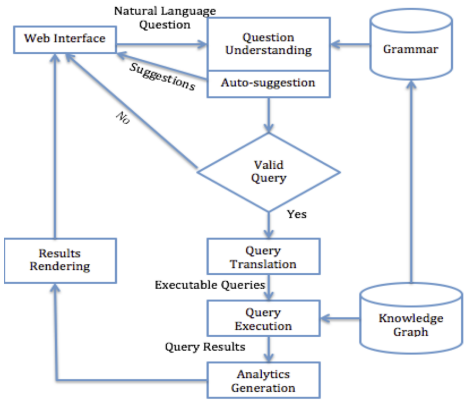

Figure 1: System Architecture

on the Web Interface and our Auto-suggestion component will guide the users in completing their questions. A complete question is then sent to the Question Understanding module again to be parsed into its first order logic representation with the grammar. As the next step, the FOL of a query is translated into an executable query with the Query Translation module. A translated query is then executed against an underlying knowledge base/graph for retrieving answers and generating corresponding analytics.

Our system currently focuses on the following domains: Drugs, Organizations, Patents, People, Finance and News. The underlying knowledge base contains about 1 million entities and 12 million relationships.

4 Question Understanding

Our system utilizes a feature-based context-free grammar (FCFG) that consists of grammar rules on non-terminal nodes and lexical rules on leaf nodes. Grammatical entries on non-terminal syntactic nodes are largely domain-independent, thus enabling our grammar to be easily adaptable to new domains. Each lexical entry to the grammar contains domain-specific features which are used to constrain the number of parses computed by the parser preferably to a single, unambiguous parse.

The following are two rules in our grammar.

1. N[TYPE=drug, NUM=pl, SEM=$<\lambda x.\text{drug}(x)>$] → 'drugs'
2. V[TYPE=[org,drug],SEM=$\lambda X x.X(\lambda y.\text{develop\_org\_drug}(x,y))>$, TNS=prog, NUM=?n] → 'developing'

Rule 1 shows a lexical entry for the word *drugs*, indicating that its TYPE is *drug*, is *plural*, and has the following semantic: $\lambda x.\text{drug}(x)$. Rule 2 specifies the verb *develop*, describing its tense (TNS) and indicating that it connects an *organization* and a *drug* via the TYPE feature. By utilizing the type constraints, we can then license the query *companies developing drugs* while rejecting nonsensical queries like *rabbits develop drugs* on the basis of the mismatch in semantic type. Furthermore, our grammar also covers wh-questions, e.g., *what, which, how many, where*, and nominal phrases and imperatives.

Disambiguation relies on the presence of features on non-terminal syntactic nodes. We mark prepositional phrases (PPs) with features that determine their attachment preference. E.g., the PP *for pain* in *how many companies develop drugs for pain?* must attach to an NP rather than a VP; thus, it must attach to *drugs* rather than *develop*. Together with other features, we filter out many of the logically possible but undesired PP-attachments in queries with many modifiers. E.g., our approach is able to generate a single parse for *companies headquartered in Germany developing drugs for pain or cancer*.

5 Auto-suggestion

Our NLI provides suggestions to help users to complete their questions. Unlike Google's query autocompletion that is based on query logs (Cornea and Weininger, 2014), our auto-suggestion utilizes the linguistic constraints encoded in the grammar.

Our auto-suggestion is based on the idea of left-corner parsing. Given a query segment qs (e.g., *drugs*, *developed by*, etc.), we find all grammar rules whose left corner fe on the right side matches the left side of the lexical entry of qs. We then find all leaf nodes in the grammar that can be reached by using the adjacent element of fe. For all reachable leaf nodes (i.e., lexical entries in our grammar), if a lexical entry also satisfies all the linguistic constraints, we then treat it as a valid suggestion.

Specifically, for the query segment *Drugs*, according to our grammar, we could be looking for a verb as the next part of the question. In our lexicon, we may have many verbs, e.g., *drive* and *developed by*. Here, *developed by* is a valid suggestion because its semantic constraints match that of drugs. We continue our suggestions to the end of the user-entered query string, and never try to interpolate material either before or inside the string.

In our current system, the automatically generated suggestions are ranked by considering their popularity. We associate each lexical entry with a node in a knowledge graph. This graph contains nodes for the entities corresponding to the lexical entries, further nodes for generic types such as Drug, Company and Technology, and yet further nodes for predicates such as *developed by* and *granted to*.The edges of the graph represent relations such as *developed by* and *filed by*. For ranking, the degree of a node is as a proxy for its quality. For example, if the node "Google" filed 10 patents and is also involved in 20 lawsuits, then its popularity will be 30.

6 Query Translation and Execution

The interpreted FOL (Section 4) of a question is further analyzed by another parser (implemented with ANTLR (Bovet and Parr, 2008)) that parses FOL expressions. Figure 3 shows the parse tree of the FOL for the query *Drugs developed by Merck*. We then traverse this parse tree, and put all the atomic logical conditions and the logical connectors into a

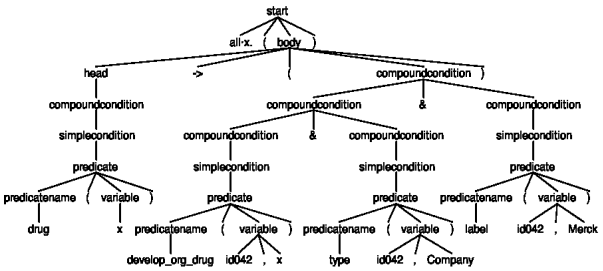

Figure 3: Parse Tree for the First Order Logic Representation of the Query "Drugs developed by Merck"

stack. When we finish traversing the entire tree, we pop the conditions out of the stack to build the query constraints; predicates in the FOL are also mapped to their corresponding attribute names (SQL) or ontology properties (SPARQL).

The following summarizes the translation from a natural language question to a SQL and SPARQL query via a FOL representation:

Natural Language: ``Drugs developed by Merck''

First Order Logic (FOL) Representation:
```
all x.(drug(x) → (develop(id042,x) & type(id042,Company) & label(id042,Merck)))
```

SQL Query:
```
select drug.* from drug
  where drug.originator_company = 'Merck'
```

SPARQL Query (prefixes for RDF and RDFS omitted):
```
PREFIX example:  <http://www.example.com#>
select ?x ?id123 ?id042
where {
?id042 rdfs:label 'Merck'.
?id042 rdf:type example:Company .
?x rdf:type example:Drug .
?id042 example:develops ?x .  }
```

We execute the SQL queries using Apache Spark (Zaharia et al., 2010), a distributed computing environment, thus providing us the potential to handle large-scale datasets. We run SPARQL queries with Jena (Carroll et al., 2004). If a question cannot be parsed into FOL or translated to SQL or SPARQL, we then treat it as a keyword query and retrieve the results from an inverted index built out of our data.

7 Analytics

Instead of only retrieving a list of entities, our system provides several different types of analytics for different result sets. In many situations, the result is a set of records rather than one single entry. This

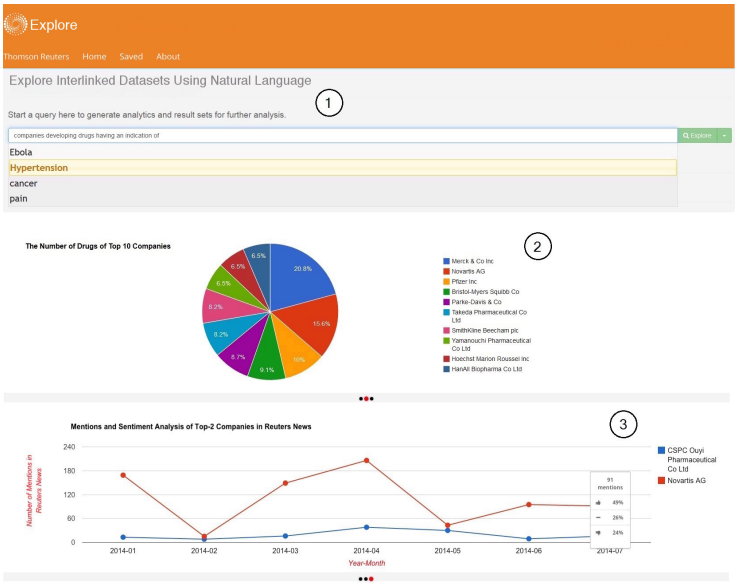

Figure 2: System Screenshot

provides us the opportunity to perform and provide further analyses of the result set for the users.

Our system provides several types of analytics. Descriptive analytics summarize the facts in the result set. For instance, for the question "show me all drugs targeting pain", our system shows the distribution of all technologies used for such drugs in the result set. We also compare the drugs in the result set on different dimensions (e.g., diseases). Moreover, we compute trends via exponential smoothing for entities that have a temporal dimension.

By linking entities from our KB to entity mentions in a large news corpus (14 million articles and 147 million sentences), we are able to perform additional analytics based on named entity recognition and sentiment analysis techniques. We adopted the Stanford CoreNLP toolkit (Manning et al., 2014) for recognizing person, organization, and location from the news corpus. Given an entity, we show its frequency count and how its sentiment may change over time. This information may provide further insights to users in order to support their own analysis.

8 Demonstration Script Outline

Figure 2 shows the beginning of the sample query: *companies developing drugs having an indication of ...?* While the user is typing, a variety of possible extensions to the query are offered, and the user selects *Hypertension* (1). Our system shows a pie chart of each company's market share for hypertension drugs (2); we also show news mentions and sentiment analysis for the most discussed companies (3).

For the demo, we will first motivate the use of natural language question answering for extracting information from complex, interlinked datasets. Next, we will demonstrate how the user can compose a variety of questions with auto-suggestion. Finally, we will walk through the generated analytics and various visualizations for different natural language questions in order to show how it allows the user to gain deeper insights into the data.

9 Conclusion and Future Work

In this paper, we presented a Natural Language Interface for answering complex questions over linked data. Our system parses natural language questions to an intermediate logical representation based on a grammar derived from multiple interlinked datasets. Different translators are developed to translate a question from its FOL representation to SQL and SPARQL queries, which are then executed against an underlying knowledge graph/base for retrieving the answers and generating corresponding analytics. In future work, we intend to cover more domains and provide more complex analytics. We will also perform a thorough evaluation of our system.

References

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 615–620.

Jean Bovet and Terence Parr. 2008. Antlrworks: an ANTLR grammar development environment. *Software: Practice and Experience*, 38(12):1305–1332.

Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. 2004. Jena: implementing the semantic web recommendations. In *Proceedings of the 13th international conference on World Wide Web - Alternate Track Papers & Posters*, pages 74–83.

Radu C Cornea and Nicholas B Weininger. 2014. Providing autocomplete suggestions, February 4. US Patent 8,645,825.

Mathieu d'Aquin and Enrico Motta. 2011. Watson, more than a semantic web search engine. *Semantic Web Journal*, 2(1):55–63.

Li Ding, Timothy W. Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, and Joel Sachs. 2004. Swoogle: a search and metadata engine for the semantic web. In *Proceedings of the 2004 ACM International Conference on Information and Knowledge Management*, pages 652–659.

Bert F. Green, Jr., Alice K. Wolf, Carol Chomsky, and Kenneth Laughery. 1961. Baseball: An automatic question-answerer. In *Papers Presented at the Western Joint IRE-AIEE-ACM Computer Conference*, pages 219–224.

Shizhu He, Kang Liu, Yuanzhe Zhang, Liheng Xu, and Jun Zhao. 2014. Question answering over linked data using first-order logic. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1092–1103.

Gary G. Hendrix, Earl D. Sacerdoti, Daniel Sagalowicz, and Jonathan Slocum. 1978. Developing a natural language interface to complex data. *ACM Transactions on Database Systems*, 3(2):105–147.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke S. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556.

Jens Lehmann, Tim Furche, Giovanni Grasso, Axel-Cyrille Ngonga Ngomo, Christian Schallhart, Andrew Jon Sellers, Christina Unger, Lorenz Bühmann, Daniel Gerber, Konrad Höffner, David Liu, and Sören Auer. 2012. DEQA: Deep web extraction for question answering. In *11th International Semantic Web Conference*, pages 131–147.

Vanessa Lopez, Michele Pasin, and Enrico Motta. 2005. Aqualog: An ontology-portable question answering system for the semantic web. In *The Semantic Web: Research and Applications, Second European Semantic Web Conference*, pages 546–562.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 55–60.

Michael Minock. 2005. Where are the killer applications of restricted domain question answering. In *Proceedings of the IJCAI Workshop on Knowledge Reasoning in Question Answering*, page 4.

Giovanni Tummarello, Renaud Delbru, and Eyal Oren. 2007. Sindice.com: Weaving the open linked data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference*, pages 552–565.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over RDF data. In *Proceedings of the 21st World Wide Web Conference*, pages 639–648.

William A. Woods. 1973. Progress in natural language understanding: an application to lunar geology. In *American Federation of Information Processing Societies: 1973 National Computer Conference*, volume 42, pages 441–450.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. 2013. Robust question answering over the web of linked data. In *22nd ACM International Conference on Information and Knowledge Management*, pages 1107–1116.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 956–966.

Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2010. Spark: Cluster computing with working sets. In *2nd USENIX Workshop on Hot Topics in Cloud Computing*, pages 1–10.

Xingjian Zhang, Dezhao Song, Sambhawa Priya, and Jeff Heflin. 2013. Infrastructure for efficient exploration of large scale linked data via contextual tag clouds. In *12th International Semantic Web Conference*, pages 687–702.

Xingjian Zhang, Dezhao Song, Sambhawa Priya, Zachary Daniels, Kelly Reynolds, and Jeff Heflin. 2014. Exploring linked data with contextual tag clouds. *Journal of Web Semantics*, 24:33–39.