TR Discover: Querying Interlinked Datasets with Natural Language

Dezhao Song[1], Frank Schilder[1], Charese Smiley[1], Chris Brew[2] and Tom Zielund[1], Hiroko Bretz[1], Robert Martin[3], Chris Dale[1], Steven Pomerville[3], John Duprey[3], and Tim Miller[4]

[1]Research and Development, Thomson Reuters, Eagan, MN 55123, USA
[2] Research and Development, Thomson Reuters, London, UK
[3] Research and Development, Thomson Reuters, Rochester, NY 14694, USA
[4] Intellectual Property and Science, Thomson Reuters, London, UK
{dezhao.song,frank.schilder,charese.smiley,chris.brew,thomas.zielund,
hiroko.bretz,robertd.martin,chris.dale,steven.pomerville,john.duprey,
tim.miller}@thomsonreuters.com

Abstract. We present TR Discover, a natural language interface for querying interlinked datasets. Using a feature-based grammar, TR Discover parses a natural language question to its First Order Logic representation, which is further translated into structured query languages, including SPARQL or SQL. Because users will not necessarily know what the coverage of the system is, TR Discover offers a novel auto-suggest mechanism that can help users to construct well-formed and useful natural language questions. In addition to only showing the result set to end users, our system also performs further analytics on the result set in order to help users to obtain deeper insights into the data. We have applied TR Discover to Thomson Reuters datasets in the Life Sciences and the Legal domains.

Keywords: Natural Language Interface, Question Answering, Feature-based Grammar, Auto-Suggestion, Analytics

1 Introduction and Motivation

Imagine you are a non-technical domain expert (e.g., a journalist or patent attorney) or generally interested in a question that could be answered given the vast amount of semantic web data we have. For example, you are asking yourself how many drugs the company Pfizer currently has under development and which diseases they are currently focusing on. A keyword based search will not give you an answer, but just a list of links to documents. One could learn how to query a relational database or a triple store, but that would require learning the corresponding query languages. Therefore, one question would be: ***Can we enable non-experts to query a knowledge base using their own words (i.e., natural language), which can be understood by a computer system in order to retrieve the exact results the users are looking for?***

Furthermore, given a dataset, do users always know what types of questions they can ask? On the one hand, users may not know what types of entities are in the data. Does the underlying dataset cover companies, music, books, or drugs? Without knowing such information, it would be difficult for users to even start formulating their questions. On the other hand, by knowing the types of entities, they may not know what query conditions they can put on those entities. Assuming a pharmaceutical researcher wants to know more about drugs, do they know if they can ask for drugs for a particular disease, drugs using certain technologies, drugs from one or more specific companies, and so on? ***Rather than letting users struggle to formulate their own natural language questions, a mechanism that could guide them to build questions that are likely answerable by the underlying data would be extremely helpful.***

Presenting a result set to users is the first step. With the query results, users often times still need to perform various types of analyses on their own to really understand the data. For example, assuming a patent attorney asks "show me patents filed in China" and is presented with a huge result set of patents, what could the patent attorney learn from this? Is she going to know the most active company filing patents? Can she compare different companies/organizations by their patent filing amount? As a non-technical user, it may indeed take some time to find the right tool and to analyze the data in order to derive further insights. Thus, another important aspect would be: ***In addition to simply showing the results of a query, can the system perform some analysis on the results in order to provide some insights to the users?***

Our system, *TR Discover*, is designed as an intuitive interface between non-technical users and the underlying data. With *TR Discover*, users write questions in natural language which are then mapped into a logic-based intermediate language via a feature-based grammar with full formal semantics. We develop an auto-suggest mechanism that steers the user towards logically well-formed questions that are likely to generate useful answers from the available data. Next, the logical representation of a natural language question is further translated into an executable query (e.g., SPARQL or SQL) thereby allowing the system to use robust existing querying technologies. Finally, our system generates various types of analytics in order to help users to more easily gain insights into the data. Overall, *TR Discover* enjoys both the advantages of keyword-based search and database query systems by allowing domain experts but non-technical users to use natural language which they already know while retaining precision by mapping from the logical formalism to the query language and generating useful structured analytics. *TR Discover* is available at `http://discover.cortellislabs.com:9000`.

2 Using the TR Discover System

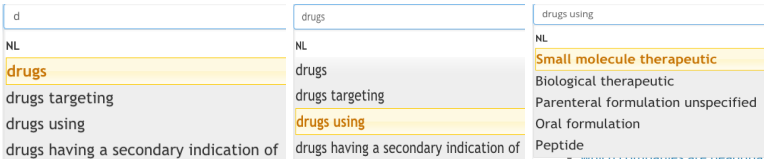
In this section, we present use cases for *TR Discover*. Due to licensing issues, our online demo only supports questions for our Life Sciences dataset. However, if selected into the second round, we will also demonstrate the system on our

Legal dataset using a local laptop. Our Life Sciences dataset, Cortellis, targets users in the Pharmaceutical industry. This dataset covers a variety of domains, including Life Sciences, Intellectual Property, Legal and Finance. In terms of entities, it contains data about drugs, companies, technologies, patents, etc.

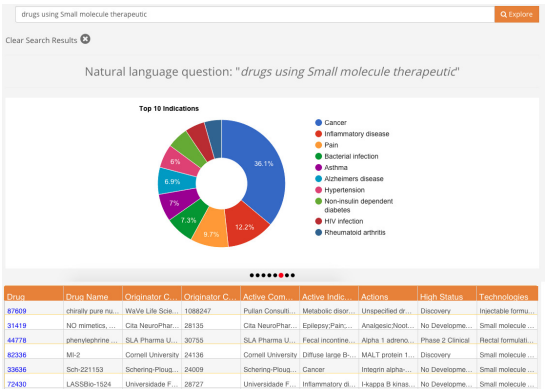
First-time Users of TR Discover. Our first use case targets first-time users of *TR Discover* or users with limited knowledge of the underlying data. This user, User A, may be interested in broad, exploratory questions; however, due to lack of familiarity with the data, guidance (from our auto-suggest module, Section 3) will be needed to help him build a valid question in order to explore the underlying data. Figures 1(a)-1(c) demonstrate this question building process. Assuming that User A starts by typing in *d*, *drugs* will then appear as a possible completion. He can either continue typing *drugs* or select it from the drop down list on the user interface. Upon selection, suggested continuations to the current question segment, such as *using* and *targeting*, are then provided to User A. Suppose our user is interested in exploring drugs that utilize certain technologies and thus selects *using*. In this case, specific technologies instances like *Small molecule therapeutic* and *Biological therapeutic* are offered as suggestions. User A can then select *Small molecule therapeutic* to build the valid question, *drugs using Small molecule therapeutic* thereby retrieving answers from our data stores along with the corresponding analytics (Figure 1(d)). In addition, users can also click the "DrugID" (i.e., the values of the first column in the result area) to examine similar drugs of a given drug as shown in Figure 1(e). Such drug similarity is computed based upon their different attributes, e.g., technologies used, diseases targeted, etc. The goal here is to help end users (e.g., pharmaceutical researchers) to gain further insights into the data.

Expert Users. The second use case targets expert professional users (e.g., medical professionals, financial analysts, or patent officers). This user, User B, understands the domain, and has specific questions in mind that may require material from multiple slices of data. She need not be concerned with how the data is partitioned across database tables because she is sheltered from this level of implementation detail. Suppose User B works for a pharmaceutical company and is interested in searching for patents relevant to a particular line of drug development. Guided by our structured auto-suggest, she could pose the detailed question, *patents filed by companies developing drugs targeting PDE 4 inhibitor using Small molecule therapeutic that have already been launched*. Our system returns 12 patents for this question and from the generated analytics (Figure 2), she can immediately see a general view of the competitive field. User B can then drill further into the patents, and begin to develop a strategy that navigates around potential infringements of her competitors' protected rights, for example.

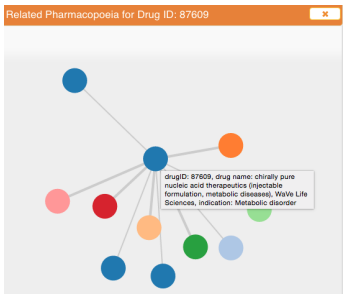
The Legal Dataset. Although we cannot release this dataset as part of the online demo due to licensing issues, we plan to demonstrate *TR Discover* on this dataset during the conference on a local laptop. This dataset consists of more than 11 million legal cases, attorneys, law firms, judges, etc. Here, users may be interested in legal cases in a specific location or on a specific legal topic. Users



(a) "d" is typed (b) "drugs" is selected and suggestions are provided (c) "using" is picked and "Small molecule therapeutic" can be chosen to complete a question



(d) Query Results and Analytics



(e)

Fig. 1. Use Case: First-time Users of TR Discover

(e.g., attorneys) may also be interested in the cases associated with a certain judge. We list a few sample questions below supported by our data:

- Cases tried in Minnesota
- Law firms representing cases tried in Minnesota containing headnotes on the topic of Criminal Law
- Cases presided over by Hon. Jane Doe

3 System Description

Question Understanding. In *TR Discover*, we parse natural language questions by adopting a feature-based context-free grammar (FCFG). Our FCFG consists of a set of grammar rules that are used to understand the syntactic structure of the questions. The vast majority of these rules are domain-independent, and as such can be re-used when moving to a new domain. As shown below, *G1* - *G3* are a few sample grammar rules. Here, *G3* indicates that a verb phrase (VP) may contain a verb (V) followed by a noun phrase (NP).

G1: NP $\rightarrow$ (N')

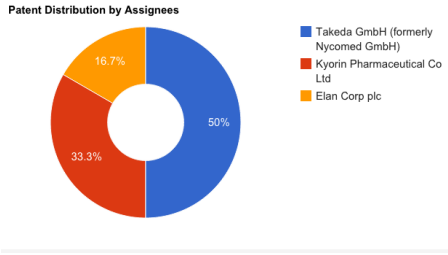


Fig. 2. Analytics for Complex Question from Professional Users

G2: NP→ NP VP
G3: VP → V NP
L1: N[TYPE=patent, NUM=pl, SEM=<λx.patent(x)>] → 'patents'
L2: V[TYPE=[patent,org,file], SEM=<λX x.X(λy.file_org_patent(y,x))>, NUM=?n] → 'filed by'
L3: V[TYPE=[drug,molecule,target], NUM=?n] → 'targeting'

The lexicon is another component of our FCFG. Each lexical entry contains a variety of domain-specific semantic features which are used to restrict the number of parses that a natural language question may have. In the above example, *L1* represents the lexical entry for *patents*, and specifies its TYPE and semantic information, SEM. Unlike nouns (*L1*), the TYPE of verbs (*L2* and *L3*) specifies both the potential subject-TYPE and object-TYPE, and the predicate name, which helps to filter out nonsensical questions like *patents developed by Anticancer*.

Auto-suggest. Left on their own, users may not know how to begin formulating questions for *TR Discover*. Therefore our system provides suggestions in order to help users to build questions that are likely to be answerable. Unlike Google's auto-completion that is based on query logs, our auto-suggest mechanism provides suggestions computed based upon the entities and their relationships in the dataset and by utilizing the linguistic constraints in our grammar.

We rank the suggestions based upon statistics extracted from an RDF graph. Each node in the RDF graph represents a lexical entry (i.e., a potential suggestion), including entities (e.g., specific drugs, drug targets, diseases, companies, and patents), predicates (e.g., *developed by* and *filed by*), and generic types (e.g., Drug, Company, Technology, etc.). The 'popularity' (i.e., ranking score) of a node is defined as the number of relationships it is involved in. For example, if a company filed 10 patents and is also involved in 20 lawsuits, then its popularity will be 30. Our current ranking is computed based only upon the data; in future work, it may be possible to tune the system's behavior to a particular individual user by mining our query logs for similar queries previously made by that user.

FOL Translation and Query Execution. Given a completed natural language question, our system first parses it into a First Order Logic representation (FOL). The FOL of a natural language question is further translated to other executable queries (e.g., SPARQL and SQL). This intermediate logical representation provides us the flexibility to develop different query translators for various types of data stores. There are two sub-steps for translating an FOL to SPARQL/SQL. We first parse the FOL into a parse tree according to an FOL

parser, implemented with ANTLR [1]. This FOL parse tree is then translated to executable queries. Finally, the translated queries are executed against their corresponding data stores, i.e., a relational database for SQL queries and a Jena TDB triple store [2] for SPARQL queries.

The following example demonstrates the process of understanding a natural language question and translating it to a SQL and SPARQL query via FOL:

Natural Language Question: `Patents filed by Pfizer`
FOL: `all x.(patent(x) → (file_org_patent(id01,x) & type(id01,Company) & label(id01,Pfizer)))`
SQL Query: `select patent.* from patent where patent.filed_by = 'Pfizer'`
SPARQL Query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX example: <http://www.example.com#>

select ?x
where {
?id01 rdfs:label 'Pfizer'.
?id01 rdf:type example:Company .
?x rdf:type example:Patent .
?x example:filed_by ?id01 .
}
```

Analytics. We provide an overview of the result set with descriptive analytics. For instance, for the question "show me all drugs developed by Pfizer Inc", we show the distribution of the technologies used by Pfizer's drugs. In addition, we perform named entity recognition (NER), using the Stanford CoreNLP toolkit [3], on the Reuters News Archive (RNA). There are about 14 million documents and the NER is done in a distributed environment using Apache Spark. The entire process took roughly 48 hours and discovered about 280 million entities.

As a second step, we ran an open-source sentiment analysis tool over the entire corpus [4]. Given an entity from the NER process, we retrieve documents from RNA that contain this entity. For each document, we then find all sentences that contain this entity and perform sentiment analysis on each of the sentences. The outcome for each sentence could be: Positive, Neutral, or Negative. Finally, we determine the overall sentiment of this document on the given entity by majority vote on the sentence-level results. By linking the recognized companies to those in our database, we show the frequency count of these companies and how their sentiment analysis results change over time. This information may provide further insights to users in order to support their own analyses.

4 Conclusion and Future Work

TR Discover was designed with non-technical information professionals in mind in order to allow them fast and effective access to large-scale interlinked datasets. Going beyond keyword-based search, *TR Discover* produces precise result sets and generates analytics for natural language questions asked by information professionals, such as journalists or patent lawyers. Rather than asking users to provide an entire question on their own, *TR Discover* provides suggestions (i.e., auto-suggest) in order to facilitate this question building process. Given a completed natural language question, *TR Discover* first parses it into its First Order

Logic (FOL) representation, by using a feature-based grammar with full formal semantics derived from interlinked datasets. By further translating the FOL representation of a natural language question into different executable queries (SPARQL and SQL), our system retrieves answers from the underlying data stores and generates corresponding analytics for the results. In future work, we plan to develop personalized auto-suggestion by using user query logs, and apply *TR Discover* on more and larger datasets to examine the response time of its various components. Furthermore, it would be interesting to seek feedback from real users on the performance and usability of our system. Finally, we plan to better handle synonyms, e.g., "medicines" for "drugs'.

References

1. Bovet, J., Parr, T.: Antlrworks: an ANTLR grammar development environment. Software: Practice and Experience 38(12), 1305–1332 (2008)
2. Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: implementing the semantic web recommendations. In: Proceedings of the 13th international conference on World Wide Web - Alternate Track Papers & Posters. pp. 74–83 (2004)
3. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. pp. 55–60 (2014)
4. Narayanan, V., Arora, I., Bhatia, A.: Fast and accurate sentiment classification using an enhanced naive bayes model. CoRR abs/1305.6143 (2013)
5. Song, D., Schilder, F., Smiley, C., Brew, C., Zielund, T., Bretz, H., Martin, R., Dale, C., Duprey, J., Miller, T., Harrison, J.: TR Discover: A natural language question answering system for interlinked datasets. In: The 14th International Semantic Web Conference (2015)

Appendix: Criteria

Minimal requirements

–The application has to be an end-user application, i.e. an application that provides a practical value to general Web users or, if this is not the case, at least to domain experts. It should show-case functionalities that the use of semantic web technologies can bring to an application. *The WebApp is available to general Web users and the auto-suggest makes it easy to construct questions. Domain experts, however, will benefit the most from the application because they can explore specific datasets based on their domain knowledge and discover new connections between various datasets.*

–The information sources used should be under diverse ownership or control; should be heterogeneous (syntactically, structurally, and semantically); and should contain substantial quantities of real world data (i.e. not toy examples). *Our online demo covers pharmaceutical data, company information, patents (but we will demonstrate using extended datasets at the conference). By linking company*

information from pharmaceutical companies to patents or legal cases we show how various heterogeneous data sets can be interlinked. We cover a substantial amount of entities and triples. The pharmaceutical data set contains 1 million entities and 12 million triples and the legal data set consists of 11 million cases.
–The meaning of data has to play a central role. Meaning must be represented using Semantic Web technologies; Data must be manipulated/processed in interesting ways to derive useful information; and this semantic information processing has to play a central role in achieving things that alternative technologies cannot do as well, or at all; *Since the natural language questions are parsed and translated into a logical representation, we rely heavily on the meaning of the data and the way users ask for information. The questions can be translated into any kind of query language including SPARQL and the result sets are complex analytics rather than a simple list of entities.*

Additional Desirable Features

–The application provides an attractive and functional Web interface (for human users) *The WebApp is easy to use and allows users to navigate through various result sets.*
–The application should be scalable (in terms of the amount of data used and in terms of distributed components working together). Ideally, the application should use all data that is currently published on the Semantic Web. *We used mainly in-house data, but also connected to DrugBank, an open dataset. Although the current focus is not the entire Semantic Web content, we could expand the grammar and translation modules to cover these datasets as well.*
–Rigorous evaluations have taken place that demonstrate the benefits of semantic technologies, or validate the results obtained. *We carried out a thorough evaluation of all components in terms of run time and performance [5].*
–Novelty, in applying semantic technology to a domain or task that have not been considered before *Using natural language questions allows for easy access to complex data sets and the ability to construct complicated questions brings together multiple data sets.*
–Functionality is different from or goes beyond pure information retrieval *In addition to retrieving the result set, we generate interesting analytics (descriptive and comparative) on the fly. We also incorporated sentiment analysis into the result set for companies mentioned in the news.*
–Contextual information is used for ratings or rankings *The frequency of the entities is used for ranking the suggestions for the auto-suggest.*
–Multimedia documents are used in some way *We deployed charts and maps for representing analytics.*
–The results should be as accurate as possible (e.g. use a ranking of results according to context) *Since we retrieve the complete set and compute aggregate information, results are very accurate.*
–There is support for multiple languages and accessibility on a range of devices *We currently only support English. In addition to traditional computers, the WebApp also runs on Android tablets and allows users to use Google voice recognition as an input method.*