USI Answers: Natural Language Question Answering Over (Semi-) Structured Industry Data

Ulli Waltinger$^\alpha$, Dan Tecuci$^\beta$, Mihaela Olteanu$^\gamma$, Vlad Mocanu$^\gamma$, and Sean Sullivan$^\delta$
Siemens AG$^\alpha$, Corporate Technology, Munich, Germany
Siemens Corporation, Corporate Technology $^\beta$, Princeton, NJ, USA
Siemens AG$^\gamma$, Corporate Technology, Brasov, Romania
Siemens Energy Inc.$^\delta$, Orlando, USA
{ulli.waltinger, dan.tecuci, mihaela.olteanu, vlad.mocanu, sean.sullivan}@siemens.com

Abstract

This paper describes *USI Answers* - a natural language question answering system for semi-structured industry data. The paper reports on the progress towards the goal of offering easy access to enterprise data to a large number of business users, most of whom are not familiar with the specific syntax or semantics of the underlying data sources. Additional complications come from the nature of the data, which comes both as structured and unstructured. The proposed solution allows users to express questions in natural language, makes apparent the system's interpretation of the query, and allows easy query adjustment and reformulation. The application is in use by more than 1500 users from *Siemens Energy*. We evaluate our approach on a data set consisting of fleet data.

Introduction

Today's enterprises need to make decisions based on analyzing massive and heterogeneous data sources. More and more aspects of business are driven by data, and as a result more and more business users need access to data. Offering easy access to the right data to diverse business users is of growing importance. There are several challenges that must be overcome to meet this goal. One is the sheer volume: enterprise data is predicted to grow by 800 percent in the next five years. The biggest part (80 percent) is stored in documents, most of them missing informative meta data or semantic tags (beyond date, size and author) that might help in accessing them. A third challenge comes from the need to offer access to this data to different types of users, most of whom are not familiar with the underlying syntax or semantics of the data.

Unified Service Intelligence (USI) is a project of Siemens Corporation, Corporate Technologies and Siemens Energy focused on generating actionable insight from large bodies of data in the energy service domain. *USI Answers*, the focus of this paper, is a sub-project of USI, focused specifically on offering easy and reliable natural language access to the large bodies of data that are used in the planning and delivery of service by Siemens Energy. The focus is on detecting and responding to events and trends more efficiently and enabling new business models.

Related Work

Natural Language Understanding (NLU) has long been a goal of AI. Considered an AI-complete task, it consists of mapping natural language sentence into a complete, unambiguous, formal meaning representation expressed in a formal language which supports other tasks such as automated reasoning, or question answering.

Natural Language access to databases (NLIDB) is a NLU task where the target language is a structured query language (e.g. SQL). NLIDB has been around for a long time, starting with the LUNAR system (Woods 1970). Early NLIDB systems took mainly a hand-built, syntax-based approach (Woods 1970; Warren and Pereira 1982; Dowding et al. 1993; Bos et al. 1996) which proved to be not only labor-intensive but also brittle. A number of learning approaches were developed (Zelle and Mooney 1996; Miller et al. 1996) and more recently (Kate, Wong, and Mooney 2005; Kate and Mooney 2006; Zettlemoyer and Collins 2005; Wong and Mooney 2006; 2007), and (Lu et al. 2008). With two exceptions (Miller et al. 1996) and (Zettlemoyer and Collins 2005), they all adopted a semantic driven approach.

Academic question answering systems showed great promise: (Gunning et al. 2012) showed that domain experts with little training and no knowledge of the underlying knowledge base can use such systems to answer complex questions in scientific domains like Chemistry, Biology, and Physics.

Recently there has been an emerging interest from the industry sector to have computer systems not only to analyze the vast amount of relevant information (Ferrucci et al. 2010), but also to provide intuitive user interface to pose questions in natural language in an interactive dialogue manner (Sonntag 2009; Waltinger, Breuing, and Wachsmuth 2012). Several industrial applications of question answering have raised the interest and awareness of question answering as an effective way to interact with a system: *IBM Watson's Jeopardy* challenge (Ferrucci et al. 2010) showed that open domain QA can be done accurately and at scale. *Wolfram Alpha*'s[1] computational knowledge engine centered around *Mathematica* is one source behind *Apple's Siri*[2], which has proven a successful interaction medium for mobile devices.

[1]http://www.wolframalpha.com/
[2]http://www.apple.com/de/ios/siri/

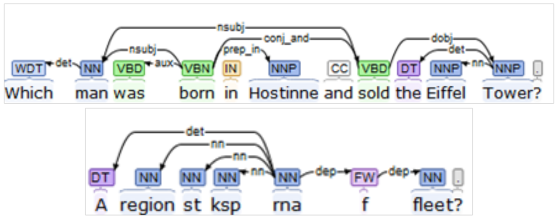

WDT det NN nsubj VBD aux VBN prep_in IN NNP CC VBD DT NNP nn NNP nsubj conj_and dobj det
Which man was born in Hostinne and sold the Eiffel Tower?
DT NN NN NN NN FW NN det nn nn nn dep dep
A region st ksp rna f fleet?

Figure 1: Parse tree representation as produced by *Stanford Core* contrasting open domain question vs. a domain-specific question as posed with the *USI Answers* setup. Note that *A region* is a custom expression for regions that have the letter *a* within their abbreviation.

Challenges of Building an Industrial QA System

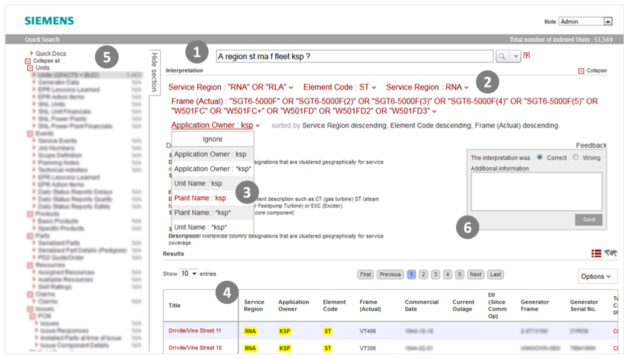
The challenges of building an industrial grade question answering system are many fold, due not only to the domain specificity of the underlying knowledge bases but also to the need to cover a wide range of queries that might come up during the interaction with the user.

The most pressing challenge is *run-time performance* on commodity hardware: in our current setup, an acceptable speed is defined as computing the answer representation within 800 ms. The system should be *scalable*, in that the response time should not be proportional to the size of data being accessed.

Enterprise data is heterogeneous and dynamic. A QA system needs to *integrate* these sources and accommodate their changing nature. Part of the integration process consists of offering unified semantics for the data.

Different business users need access to enterprise data, most of them know what they want but not exactly how to get it. An industrial QA system needs to allow them to *express queries easily*, as close to natural language as possible. This requirement is complicated by the fact that most business use *domain specific terms and concepts* to refer to their data. This terminology needs to be captured and used in the question answering process. Given how used we are with conversing in natural language, such a system has to offer *intuitive interfaces* for fixing errors (i.e. getting to the right meaning of a question) and visualizing the subsequent answer. That is, the system users demand to use not only (valid) natural language questions (e.g. `show me all active units in China`), query language constructs (e.g. `select unit name by performance sorted by capacity desc`), but also (traditional) keyword search (e.g. *a region st rna f fleet ksp*), or a mixture of these. This is important, since the regular syntax-driven approaches (e.g. identifying relationships by their parse tree (de Marneffe, MacCartney, and Manning 2006)) can hardly be used as a reference (see Figure 1 for an example).

Security is an important aspect of accessing data in an industrial setting: verification that the questioner has access to all pieces of data involved is required.



A region st rna f fleet ksp ?

Service Region : "RNA" OR "RLA" Element Code : ST Service Region : RNA
Frame (Actual) : "SGT6-5000F" OR "SGT6-5000F(2)" OR "SGT6-5000F(3)" OR "SGT6-5000F(4)" OR "SGT6-5000F(5)" OR "W501FC" OR "W501FC+" OR "W501FD" OR "W501FD2" OR "W501FD3"
Application Owner : ksp sorted by Service Region descending, Element Code descending, Frame (Actual) descending.

Application Owner : ksp
Application Owner : *ksp*
Unit Name : ksp
Plant Name : ksp
Plant Name : *ksp*
Unit Name : *ksp*

The interpretation was [✓ Correct] Wrong
Additional information

Figure 2: Overview of the *USI Answers* question answering interface. Each question (1) is represented via a list of *concept-instance-relationships* (i.e. Application Owner *hasValue* KSP) interpretations (2). Subsequently, the user is able to adjust the interpretation or to reformulate the input question (3). The results (4) are displayed as an list or as an direct answer. Different data sources (5) can be selected to apply the question answering process. Users are able to provide feedback (6) of the existing question interpretation.

USI Answers - Question Answering on Enterprise Data

USI Answers aims to give the right users with the right information at the right time using an automatic question answering system and to enable them to turn massive amounts of structured and unstructured service data into actionable knowledge using natural language queries. The data comes from unstructured sources (e.g. company news, products reports) as well as from various structured ones (e.g. Oracle data base views). The system allows users, even with a limited familiarity with technical systems and databases, to pose questions in a natural way and gain insights into the available data.

From the point of view of the user, the question answering process consists of several steps: *question interpretation*, during which a natural language question is transformed into one or more executable queries (called interpretations), *query adjustment*, during which the user is shown the available interpretations and he/she can select or modify one of these interpretations, and, finally, *query execution*, during which the selected interpretation is issued against the relevant data sources and an answer is returned (see Figure 2).

Similar to other approaches, our system needs to compute confidence in the interpretations generated. While questions may be ambiguous, our approach accumulates different confidence scores to compute an overall confidence in an interpretation. Different from other QA approaches, *USI Answers* system computes not only direct, but also list-based answers[3]. Besides producing a lexical representation of certain entities or phrases, the traditional QA output, it also constructs and validates structured queries (e.g. Lucene- and SQL-based) that can be executed against the list of indices/data bases. This is of high importance, since one of

[3]see Table 2 for the whole set of answer types

Segment	Lexical Level	Concept	Instance	Relation	Potential Reference
q1	Which units	-	-	hasLabel	Unit Name/...
q1	are operating	-	Operating	isValue	Unit Status/...
-	and	-	-	hasAnd	q1/q2
q2	have a provisional acceptance	PAC Date	-	hasValue	-
q2	after	-	-	isLaterAs / followsAfter	PAC Date/1968
q2	1968	-	1968	isValue	Number/Date Concepts

Table 1: Example trichotomy-oriented representation within the processing for question: *Which units are operating and have a provisional acceptance after 1968?*

the most common use cases of questions posed against the system are list-based answers.

Question	Type
What is a Mega Cluster ?	Definition
Active units in China ?	Domain List
Service region of New York ?	Geo-spatial Factoid
ST and KSP with RNA ?	Keyword
GT units with capacity $\leq$ 60MW ?	Numeric
PAC date by next fiscal year ?	Time/Date
SGT6 2000E NG60 ?	Abstract

Table 2: Example questions by type posed against the system.

Architecture

The question answering process in built as an *Apache UIMA*[4] pipeline. For each corresponding component depicted in Figure 4, a *UIMA* annotator has been developed and incorporated. The overarching semantic principle of the *USI Answers* system is the trichotomy of the representation of concept, instance, and the relation that connects them. That is, given an input question, the systems first tries to identify those information units that represent domain-, or database-specific concepts, and then the information entries that represents an associated value or instance of an concepts. Third, it tries to detect whether there is a relationship between the identified objects (concept-instance relationship). See Table 1 for an example trichotomy-oriented representation produced by the system.

This approach is needed since the data used in *USI Answers* consists primarily of (semi-) structured key-value associations stored within multiple Oracle database views. We refer to semi-structured properties, as the considered information units are not only single dates, numbers, temperatures or entities, but also entire sentences, phrases, or comment blocks. Due to this database-oriented nature of the target application, the expected answer type also differs to traditional (mostly factoid-based) QA systems. More precisely, since a requirement of the system was to offer access to a number of databases already in use, the developed QA system was developed as a *semantic layer* that connects and

[4]Unstructured Information Management Architecture http://uima.apache.org/

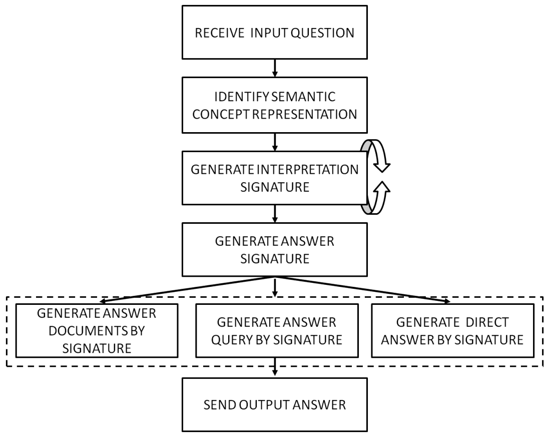

RECEIVE INPUT QUESTION
IDENTIFY SEMANTIC CONCEPT REPRESENTATION
GENERATE INTERPRETATION SIGNATURE
GENERATE ANSWER SIGNATURE
GENERATE ANSWER DOCUMENTS BY SIGNATURE
GENERATE ANSWER QUERY BY SIGNATURE
GENERATE DIRECT ANSWER BY SIGNATURE
SEND OUTPUT ANSWER

Figure 3: Overview of the system processing pipeline with regards to the respective interpretation and answer signatures. The interpretation signature can be adjusted based on user interaction.

manipulates existing query interfaces and the respective associated data bases. The respective answers are thereby primarily list-based that additionally involve joining of multiple database tables. The overall system pipeline of *USI Answers*, as depicted in Figure 3, shows the discrimination of the answer types that are send to the output controller.

The system pipeline works as follow: Each input question is processed by identifying its semantic concept representation. The semantic concept representation is defined as the typified representation of the input question. Subsequently, the respective interpretation signatures are generated (e.g 1968 $\mapsto$ date(1968); number(1968); ...). Thereupon, the answer signature is constructed. An answer signature consists of an answer type (e.g. direct answers or SQL-based), an answer property (e.g. numeric, date), and the expected database field where the answer may be found. That is, the most confident interpretations are selected and combined into individually ranked answer signatures. On the basis of the individual answer signature the system constructs either an answer document (e.g. report), an answer query (e.g. SQL), or produces the direct answer (e.g. factoid answer phrase), which eventually is send to the output component. Similar to other confidence-based approaches, each of the integrated component produces confidences that are used to score the individual interpretation. In the following section, we describe the QA component with regards to confidence scoring and signature generation in more detail.

Knowledge Representation

This section describes the knowledge that is used in the question interpretation and answering process, how it was acquired and how it is represented.

USI Ontology Information units and concepts related to the Oracle DB's are defined and represented in an ontology. That is, each of the domain-specific concepts have been defined, a short description was provided for each of them, along with their database identifier. A set of 1,520 most commonly used synonyms have been identified and captured in the ontology.

Primary Knowledge Bases We are using the information structure as defined in the Oracle DBs, though, converting each individual connected database view into a full text representation by means of its *Apache Lucene* index representation (Hatcher, Gospodnetic, and McCandless 2010). Note, that this data is both highly structured, in terms of clear key-value association given (e.g. *Country Name* $\mapsto$ *China*), but also consists of unstructured data (e.g. text extracted from PDF reports). Currently, USI Answers uses 38 different DB views, 36 different *Apache Lucene* indices, and 2 different *SPARQL* endpoints. We refer to these data sources as the *Primary Knowledge Bases*.

Secondary Knowledge Bases Secondary knowledge bases are used as a resource for gathering additional evidence for certain interpretation hypotheses, and for generating additional potential answers, which are not present within the primary knowledge bases. For example, Siemens Energy divides the world into service regions, geographical units that correspond roughly to continents. Mapping countries to service regions requires a list of all countries in the world. In the current release, the systems uses also multiple open domain-based resources, such as *DBpedia*[5], *FreeBase*[6], and *GeoNames*[7].

In addition, various domain-specific dictionaries have been compiled in order to capture about 12,000 regular expressions used to identify organization names as well domain specific objects (e.g. serial numbers). This information is represented in RDF and stored using Franz's Allegro Graph[8]. Domain specific knowlege was obtained through several interviews with a domain expert and formalized by knowledge engineers. We continue to enrich and refine this knowlegde.

Question Analysis

One of the first components that is applied within the question-answering process is question analysis. Within this analysis step, the systems normalizes the input stream, as passed through the input dispatcher, by applying:

[5]http://www.dbpedia.org/

[6]http://www.freebase.com/

[7]http://www.geonames.org/

[8]http://www.franz.com/agraph/allegrograph/

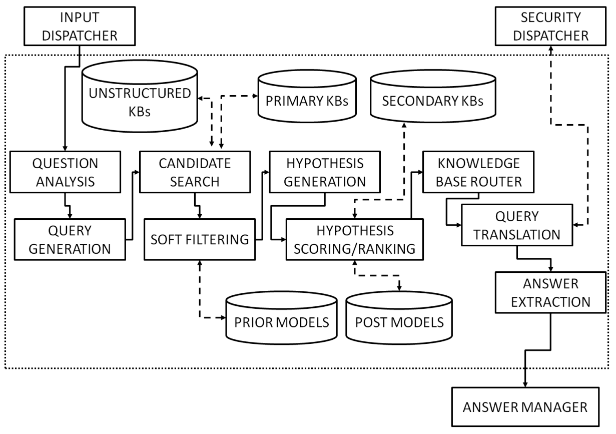


Figure 4: Overview of the *USI Answers* workflow with regards to the individual components.

- *Question Normalization:* Analyzing bracket-based grouping, resolving multiword units as indicated by quotes. Further normalization can be augmented through a configurable rule-set.
- *Metadata Annotation:* Adding meta data information such as user and session keys, prior selected data source restrictions, and security constraints needed downstream by the security dispatcher component.
- *Question Parsing:* As the most traditional step in question analysis, shallow parsing is applied. It includes lemmatization, PoS-tagging, named entity recognition and disambiguation, syntactic chunk and the dependency parse tree using the *UIMA*-based *ClearTK* annotator (Ogren, Wetzler, and Bethard 2008) in conjunction with *Stanford Core NLP*[9].
- *Analysis Validation:* This step is needed, so the system can handle domain-specific input terms such as product numbers and serial codes (e.g. 223/2 a39), which may have been erroneously split, tagged, parsed or concatenated within the shallow parsing phase. The system applies a list of syntax rules that re-validate the entity information. There are currently 8 rules, all domain specific (e.g. an *x* between two numbers is a wildcard for any expression between them (e.g. *25x3a1* $\mapsto$ *25A3a1;25B3a1;25C3a1;*).
- *Question Classification:* Focuses on the identification of the answer signature. That is, analyzing the question type (e.g. factoid or list-based), the representation mode (e.g. direct answer, sql-statement), and the question focus (e.g. referenced entity object). The latter is identified by applying the rule-based approach as proposed by (Schlaefer, Gieselman, and Sautter 2006). We use 12 syntactic rules (e.g. a WP-VBZ-[NE/PER] sequence refers to a PERSON).

For each of the corresponding modules an *UIMA* annotator has been developed and incorporated, as the *Question Anal-*

[9]http://nlp.stanford.edu/software/corenlp.shtml

ysis component in the overall QA processing pipeline (see Figure 4).

Query Generation

The next task in the processing pipeline is query generation. As a result of the question analysis, we are able to directly access the individual (parsed) question tokens and objects respectively. The query generation component produces a search query with reference to the specific knowledge base query syntax for each accounted input object. Currently, this component supports *Dictionary*- and *Regular Expression*-based look-ups, but also *Apache Lucene*-, *SparQl*-, and *SQL*-based query syntax formats.

Candidate Search

Within the candidate search component, the system aims to identify and resolve the different concepts that may be interlinked. In general, following the trichotomy-oriented representation as described and depicted in Table 1, the system tries to search for and distinguish between concepts (called *answerFields* - e.g. *PAC Date*), concept values instances (called *searchFields* - e.g. *operating*) or already augmented key-value pairs (called *domainFields* - e.g. *Country Name : China*). In addition, this component annotates the relationship properties between key-value pairs and identifies *time-and-date-references* within the query. That is, each time reference, such as the expression *today* will be annotated by its time value in terms of *Oracle* time stamp. The *query-expansion* module queries the *SPARQL* endpoint trying to collect different surface forms of a single entity (e.g. *GE vs. General Electric*). The *open domain knowledge* module collects data as gathered within the *DBpedia* dataset (Auer et al. 2008). For each of the corresponding components an *UIMA* annotator has been incorporated in the overall QA processing pipeline.

Soft Filtering

Soft Filtering is applied to detect and (pre-)validate the different relations and objects assigned within by candidate search component. Based on pre-learned *prior models*, we remove first relationships (e.g. *searchFields* annotations) and initially rank the different annotations referenced to the respective query question tokens (e.g. *MW* has an higher probability to refer to *mega watt* than to *milli watt*).

This component is of importance as the different connected knowledge bases may assign a number of different annotations to single and multiple terms of the input question.

Hypotheses Generation

This component generates different question interpretations (i.e. hypotheses of what the question might mean). More precisely, on the basis of the *candidate search* component, it generates different hypothesis of how the *answerFields* (concept) and *searchFields* (instance) are connected to each other (e.g. direct or implicit) (relation):

```
con(Serial No.); rel(larger than); ins(1968);
hypothesis1(con,rel,ins);
confidence1(hypothesis1,0.76);
```

Since each concept and value have multiple representations assigned, which may be connected over multiple relationships (e.g. textual representation of date or numbers), the list of different hypothesis can get very complex. In addition, this component gathers also hypotheses (on e.g. *geo-reasoning*) that need to be applied if the focus of the question is targeting a location. An *RDF-based open topic grammar* gathers hypothesis on definitions that may be needed to answer the question. For example, given a pattern such as WP-VBZ-NE, a *DBpedia* query is constructed by focusing on the entity type *has abstract* (e.g. *http://dbpedia.org/ontology/abstract*) for definition answers.

ST	Rt	Conf.	**1968**	Rt	Conf.
Elem. Co.	isValue	0.95	Serial No.	partOf	0.23
Unit Type	hasValue	0.67	PAC	timeRef	0.88
...	...	...	...	...	...

Table 3: Example ranking of concept hypothesis relations for the lexical entries *ST* and *1968*. Each lexical entry is interlinked to a list of possible interpretations (e.g. Elem. Co.,...), relations types (Rt), and confidence scores (Conf.)

Hypothesis Scoring and Ranking

The next component in the question answering pipeline uses the different hypotheses generated from the latter module to assign confidence scores, which indicates the probability that a given surface (terms/phrase) is represented through a given concept (see Table 3). For example, the phrase: " *country is not mx* " is mapped to *country-name* $! =$ *mexico* $||$ *country-code* $! =$ *mx* .

For each hypothesis, the system tries to collect *evidence support* to have a hypothesis validated. In the latter example, it connects to the list of *secondary knowledge bases* to resolve mx as an possible *country code* that may be a represented through the label *mexico*, though *country* needs to be converted either into *country code* or *country name*.

In addition, the system utilizes learned models (referred to as *post models*) to re-validate and re-rank certain key-value associations. These models have been trained by using user-defined database views and their associated labels. For example, the users have defined views on the utilized data base in the form of a simple query syntax: *company : siemens AND primary-fuel : coal AND turbine-status: st*. On the basis of these data queries, we have trained the model to perform not only a confidence-based disambiguation of the gathered hypothesis, but also to iteratively and automatically capture domain knowledge as authored by the Energy experts. We have used 1,770 user-defined views for training.[10] In the last step, after all confidence scores are assigned to the different hypotheses, a final interpretation object is build to be passed to the *knowledge base router* module.

[10]Note that for the experiments we have applied a 5-fold-cross-validation, not the entire gold-standard set.

Knowledge Base Router

As our primary knowledge sources consist of multiple data base views and *Lucene* indices, this component is needed to detect and select the appropriate data sources for joining and querying. That is, based on the ranked confidence scores of hypotheses it collects data sources that are involved and ranks them by their distribution (in terms of number of hypotheses used) and their associated confidences. In terms of SQL syntax, it detects the order of database joins of different tables to be applied. In addition, it detects whether we need to combine structured and/or unstructured data sources to answer the question.

Query Translation

The *query translation* component uses the information gathered by the latter two modules, *hypothesis ranking* and *knowledge base router*, to construct the final query in the representative query language. Within the current setup, the system incorporates four different translation modes. It automatically constructs *SQL*, *Apache Lucene*, *SparQL* and *Solution Object* queries. The latter refers to domain-specific object representation used with the target application. In addition, this component defines also the *DB*, *RDF* or *Lucene* columns where the potential answer value is found. Finally, it defines the so-called *representation mode* and its property that needs to be propagated to the user. For example, it indicates whether the answer is already generated, in terms of a direct factoid, or it refers to a list-based answer and therefore the generated query needs to be executed to gather the answer list.

Answer Extraction

This component focuses on the actual answer projection for a given input question. That is, it applies either the factoid filter, with regards to definitional questions, or applies the answer post-processor by using post-calculation and computation. Finally, based on the resultant answer mode, if present, the direct answer or the answer query is passed to the output dispatcher.

Answer Manager

The *answer manager* coordinates the *backend* and *frontend* communication within the question answering. It executes the query by means of the respective representation mode (e.g. *SQL*, *SparQL* or direct answer) and communicates the results via an interface to the *frontend*. In addition, this component provides a number of web services with regards to geographic and statistical visualization of the answers.

Security Dispatcher

The question answering pipeline connects to the *security dispatcher* to validate both that the user has access to the required data sources as well as to the specific data items that might be part of an answer. The security model divides users into certain groups, and assigns row based-access per such group. A user can belong to one or more such groups.

Development, Deployment, and Use

The system was developed by a team of about ten to twelve engineers and scientists from Siemens Corporation, Corporate Technology, located in US, Germany, and Romania, over the course of 3 years[11]. *USI Answers* has been in use by Siemens Energy Service since May 2012 and is being accessed regularly by more than a thousand users in 18 countries. It is still under active development and it receives regular updates, several times per year.

User experience has been very positive, the general feedback being that semantic search integration simplified user experience, particularly for new users. It lowered medium complexity cases by about 75 percent. For example, the search for *North American open market units missing next outage* went from 90 seconds to 15 seconds (time includes definition and execution). The number of steps required for the user to perform more complex searchs dropped by 90 percent. The number of failure points (i.e. points where users typically get confused about what to do) dropped by more than 95 percent. Consequently, Siemens Energy was able to lower the initial new user training course times by more than 50 percent while more than doubling the rate of user retention more than doubled. (Usage of the system is optional).

User comments also shifted from comments such as "system is impressive but too overwhelming" to "system thinks like me. It is very intuitive".

Experiments

Due to the absence of an existing evaluation corpus in the context of automatic *SQL* conversion, we constructed such a corpus by ourselves, with a special emphasis on *abstract* question/query types (see Table 2). The goals of the experiments conducted were threefold. First, we were interested in the onverall performance of *hypothesis generation and ranking* by observing directly the *concept-instance* pairs generated by the system. Is the system able to rank and assign the right concept relation by just looking at a single value, without any references given? Second, the entire query was analyzed by removing again all concept references, and trying to have the system disambiguate and resolve the concept references, and to construct the gold standard representation in *SQL* syntax. Third, we were interested in analyzing the performance of the individual components of the QA pipeline, in particular we focused on the influence of the *prior-* and *post model* learning components within the *USI Answers*. For all experiments, we applied a *5-fold-cross-validation* technique was used, and relevance was measured as *recall* and *precision at rank k (P@k)*.

Dataset

The reference corpus (i.e. gold standard) used in the evaluation was compiled from $1,770$ database *SQL* query views, which have been converted into an abstract *key-value*-based representation (see example template below). These views were collected from 395 users that could define such

[11]This is the total time that it took to develop the whole *USI* project which *USI Answers* is part of

database views by means of aggregating different *SQL* commands through a query builder interface and provide a name for each of them.

Example entry of a named view used for evaluation

```
Template: #VIEW DESCRIPTOR (
     KEY:VALUE AND/OR/NOT
     KEY:VALUE ...
   )
...
Example: Generator 156(
  GENERATOR_FRAME_T:*115/36*  AND
  GENERATOR_SERIAL_NUMBER_T:12*  AND
  FRAME_FAMILY_GR_DISPLAY_T:(SGT5-4000F) AND
  ...
   )
```

We have used only those views that had a minimum number of two *key-value* pairs. The resultant reference corpus consisted of $1,142$ named views, with $8,733$ key-value pairs. Note that, on average, each SQL view comprises 7.6 *key-value* pairs. For the experiments, all *key* references (e.g. *generator frame,...*) have been removed. That is only the respective values (e.g. *115/36*) have been used as an input to the QA system.

Example input question representation, (from above example), used as an input within the evaluation setup.

```
Input: *115/36* 12* (SGT5-4000F)
```

For each input representation, the system was evaluated by measuring the *hypothesis ranking* of the *concept-instance* pairs, as well as with regards to the prediction of the full initial input query. As an *answer signature*, the automatic SQL translation mode has been used.

Generated SQL *result entry example, as constructed within the evaluation setup.*

```
Output: select GENERTOR_FRAME from T1 where
        GENERATOR_FRAME like '%115/36%' and
        GENERATOR_SERIAL_NUMBER like '12%' and
        FRAME_FAMILY_GR_DISPLAY = 'SGT5-4000F'
```

Results

The results of the experiments are depicted in Table 4 and Table 5. In order to evaluate the contribution of each component in the QA pipeline, we constructed fours versions of the system by ablating different modules: *All* refers to the use of the entire QA pipeline; *No Post* refers to the pipeline without the usage of the post models; *No Prior* refers to the version without the usage of the prior models, and *No Init* to that without the initialized confidence scores as produced by the candidate search component.

The analysis of the *hypothesis generation* and *ranking* components of the system, done by means of observing concept instance pairs, the results, as depicted in Table 4, show with a *recall* of 0.948 and a *precision at rank 1* of 0.765, a good performance with regards to accuracy, and a very good performance with regards to recall. The experiments on the entire *view name* prediction, as depicted at Table 5, highlight again the effectiveness of the system, even though, handling only a partial re-constructed representation. More precisely, as Table 5 shows, applying the QA pipeline on the entire query, with on average 7.6 *key-value* pairs, the system is able to re-construct *concept-instance* references with a precision of 0.765 (see Table 4), but is still able to rank, with a p@1 of 0.908, the correct full *SQL* query *view* at rank one.

Rank P@k	**All**	**No Post**	**No Prior**	**No Init**
Recall@1	**0.948**	0.947	0.948	0.948
Rank P@1	**0.765**	0.517	0.365	0.290
Rank P@5	0.779	0.533	0.387	0.314
Rank P@10	0.779	0.533	0.387	0.315

Table 4: Results of the evaluation experiments using $8,733$ key-value pairs from $1,142$ named views. Each *key* has been deleted and tried to reconstruct by *USI Answers* application.

Rank P@k	**All**	**No Post**	**No Prior**	**No Init**
Recall@1	**0.921**	0.921	0.920	0.858
Rank P@1	**0.908**	0.901	0.898	0.742
Rank P@5	0.957	0.949	0.955	0.895
Rank P@10	0.974	0.969	0.973	0.935

Table 5: Results of the evaluation experiments using $1,142$ named views. For each *view* the entire list of *keys* has been removed. The QA system used the partially reconstructed *key-value* representation, to predict the representation of the initial golden-standard view name.

Analyzing the individual components of the QA pipeline, a clear influence of the *learning* modules can be identified. Note that the *post* model focus on reassessing the predicted *concept-instance-relationship* triple by its confidence scores, while the *prior* models emphasize the lexical representation of the input question only. The *initial* confidence scores as produced by the *candidate search* component focuses also on the lexical representation by means of domain dictionaries and lexicons. Not surprisingly, without the *learning* components the precision drops within both experimental setup significantly. The initial precision of 0.742 (see Table 5) of the system without the learning components can be traced to the actual setup of the experiments. More precisely, as a baseline the system compares the value-only-based representation with the fully typified (gold standard) representation, which reaches, in the *5-fold-cross-validation* scenario, a precision of more than 0.74 and a recall of 0.85, by means of computing the cosine similarity between both lexical representations. In the context of negative examples, the system often failed on questions that consisted of a combination of *concepts* and *instances* (e.g. *SGT6-5000F - W501F* vs. *serial number - generator number*), which have eventually the same number but are differently defined by the user within the named view. Other failures of the system could be traced to the limited coverage of the encoded domain knowledge and to incomplete tagging and pre-processing errors.

Conclusion and Outlook

We deployed *USI Answers* as a natural language question answering system for semi-structured *Siemens Energy* data. The system offers easy access to enterprise data to thousands of business users in 18 countries. The system allows users, even with a limited familiarity with technical systems and databases, to pose questions in a natural way and gain insights of the underlying data sources available. It makes aparent the system's interpretation of the query, and allows easy query adjustment and reformulation. We evaluated our approach on a dataset consisting of *SQL*-based fleet data by focusing on a threefold analysis, comprising *hypothesis generation*, *ranking* and *component* performance. While the current evaluation emphasized *abstract* question types, we aim to extend it the future, in the context of geo-reasoning and domain-specific factoid question types. Currently the *UIMA*-based QA pipeline is tailored to the *Energy* domain. However, in future work, we are interested in applying it to other usecases and data coming from domains such as healthcare and industry.

References

Auer, S.; Bizer, C.; Kobilarov, G.; Lehmann, J.; Cyganiak, R.; and Ives, Z. 2008. DBpedia: A Nucleus for a Web of Open Data. In *Proceedings of the 6th International Semantic Web Conference (ISWC)*, volume 4825 of *Lecture Notes in Computer Science*, 722–735. Springer.

Bos, J.; Mori, Y.; Gambäck, B.; Pinkal, M.; Lieske, C.; and Worm, K. 1996. Compositional semantics in verbmobil. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, 131–136. Association for Computational Linguistics.

de Marneffe, M.-C.; MacCartney, B.; and Manning, C. D. 2006. Generating typed dependency parses from phrase structure trees. In *LREC*.

Dowding, J.; Gawron, J.; Appelt, D.; Bear, J.; Cherny, L.; Moore, R.; and Moran, D. 1993. Gemini: A natural language system for spoken-language understanding. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, 54–61. Association for Computational Linguistics.

Ferrucci, D.; Brown, E.; Chu-Carroll, J.; Fan, J.; Gondek, D.; Kalyanpur, A. A.; Lally, A.; Murdock, J. W.; Nyberg, E.; Prager, J.; Schlaefer, N.; and Welty, C. 2010. Building Watson: An overview of the DeepQA project. *AI Magazine* 31(3).

Gunning, D.; Chaudhri, V.; Clark, P.; Barker, K.; Chaw, S.; Greaves, M.; Grosof, B.; Leung, A.; McDonald, D.; Mishra, S.; Pacheco, J.; Porter, B.; Spaulding, A.; Tecuci, D.; and Tien., J. 2012. Project halo update-progress toward digital aristotle. *AI Magazine* 31(3).

Hatcher, E.; Gospodnetic, O.; and McCandless, M. 2010. *Lucene in Action*. Manning, 2nd revised edition. edition.

Kate, R. J., and Mooney, R. J. 2006. Using string-kernels for learning semantic parsers. In *ACL 2006: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, 913–920. Morristown, NJ, USA: Association for Computational Linguistics.

Kate, R. J.; Wong, Y. W.; and Mooney, R. J. 2005. Learning to transform natural to formal languages. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, 1062–1068.

Lu, W.; Ng, H.; Lee, W.; and Zettlemoyer, L. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the EMNLP*, 783–792. Association for Computational Linguistics.

Miller, S.; Stallard, D.; Bobrow, R.; and Schwartz, R. 1996. A fully statistical approach to natural language interfaces. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, 55–61. Association for Computational Linguistics.

Ogren, P. V.; Wetzler, P. G.; and Bethard, S. 2008. ClearTK: A UIMA toolkit for statistical natural language processing. In *Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP workshop at Language Resources and Evaluation Conference (LREC)*.

Schlaefer, N.; Gieselman, P.; and Sautter, G. 2006. The ephyra qa system at trec 2006. In *Proceedings of the Fifteenth Text REtrieval Conference, TREC 2006, Gaithersburg, Maryland, November 14-17, 2006*, volume Special Publication 500-272. National Institute of Standards and Technology (NIST).

Sonntag, D. 2009. Introspection and adaptable model integration for dialogue-based question answering. In *Proceedings of the 21st international jont conference on Artifical intelligence*, 1549–1554. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Waltinger, U.; Breuing, A.; and Wachsmuth, I. 2012. Connecting question answering and conversational agents - contextualizing german questions for interactive question answering systems. *KI* 26(4):381–390.

Warren, D., and Pereira, F. 1982. An efficient easily adaptable system for interpreting natural language queries. *Computational Linguistics* 8(3-4):110–122.

Wong, Y. W., and Mooney, R. J. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of HLT-NAACL-06*, 439–446.

Wong, Y. W., and Mooney, R. J. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *Proceedings of NAACL-HLT-07*, 172–179.

Woods, W. 1970. Transition network grammars for natural language analysis. *Communications of the ACM* 13(10):591–606.

Zelle, J. M., and Mooney, R. J. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, 1050–1055. Portland, OR: AAAI Press/MIT Press.

Zettlemoyer, L., and Collins, M. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *Proceedings of 21st Conference on Uncertainty in Artificial Intelligence (UAI)*.