Natural Language Access to Enterprise Data

Ulli Waltinger, Dan Tecuci, Mihaela Olteanu, Vlad Mocanu, Sean Sullivan

■ *This article describes USI Answers — a natural language question-answering system for enterprise data. We report on the progress toward 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.*

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 sheer volume: enterprise data is predicted to grow by 800 percent in the next five years (Chuba 2012). The biggest part (80 percent) is stored in documents, most of them missing informative metadata 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 its underlying syntax or semantics.

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 article, is a subproject 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 of the question-answering system (QA) 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 a natural language sentence into a complete, unambiguous, formal meaning representation expressed in a formal language that supports other tasks such as automated reasoning, or question answering.

Natural language interfaces for databases (NLIDB) is an NLU task where the target language is a structured query language (for example, 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 in papers by Zelle and Mooney (1996) and Miller et al. (1996) and more recently in papers by Kate, Wong, and Mooney (2005), Kate and Mooney (2006), Zettlemoyer and Collins (2005), Wong and Mooney (2006), Wong and Mooney (2007) and Lu et al., (2008). With the exception of the papers by Miller et al. (1996) and Zettlemoyer and Collins (2005), they all adopted a semantics-driven approach.

Academic question-answering systems displayed great promise. The paper by 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 a renewed interest from industry in having computer systems not only analyze the vast amounts of information (Ferrucci et al. 2010), but also in providing intuitive user interfaces to pose questions in natural language in an interactive dialogue manner (Sonntag 2009; Waltinger 2011; Waltinger, Breuing, and Wachsmuth 2011; 2012). Several industrial applications of question answering have raised the interestand awareness of this technology 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.

The Challenges of Building an Industrial QA System

The challenges of building an industrial-grade question-answering system are manifold, due to the domain specificity of the underlying knowledge bases as well as 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 milliseconds. 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 such sources and accommodate their changing nature. Part of the integration process consists of offering a 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 all of them to express queries easily, as close to natural language as possible. This requirement is complicated by the fact that most businesses 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 to conversing in natural language, such a system has to offer intuitive interfaces for fixing errors (that is, 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 (for example, *show me all active units in China*), query language constructs (for example, *select unit name by performance sorted by capacity desc*), but also (traditional) key-word search (for example, *a region st rna f fleet ksp*), or a mixture of these. This is important because the regular syntax-driven approaches (for example, 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.

USI Answers — Question Answering on Enterprise Data

USI Answers aims to provide the right users with the right information at the right time through automatic question answering and to enable them to turn massive amounts of structured and unstructured service data into actionable knowledge using natural language queries.

More precisely, USI Answers is used by different user groups and communities, from service engineers

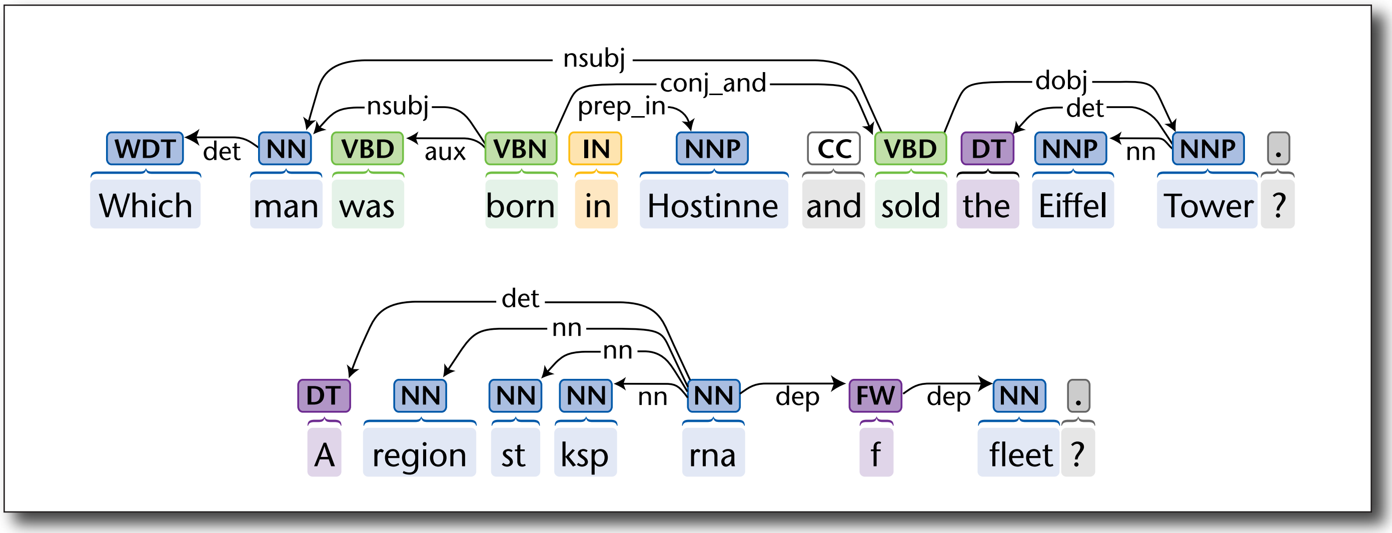


Figure 1. Parse Tree Representation.

The representation is as produced by Stanford Core contrasting open domain question with a domain-specific question typical to the USI Answers. Note that *A* region is a domain-specific term for regions that have the letter *a* within their abbreviation.

to sales people within the Energy sector of Siemens. Users asks questions relevant to their roles, such as: When are the next service events for my units?, How well are they performing? (engineering perspective), or What issues have they had since I last visited? and How many nonfossil units are currently operating in China? (sales perspective). The data comes from unstructured sources (for example, company news, products reports) as well as from various structured ones (for example, Oracle database views).

The system enables 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. The system needs to be able to successfully answer a broad set of domain-specific questions under uncertainty conditions imposed by the inherent ambiguity in language. The overall guiding idea behind the USI Answers system is simple: Train the system on how people query for information instead of training people on how to query the system.

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).

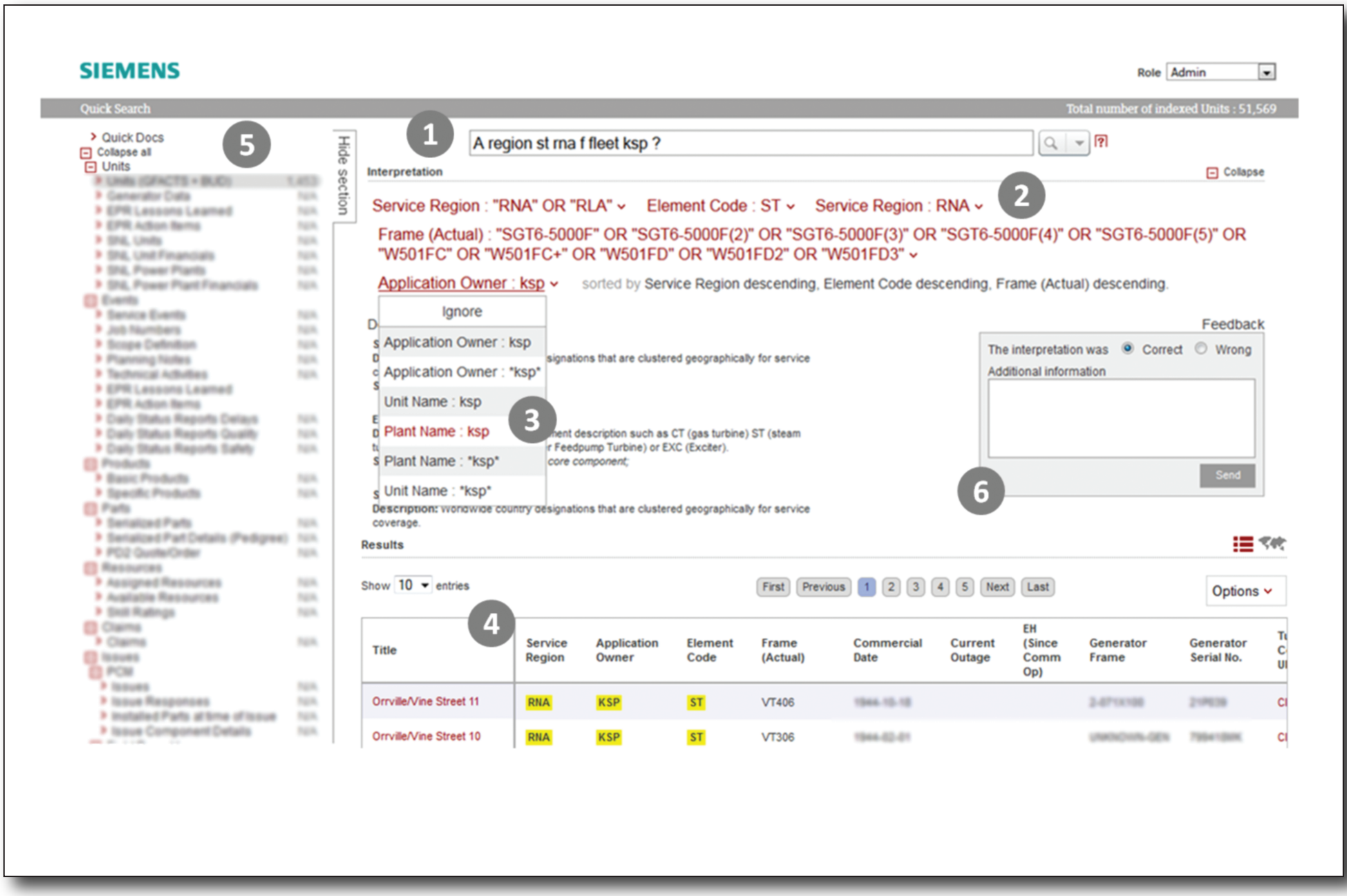
Similar to other approaches, our system computes confidence in each generated interpretation. The overall confidence in an interpretation is accumulated from different confidence scores. In USI Answers we distinguish between direct answers, for which a single piece of information is an appropriate answer (for example, for a question like what is a megacluster?) and list-based answers that need to provide a set of answers (for example, for questions like show me all units active in china?).[3] The latter type is the most common question type posed against the system.

Besides producing a lexical representation of the question (that is, the traditional QA output), our system also constructs and validates structured queries (for example, Lucene-, SQL-, and SPARQL-based) that can be executed against the list of indices or databases. An important feature is that the system can retrieve both structured and unstructured information, consisting of information units that may not have been extracted or defined in advance (for example, product and serial numbers).

Currently, the system processes over 1.5 million different data sheets and information items. It integrates more than 80 different sources, many of which have their own syntax and semantics. Even at current volume and especially as data expands, users cannot scroll through or be aware of the hundreds of database fields and reports that may be of relevance to their information need. The system should understand their information need and retrieve the relevant data.

Architecture

The question-answering process is built as an Apache UIMA[4] pipeline. UIMA is a software component architecture for analysis of unstructured information and integration in search. A typical UIMA application consists of a set of annotators connected in a pipeline. The USI Architecture will be depicted later in figure 6 where each component is implemented by such an



A region st rna f fleet ksp ?

Interpretation

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.

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

Description: Worldwide country designations that are clustered geographically for service coverage.

Feedback
The interpretation was [✓ Correct] Wrong
Additional information

Results

Show 10 entries First Previous 1 2 3 4 5 Next Last

Title	Service Region	Application Owner	Element Code	Frame (Actual)	Commercial Date	Current Outage	EH (Since Comm Op)	Generator Frame	Generator Serial No.
Orrville/Vine Street 11	RNA	KSP	ST	VT406	[redacted]			[redacted]	[redacted]
Orrville/Vine Street 10	RNA	KSP	ST	VT306	[redacted]			[redacted]	[redacted]

Figure 2. Overview of the USI Answers Question-Answering Interface.

Each question (1) is transformed into a set of representations (2) of the form concept-instance-relationships like *ServiceRegion hasValue RNA or RLA*. Subsequently, the user is able to adjust the interpretation or to reformulate the input question (3). The results (4) are displayed as a list or as a direct (that is, single) answer. Different data sources (5) can be selected to apply the question-answering process. Users are able to provide feedback (6) about individual question interpretations.

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. Trichotomy-Oriented Representation Produced by the System.

This is an example trichotomy-oriented representation within the processing for the question: Which units are operating and have a provisional acceptance after 1968?

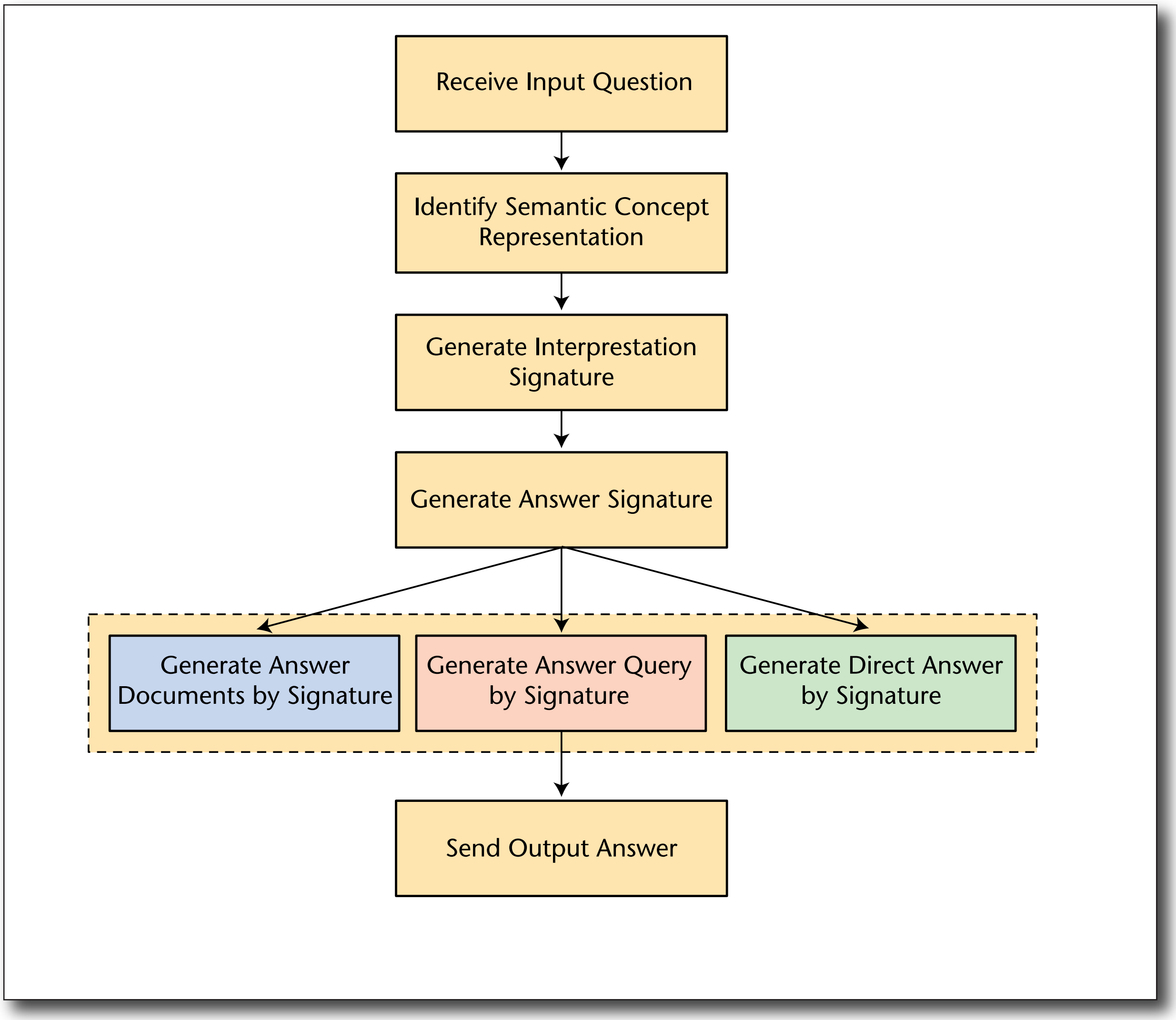

Receive Input Question
Identify Semantic Concept Representation
Generate Interprestation Signature
Generate Answer Signature
Generate Answer Documents by Signature
Generate Answer Query by Signature
Generate Direct Answer by Signature
Send Output Answer

Figure 3. Overview of the System Processing Pipeline with Regard to the Respective Interpretation and Answer Signatures.

The interpretation signature can be adjusted based on user interaction.

annotator. 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 system first tries to identify those information units that represent domain- or database-specific concepts, and then the information entries that represent an associated value or instance of a concept. 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.

The trichotomy representation 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 semistructured properties, as the considered information units are not only single dates, numbers, temperatures, or entities, but also entire sentences, phrases, or comment blocks. We process and link also unstructured information to domain entities. For example, we capture the relation between a specific (unstructured) report and a specific unit that is mentioned in it and store this association in the database. Due to this database-oriented nature of the target application, the expected answer type also differs from traditional (mostly factoid-based) QA systems. More precisely, 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 manipulates existing query interfaces and the respective associated databases. The respective answers are thereby primarily list-based answers that additionally involve joining of multiple database tables. The overall system pipeline of USI Answers, as depicted in figure 3, shows the dis-

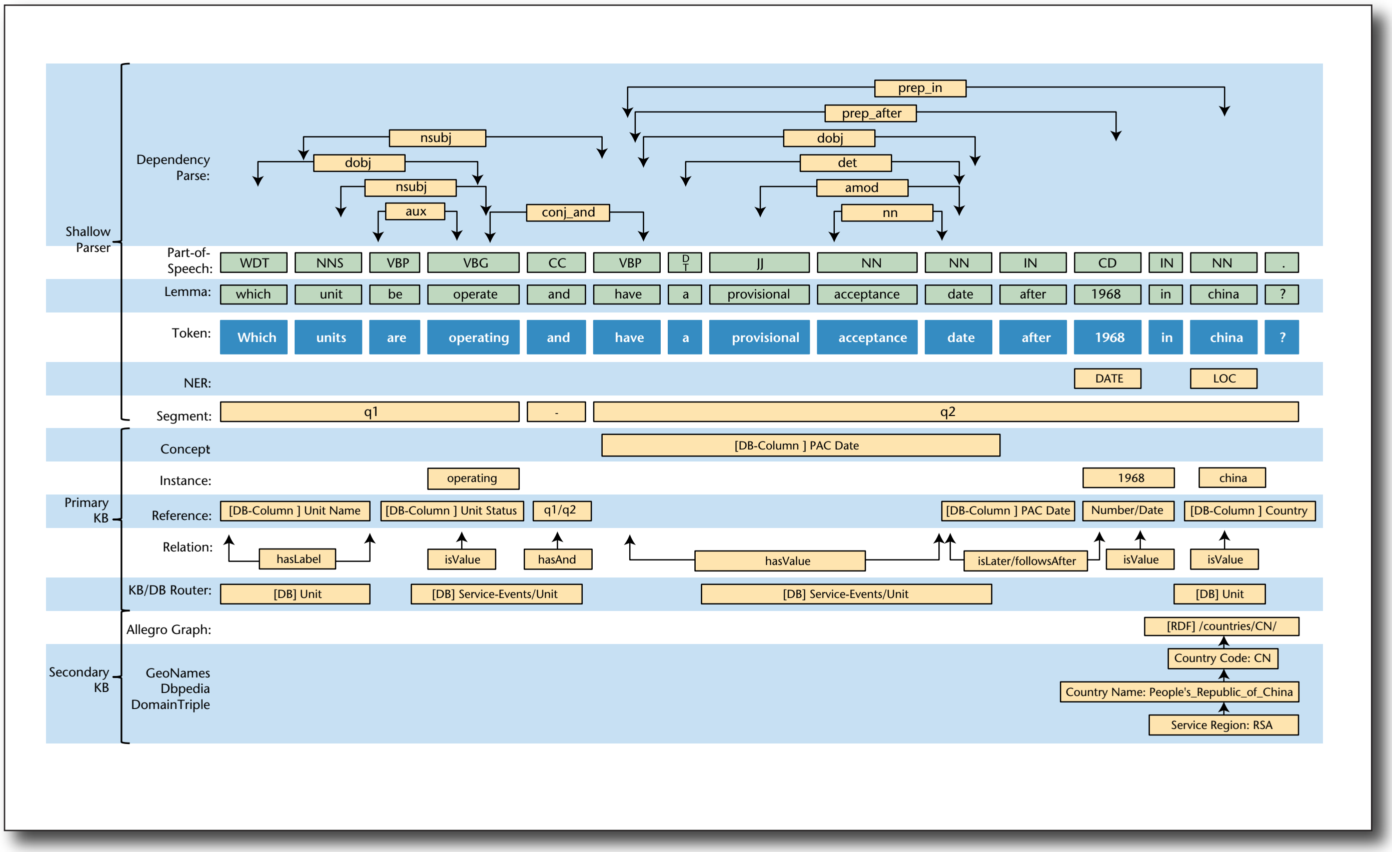


Figure 4. Building the Semantic Concept Representation for a Given Question.

This figure provides an overview of building a semantic concept representation of an input question using UIMA-based annotators for the example question: Which units are operating and have a provisional acceptance date after 1968 in China? The shallow parsing component augments the tokenization, lemmatization, part-of-speech tagging, dependency parse structure, and NER. The annotation components of the PrimaryKB enables resolution of database-based column field references and their associated relations to the different databases. The Knowledge Representation section will describe this process in more detail.

crimination of the answer types that are sent to the output controller.

Let us consider the example question, which units are operating and have a provisional acceptance date after 1968 in China?. This is a typical question for this domain, combining obvious domain terms like *provisional acceptance* and less obvious ones like *operating*, as well as generic notions of time and location. Figure 3 describes the overall system functioning: each input question is processed by building its semantic concept representation, defined as the typified representation of the input question. Subsequently, the respective interpretation signatures are generated (for example, 1968 $\mapsto$ date[1968]; number[1968]). There can be multiple such interpretations for a given input question. All interpretations are ranked by confidence values (for example, 1968 is more likely a year than a number), and finally filtered by a predefined threshold. The most confident interpretations are selected and combined into individually ranked answer signatures. An answer signature consists of an answer type (for example, direct answers or SQL based), an answer property (for example, numeric, date), and the expected database field where the answer may be found.

On the basis of the individual answer signatures the system constructs either an answer document (for example, report), an answer query (for example, SQL), or produces the direct answer (for example, factoid answer phrase), which eventually is sent to the output component. Similar to other confidence-based approaches, each of the integrated components produces confidences that are used to score the individual interpretation.

We will refer to figure 4 in order to describe in detail how the semantic concept representation is built for a given question. The first step in the semantic analysis is the shallow parsing component that does tokenization, lemmatization, part of speech tagging, generation of the dependency parse structure, and named entity recognition (NER). Following that, annotation components are applied that make use of the primary knowledge bases (PrimaryKB) that resolve database-based column references and their relations to the different databases. Following that, annotations based on secondary knowledge bases

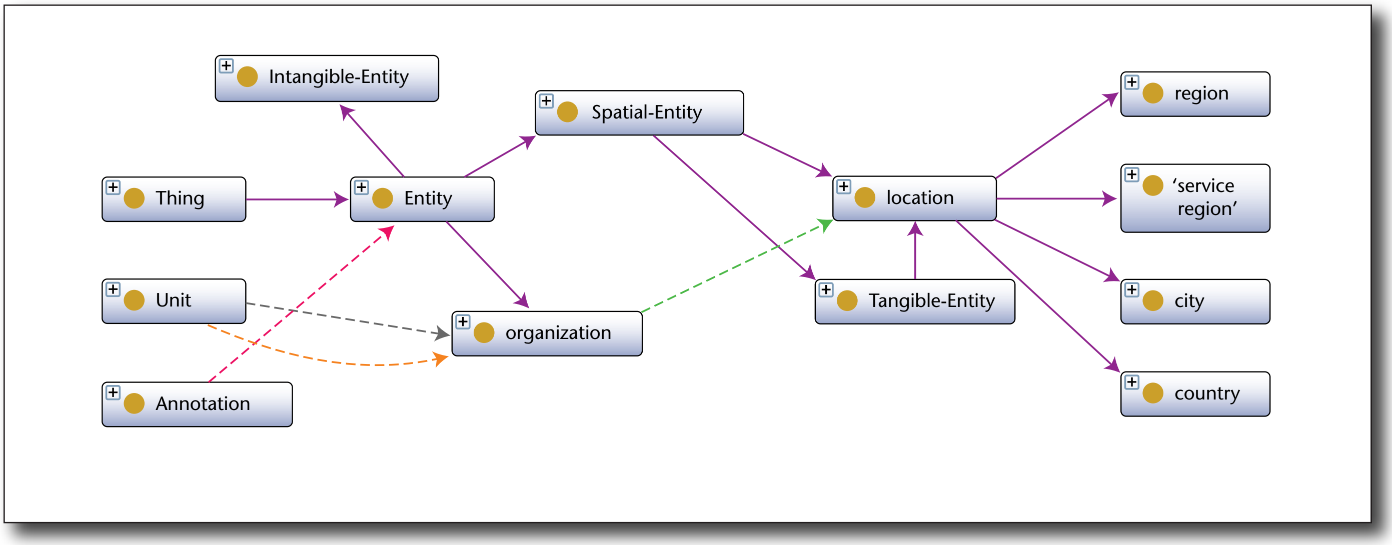


Figure 5. Excerpt of the Domain Model.

An excerpt of the USI ontology capturing the relation between locations and service regions within the concept of entities.

(SecondaryKB) allow the interlinking to standard (that is, URI-based) entities within the various open and domain-specific resources.

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 DBs are defined and represented in an ontology. The knowledge elicitation was conducted through face-to-face and email interviews with domain experts. In this knowledge-acquisition process domain-specific concepts and their relations to existing reports and database views have been captured and validated. Eventually, each of the domain-specific concepts has been defined and a short description provided for each of them, along with their database or report identifier. A set of 1520 most commonly used synonyms have been identified and additionally captured in the ontology. The knowledge model is represented using a Web Ontology Language (OWL) representation — a family of knowledge representation languages for authoring ontologies that allows data to be shared and reused across application, enterprise, and community boundaries. During the ontology construction process, Protege[5] — an OWL editor — was used (see figure 5 for an excerpt of the domain model).

Primary Knowledge Bases

Currently, the system processes more than 1.5 million different data sheets and information items. As a primary knowledge base, we are using the information structure as defined in legacy Oracle DBs, though, converting each individual connected database view into a full text representation by means of its Apache Lucene[6] index representation (Hatcher, Gospodnetic, and McCandless 2010). Note that this data is both highly structured, in terms of clear key-value association given (that is, Country Name 1 ↦ China), but also consists of unstructured data (that is, Unit X 1 ↦ hasReport 1 ↦ text extracted from PDF report Y). More precisely, we represent each primary data source and each database view by means of their concepts (that is, DB columns such as PAC Date in figure 4) and the respective list of instances individually. Currently, USI Answers uses 38 different DB views, 36 different Apache Lucene indices, and 3 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 resources, such as DBpedia,[7] FreeBase,[8] and GeoNames.[9]

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 as domain-specific objects (for example, serial numbers). This information is represented in RDF[10] and stored using Franz's Allegro Graph.[11]

Domain-specific knowledge was obtained through

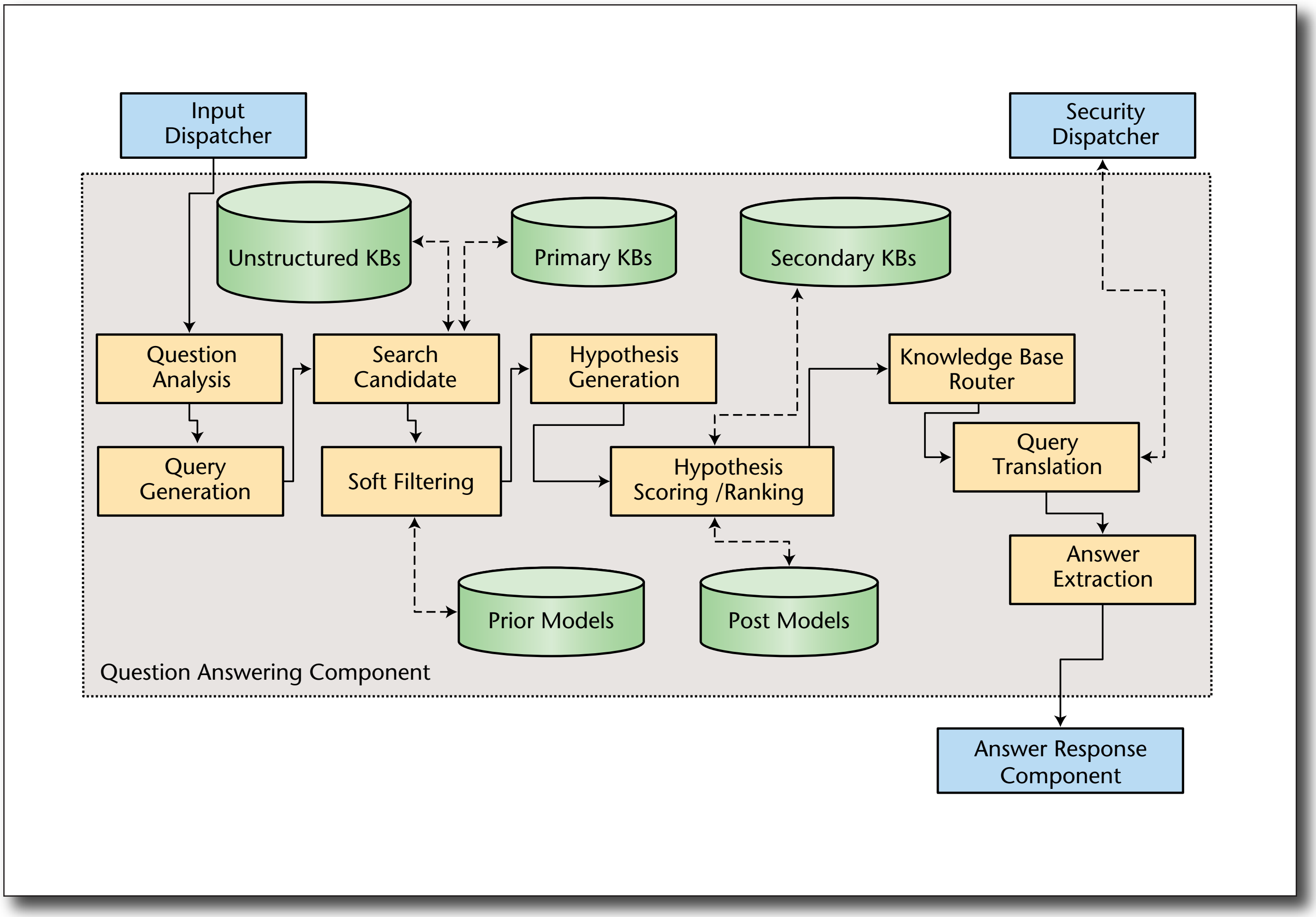


Figure 6. Overview of the USI Answers Workflow with Regard to the Individual Components.

several interviews with a domain expert and formalized by knowledge engineers. We continue to enrich and refine this knowledge.

Question Analysis

One of the first components that is applied within the question-answering process is question analysis. Within this analysis step, the system normalizes the input stream, as passed through the input dispatcher, by applying question normalization, metadata annotation, question parsing, analysis validation, and question classification.

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 metadata 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[12] (see figure 4 for an example of the annotation layers with regard to the question parsing component).

Analysis Validation: This step is needed so the system can handle domain-specific input terms such as product numbers and serial codes (for example, 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 revalidate the entity information. There are currently eight rules, all domain specific (for example, an x between two numbers is a wildcard for any expression between them (for example, 25x3a1 1 ↦ 25A3a1;25B3a1;25C3a1;).

Question Classification: Focuses on the identification of the answer signature, that is, analyzing the question type (for example, factoid or list based), the representation mode (for example, direct answer, SQL

A.

```
the input serial number > 1968:
con(Serial No.); rel(larger than); ins(1968);
hypothesis1(con,rel,ins);
confidence1(hypothesis1,0.76);
```

B.

```
con1(country code); rel(isNot); ins1(cn);
con2(country name); rel(isNot); ins2(china);

hypothesis1(con1,rel,ins1);
hypothesis2(con2,rel,ins2);
hypothesis3(con2,rel,ins1);

confidence1(hypothesis1,0.84);
confidence2(hypothesis2,0.77);
confidence3(hypothesis3,0.09);
```

Figure 7. Sample Input.

(a) Input for the Serial Number > 1968 (Top). (b) Input for the query (sub)phrase country is not cn:

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 2. Example Ranking of Concept Hypothesis Relations for the Lexical Entries ST and 1968.

Each lexical entry is interlinked to a list of possible interpretations (for example, Elem. Co.,...), relations types (Rt), and confidence scores (Conf.)

statement), and the question focus (for example, 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 (for example, a WP-VBZ-[NE/PER] sequence refers to a PERSON; WDT-NNS-VBP refers to a list-based question).

For each of the corresponding modules an UIMA annotator has been developed and incorporated, as the Question Analysis component in the overall QA processing pipeline (see figure 6).

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. 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 lookups, 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 in table 1, the system tries to search for and distinguish between concepts (called answerFields — for example, PAC Date), concept values instances (called searchFields — for example, operating) or already augmented key-value pairs (called domainFields — for example, Country Name : China). It accesses the primary knowledge bases using the query-generation component in order to gather hints on the actual type of information of concepts in the input question. Note that this component can also analyze multiword units, which are mapped into the KB representation space. For example, the input query sezela sugar generates the following concepts hints: sezela 1 $\mapsto$ Company Name; sugar 1 $\mapsto$ Industry Branch; sezela sugar 1 $\mapsto$ Plant Name. 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 end point trying to collect different surface forms of a single entity (for example, GE versus General Electric). The open domain knowledge module collects data as gathered within the DBpedia data set (Auer et al. 2008). For each of these components a 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 by the candidate search component. Based on prelearned prior models, we remove first relationships (for example, searchFields annotations) and initially rank the different annotations referenced to the respective query question tokens (for example, MW has a higher probability to refer to megawatt than to milliwatt, or the query item coal has a higher probability to be related to fuel than to be a subpart of a unit name).

This component is important as the different connected knowledge bases may assign a number of different annotations to single and multiple terms of the input question. The prior models have been learned by means of analyzing a list of existing SQL queries, which were used by the end users for information access prior to having the QA system accessible.

Hypotheses Generation

This component generates different question interpretations (that is, hypotheses of what the question might mean). More precisely, on the basis of the candidate search component, it generates different hypotheses of how the answerFields (concept) and searchFields (instance) are connected to each other (for example, direct or implicit) (relation). For an example, see the input listed in figure 7.

Because each concept and value have multiple representations assigned, which may be connected over multiple relationships (for example, textual representation of date or numbers), the list of different hypotheses can get very complex. In addition, this component gathers also hypotheses (on for example, geo-reasoning) that need to be applied if the focus of the question is targeting a location. An RDF-based open topic grammar gathers hypotheses 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* (for example, http://dbpedia.org/ontology/abstract) for definition answers.

Hypothesis Scoring and Ranking

The next component in the question-answering pipeline uses the different hypotheses generated so far to assign confidence scores, which indicate the probability that a given surface (terms/phrase) is represented through a given concept (see table 2). For example, the phrase: "country is not cn" is mapped to *country-name* ! = *china* || *country-code* ! = *cn*.

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 *cn* as a possible country code that may be a represented through the label *china*, 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 revalidate and rerank 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 database 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 1770 user-defined views for training.[13] In the last step, after all confidence scores are assigned to the different hypotheses, a final interpretation object is built to be passed to the knowledge base router module.

```
confidence1(hypothesis1,0.98, datasource1);
confidence2(hypothesis2,0.56, datasource1);
confidence3(hypothesis3,0.88, datasource2);
```

Figure 8. Knowledge Base Router Ranking.

Knowledge Base Router

As our primary knowledge sources consist of multiple database 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 the generated hypotheses, it collects relevant data sources and ranks them by the number of hypotheses that refer to them and their associated confidences (figure 8).

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 or unstructured data sources to answer the question.

Query Translation

The query translation component uses the information gathered by the hypothesis ranking and knowledge base router modules in order to construct the final query in the representative query language. Within the current setup, the system supports four different translation modes. It automatically constructs SQL, Apache Lucene, SPARQL, and Solution Object queries. The latter refers to domain-specific object representation used by another USI 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 regard to definitional questions, or applies the answer postprocessor by using postcalculation and computation. In case the answer needs to be generated through a SQL or SPARQL query, the query is passed direct to the answer manager within the answer mode representation. Finally, based on the resultant answer mode, if present, the direct answer or the answer query is passed to the output dispatcher.

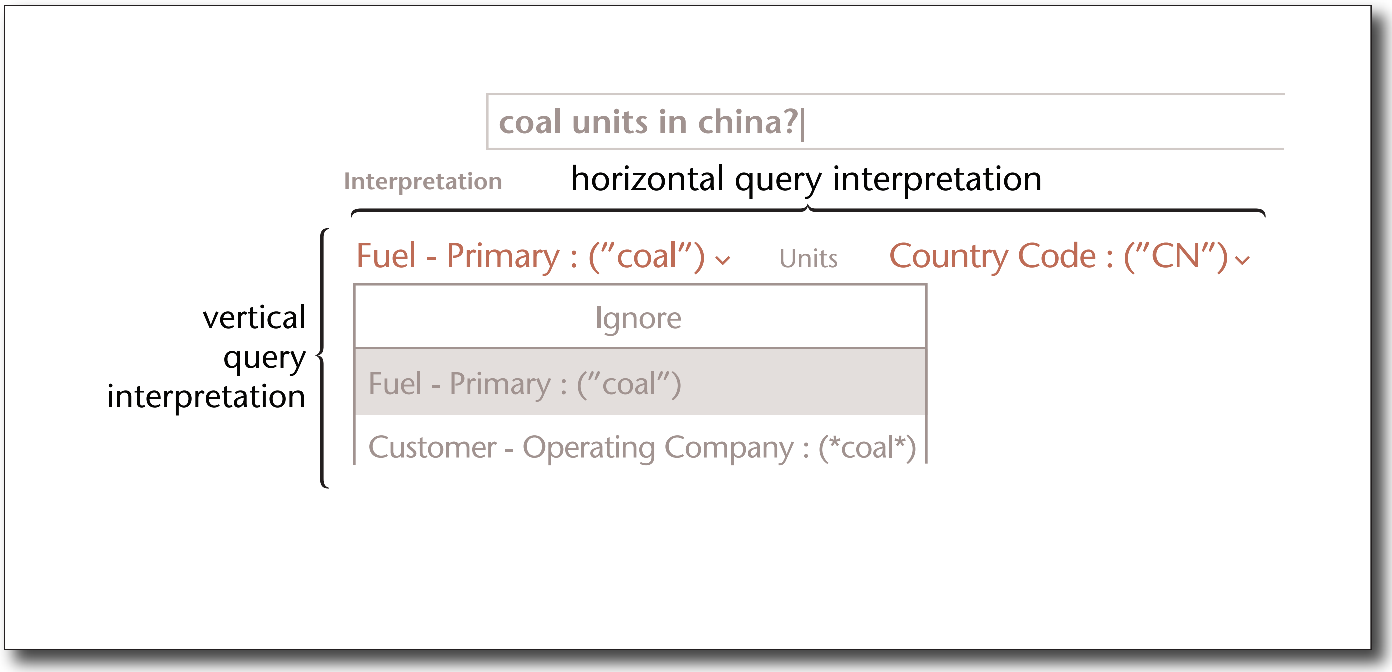

coal units in china?
Interpretation
horizontal query interpretation
vertical query interpretation
Fuel - Primary : ("coal")
Units
Country Code : ("CN")
Ignore
Fuel - Primary : ("coal")
Customer - Operating Company : (*coal*)

Figure 9. UI for Managing and Displaying Multiple Interpretations.

Each input question that is processed by the question-answering system is visually structured by means of the identified vertical and horizontal interpretation representations. The user is able to change or reorder each of the interpretations generated within the horizontal and vertical interpretation representation.

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

Table 3. Question Types Supported by USI Answers.

Answer Manager

The answer manager coordinates the back-end and front-end communication within the question answering. It executes the query by means of the respective representation mode (for example, SQL, SPARQL, or direct answer) and communicates the results through an interface to the front end. In addition, this component provides a number of web services with regard to geographic and statistical visualization of the answers. Moreover, as depicted in figure 9, the system visualizes the different interpretations as a list of key-value pairs (that is, Fuel — Primary : coal; Country Code : MX). We refer to the confidence-ranked interpretation list within a single information unit as the vertical query interpretation (that is, coal refers to fuel primary but it can be a part of a company name), and to the confidence-ranked interpretation list of interlinked information units (that is, coal — china) as the horizontal query interpretation representation. The user is able to select, change, or reorder each of the interpretations generated within the horizontal and vertical interpretation representation.

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 group. A user can belong to one or more such groups.

Development, Deployment, and User Experience

The system was developed by a team of about 10 to 12 engineers and scientists from Siemens Corporation, Corporate Technology, located in the United States, Germany, and Romania, over the course of 3 years.[14] USI Answers has been in use by Siemens Energy Service since May 2012 and is being accessed regularly by more than 1500 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 natural language query integration simplified the 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 the 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 searches dropped by 90 percent. The number of failure points (that is, 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 user retention more than doubled. Usage of the system is optional.

User feedback also shifted from comments such as "system is impressive but too overwhelming" (before deployment of USI Answers) to "system thinks like me. It is very intuitive!"

Experiments

In the absence of an existing evaluation corpus for automatic SQL conversion of natural langauge questions, we constructed such a corpus, with a special emphasis on abstract question/query types (see table 3). The goals of the experiments conducted were threefold. First, we were interested in the overall performance of hypothesis generation and ranking by directly observing the concept-instance pairs generated by the system. Is the system able to rank and assign the right concept relation just by 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 postmodel learning components within the USI Answers. For all experiments, we applied a fivefold cross-validation measured answer relevance as recall and precision at rank k ($P@k$), for k = 1, 5, and 10.

Data Set

The reference corpus (that is, gold standard) used in the evaluation was compiled from 1770 database SQL queries called named views, which have been converted into an abstract key-value representation (see example template below). These views were collected from 395 users that could define such database named views by means of aggregating different SQL commands through a query builder interface and provide a name for each of them (figure 10).

We have used only those views that had a minimum number of two key-value pairs. The resultant reference corpus consisted of 1142 named views, with 8733 key-value pairs. Note that, on average, each SQL view comprises 7.6 key-value pairs. For the experiments, all key references (for example, generator frame) have been removed. That is only the respective values (for example, 115/36) have been used as an input to the QA system. See figure 11 for an example input question representation.

For each input representation, the system was evaluated by measuring the hypothesis ranking of the

```
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
  ...
  )
```

Figure 10. Example Entry of a Named View Used for Evaluation.

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

Figure 11. Example Input Question Representation.

From figure 8, used as an input 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'
```

Figure 12. Generated SQL Result Entry Example, as Constructed Within the Evaluation Setup.

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

Table 4. Experimental Results.

Results of the evaluation experiments using 8733 key-value pairs from 1142 named views. Each key has been deleted and the system tried to recover it.

All	No Post	No Prior	No Init
0.921	0.921	0.920	0.858
0.908	0.901	0.898	0.742
0.957	0.949	0.955	0.895
0.974	0.969	0.973	0.935

Table 5. Experimental Results.

Results of the evaluation experiments using 1142 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 gold-standard view name.

concept-instance pairs, as well as with regard to the prediction of the full initial input query. As an answer signature, the automatic SQL translation mode has been used (figure 12).

Results

Experimental results are summarized in Tables 4 and 5. In order to evaluate the contribution of each component in the QA pipeline, we constructed four 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 results of analysis of the hypothesis generation and ranking components of the system, done by means of observing concept instance pairs, as depicted in table 4, show a recall of 0.948 and a precision at rank 1 of 0.765, a good performance with regard to accuracy, and a very good performance with regard to recall. The experiments on the entire view name prediction, as depicted in table 5, highlight again the effectiveness of the system, even though handling only a partial reconstructed representation. More precisely, as table 5 shows, applying the QA pipeline on the entire query, with an average of 7.6 key-value pairs, the system is able to reconstruct concept-instance references with a precision of 0.765 (see table 4), but is still able to rank first the full correct SQL view, with a *p*@1 of 0.908.

Analyzing the individual components of the QA pipeline, a clear influence of the learning modules can be identified. Note that the postmodel 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 setups 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 five-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 (for example, SGT6-5000F — W501F versus 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 preprocessing errors.

Conclusion and Outlook

With the advent of big data there is a growing need to offer easy access to diverse groups of users that could utilize such data. We developed USI Answers, a natural language question-answering system for semistructured Siemens Energy data. The system offers easy access to enterprise data to thousands of business users in 18 countries. It enables users with limited familiarity with technical systems and databases to ask questions in a natural way and gain actionable insight from business as well as public data (for example, news). It makes apparent the system's interpretation of the query and allows easy query adjustment and reformulation. We evaluated our approach on a data set 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, we believe the architecture described here is applicable to other use cases and data coming from domains such as health care and industry.

Notes

1. See www.wolframalpha.com.
2. See www.www.apple.com/de/ios/siri.
3. See table 2 for the whole set of answer types.
4. Unstructured Information Management Architecture. See /uima.apache.org.
5. See www.protege.stanford.edu.
6. Lucene is a high-performance, full-featured text search engine library. See lucene.apache.org/core.
7. See www.www.dbpedia.org/.
8. See www.www.freebase.com/.
9. See www.www.geonames.org/.
10. RDF is a standard model for data interchange on the

web (www.w3.org/RDF). OWL builds on top of RDF by providing language for defining structured, web-based ontologies that enable richer integration and interoperability of data.

11. AllegroGraph is a modern, high-performance, persistent graph database (www.franz.com/agraph/allegrograph).

12. See www.nlp.stanford.edu/software/corenlp.shtml.

13. Note that for the experiments we have applied a five-fold cross-validation, thus we have not trained on the entire set.

14. This is the total time that it took to develop the whole USI project, of which USI Answers is a part.

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),* Lecture Notes in Computer Science, volume 4825, 722–735. Berlin: 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. Stroudsburg, PA: Association for Computational Linguistics. dx.doi.org/10.3115/992628.992654

Chuba, M. 2012. Data Center Executives Must Address Many Issues in 2012. Gartner Group Report G00229650, 31 January. Stamford, CT: Gartner Group.

de Marneffe, M.-C.; MacCartney, B.; and Manning, C. D. 2006. Generating Typed Dependency Parses from Phrase Structure Trees. In *Proceedings of the International Conference on Language Resources and Evaluation.* Paris: European Language Resources Association.

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 of the Association for Computational Linguistics,* 54–61. Stroudsburg, PA: Association for Computational Linguistics. dx.doi.org/10.3115/981574.981582

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): 50–79.

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): 33–58.

Hatcher, E.; Gospodnetic, O.; and McCandless, M. 2010. *Lucene in Action,* 2nd revised edition. Shelter Island, NY: Manning Publications Co.

Kate, R. J., and Mooney, R. J. 2006. Using String-Kernels for Learning Semantic Parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics,* 913–920. Stroudsburg, PA: 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. Menlo Park, CA: AAAI Press.

Lu, W.; Ng, H.; Lee, W.; and Zettlemoyer, L. 2008. A Generative Model for Parsing Natural Language to Meaning Representations. In *Proceedings of the Empirical Methods on Natural Language Processing,* 783–792. Stroudsburg, PA: 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. Stroudsburg, PA: Association for Computational Linguistics. dx.doi.org/10.3115/981863.981871

Ogren, P. V.; Wetzler, P. G.; and Bethard, S. 2008. ClearTK: A UIMA Toolkit for Statistical Natural Language Processing. Paper presented at the Unstructured Information Management Architecture Workshop, Marrakech, Morocco, 31 May.

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. Washington, DC: 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 Joint Conference on Artificial Intelligence,* 1549–1554. San Francisco: Morgan Kaufmann Publishers Inc.

Waltinger, U. 2011. On Social Semantics in Information Retrieval. Ph.D. Thesis, Bielefeld University, Bielefeld, Germany.

Waltinger, U.; Breuing, A.; and Wachsmuth, I. 2011. Interfacing Virtual Agents with Collaborative Knowledge: Open Domain Question Answering Using Wikipedia-Based Topic Models. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence,* 1896–1902. Menlo Park, CA: AAAI Press.

Waltinger, U.; Breuing, A.; and Wachsmuth, I. 2012. Connecting Question Answering and Conversational Agents — Contextualizing German Questions for Interactive Question Answering Systems. *Künstliche Intelligenz* 26(4): 381–390. dx.doi.org/10.1007/s13218-012-0208-1

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 the Human Language Technology Conference, North American Chapter of the Association for Computational Linguistics* (HLT-NAACL), 439–446. Stroudsburg, PA: Association for Computational Linguistics.

Wong, Y. W., and Mooney, R. J. 2007. Generation by Inverting a Semantic Parser That Uses Statistical Machine Translation. In *Proceedings of the Human Language Technology Conference, North American Chapter of the Association for Computational Linguistics* (HLT-NAACL), 172–179. Stroudsburg, PA: Association for Computational Linguistics. dx.doi.org/10.3115/1220835.1220891

Woods, W. 1970. Transition Network Grammars for Natural Language Analysis. *Communications of the ACM* 13(10): 591–606. dx.doi.org/10.1145/355598.362773

Zelle, J. M., and Mooney, R. J. 1996. Learning to Parse Database Queries Using Inductive Logic Programming. In *Pro-*

ceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96), 1050–1055. Menlo Park, CA: AAAI Press / The MIT Press.

Zettlemoyer, L., and Collins, M. 2005. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. In *Proceedings of 21st Conference on Uncertainty in Artificial Intelligence* (UAI). Seattle, WA: Association for Uncertainty in Artificial Intelligence.

Ulli Waltinger is an engineer at Siemens Corporate Technology, Research and Technology Centre in Munich, Germany. His backround is in artificial intelligence, information retrieval, and computational linguistics. His primary research interests include machine learning, natural language processing, and question answering. Waltinger has contributed to journals and conferences in artifical intelligence, natural language processing, and cognitive science. In 2010, Ulli Waltinger earned his Ph.D. in computer science from the Technical Faculty at Bielefeld University.

Dan G. Tecuci is currently a research and development project manager with Siemens Corporation, Corporate Technology in Princeton, NJ, leading a team of researchers and developers working on text analytics and reasoning applications. He received a Ph.D. in artificial intelligence from the University of Texas at Austin in 2007. He is interested in applying principles of knowledge representation and reasoning, natural language processing, and question answering to practical problems in domains like energy, health care, and intellectual property.

Mihaela Olteanu graduated from the University Transilvania of Brasov, Romania, in 2008 with a B.Sc. in computer science. She obtained an M.Sc. in the field of applied computer science: modern technologies in software development. In 2013 she became an Oracle Certified Professional Java SE 6 Programmer. Mihaela Olteanu is currently a software engineer at Siemens Corporate Technology, Research and Technology Centre in Brasov, Romania. Her professional experience covers Java Enterprise Edition, service-oriented architecture, distributed systems, big data, information retrieval, and the semantic web.

Vlad Mocanu graduated from the University Transilvania of Brasov, Romania, in 2011 with a B.Sc. in computer science. He completed an internship at IBM GDC Brasov, Romania, in 2010, where he handled web design and client-side tasks. In 2013 he became an Oracle Certified Professional Java SE 6 Programmer. Vlad Mocanu joined Siemens Corporate Technology, Research and Technology Centre from Brasov, Romania in 2011 as a software engineer. His professional experience covers web design, cross-browser compatibility, client-side scripting, asynchronous communication, and general user experience.

Sean Sullivan is the PLM business architect for Siemens Energy Service Fossil Division. He is the head of the unified service intelligence initiative.