Automatic knowledge extraction from documents

J. Fan
A. Kalyanpur
D. C. Gondek
D. A. Ferrucci

Access to a large amount of knowledge is critical for success at answering open-domain questions for DeepQA systems such as IBM Watson™. Formal representation of knowledge has the advantage of being easy to reason with, but acquisition of structured knowledge in open domains from unstructured data is often difficult and expensive. Our central hypothesis is that shallow syntactic knowledge and its implied semantics can be easily acquired and can be used in many areas of a question-answering system. We take a two-stage approach to extract the syntactic knowledge and implied semantics. First, shallow knowledge from large collections of documents is automatically extracted. Second, additional semantics are inferred from aggregate statistics of the automatically extracted shallow knowledge. In this paper, we describe in detail what kind of shallow knowledge is extracted, how it is automatically done from a large corpus, and how additional semantics are inferred from aggregate statistics. We also briefly discuss the various ways extracted knowledge is used throughout the IBM DeepQA system.

Introduction

As with many problems in artificial intelligence, knowledge is an important element of the IBM Watson* solution to the open-domain question-answering problem [1]. Although formally represented deep semantic knowledge is easy to reason with when compared with other forms of representation and can be used for deep inference, acquisition of structured knowledge in open domains from unstructured data is difficult and expensive. Unfortunately, much of human knowledge is expressed in the form of unstructured text. Take the following example passage:

> Patrick "Pat" Floyd Garrett (June 5, 1850–February 29, 1908) was an American Old West lawman, bartender, and customs agent. Garrett rode as a cowhand in Texas, and he served drinks at the Beaver Smith's saloon at Fort Sumner, New Mexico.

This unstructured text contains useful knowledge, such as the birthdate, death date, and occupation of Pat Garrett, but efficiently extracting such knowledge is difficult. It requires correctly parsing the sentences, identifying key entities, type information, and relationship information, and performing co-reference resolution to merge information about the same entity.

On the other hand, one may still infer important semantic information from large amounts of shallow knowledge. For example, from the example text above and thousands of other sentences about Pat Garrett, we can infer many assertions, such as Pat Garrett is a lawman and sheriff; the most common actions that Pat Garrett was involved in are kill, shoot, and capture; and the most common object of these actions is Billy the Kid. We can also infer terminological axioms such as the type. "People" is often the subject and object of a kill action.

Our central hypothesis is that shallow syntactic knowledge and its implied semantics can be easily acquired, and it can be used in many areas of a question-answering system. For example, given the sentence "Einstein, who has published more than 300 scientific papers, won the Nobel Prize for Physics in 1921", using a dependency parser and a named entity recognizer (NER), we can extract and generalize type-based patterns such as "scientists publish papers", "scientists win Nobel prizes", "Nobel prizes are associated with subjects and years", etc., and gather statistics about these patterns from a large text corpus to build a

Digital Object Identifier: 10.1147/JRD.2012.2186519

0018-8646/12/$5.00 © 2012 IBM

large-scale shallow semantic knowledge base. The resulting knowledge base can be directly used to answer questions such as "In 1921, at the age of 42, he won his only Nobel Prize". Similarly, it can give additional evidence to support the answer "Carl Sagan" for the question "He published the first scientific paper to predict nuclear winters" because Carl Sagan is a scientist.

Existing linguistic resources such as VerbNet and FrameNet provide some selectional restriction information for verbs and frames. However, because they are manually built, they tend to specify type constraints at a very high level (e.g., agent, theme, and animate), and consequently, they do not suffice for cases such as the previous scientist-publishes-paper example.

We would like to infer more fine-grained knowledge for predicates automatically from a large amount of data. In addition, we do not want to restrict ourselves only to verbs, to binary semantic relations, or to a specific type hierarchy, because doing so will restrict the content of the resulting knowledge base. For a truly open-domain problem such as Jeopardy!**, such restriction will significantly affect the usefulness of the knowledge base. For example, lexical answer types (LATs)—question words that suggest the types of the correct answers—from past Jeopardy! questions include rare types such as *laureate*, *angel*, *monster*, *saxophonist*, and *conveyance*. If we limit the knowledge acquired to a specific type system, then we may not be able to obtain the knowledge that is critical for answering these questions. Similarly, many Jeopardy! questions refer to rare semantic relations or express semantic relations in novel ways (e.g., the *come-from* relation in "This winged creature sprang from the blood of Medusa") that may not be captured if we limit ourselves to only some parts of speech, a few semantic relations, or a specific type hierarchy.

We achieve fine-grained knowledge by taking a two-stage approach. First, shallow knowledge from large collections of documents is automatically extracted. Specifically, we are interested in extracting mostly syntactic relations within sentential contexts and representing them as frames and slots. For the previous example, frames such as "Einstein wins Nobel Prize" and "Einstein publishes papers" are extracted. Although syntactic relations do not capture all of the knowledge contained in a piece of text, the redundancy of the large corpus helps to increase the coverage of the resulting knowledge base.

Second, additional semantics are inferred from aggregate statistics of the automatically extracted shallow knowledge. Given the aggregate statistics of syntactic relation usage, we are able to use ontology-based generalization to infer assertional axioms, such as "the best known thing Einstein wins is a Nobel Prize", as well as terminological axioms, such as "scientists publish papers" and "scientists win Nobel prizes".

The extracted knowledge is used by Watson in several ways, such as generating answer candidates [2] for questions, coercing answer candidates into the LAT of a question [3], and finding missing links between clue terms and candidates in Final Jeopardy! [4].

In this paper, we present PRISMATIC, the implementation and the result of our two-stage approach that is used by Watson as a knowledge resource for question answering. In the following sections, we describe PRISMATIC in detail. PRISMATIC proves to have a significant impact at both the component and the system level, most importantly improving the overall accuracy of the Watson system by 2.4% on PRISMATIC's enhancements in type coercion alone.

Terminology

We have adopted the following terminology in the development of PRISMATIC.

- *Frame*—A frame is the basic semantic unit representing a set of entities and their relations in a piece of text. A frame is made of a set of slot and value pairs. A frame can be either intentional or extensional on the basis of whether its values are instances or types.
- *Slot*—A slot in PRISMATIC is a binary relation. Most of the slots in PRISMATIC are dependency relations extracted from the parse tree. **Table 1** lists the slots used in PRISMATIC.
- *Slot value*—A slot value is either the lemma form of a term from the sentence being extracted or a type annotated by an NER. **Table 2** shows the extracted frame (i.e., list of slots and their values) based on the parse tree in **Figure 1**. Notice that in Table 2 annotated types are shown in all uppercase.
- *Frame projection*—A frame projection is a portion of a frame (or the operation that produces the portion) that occurs with regularity in many frames and is of particular interest to the users of PRISMATIC. For example, the portion of a frame that includes only subject, verb, and object (i.e., S-V-O projection) is particularly useful for analyzing the selectional preference of verbs.

PRISMATIC can be formally defined as a bag of frames, i.e.,

$$P = \{f_1, f_2, \ldots, f_n\},$$

where $f_i = \{\langle s_1, v_1\rangle, \langle s_2, v_2\rangle, \ldots, \langle s_m, v_m\rangle\}$ is a frame made of a set of slot–value pairs $\langle s_j, v_j\rangle$.

We use $V(s,f)$ to denote the value of slot s on frame f.

A frame projection $C \subseteq P$ is a subbag of frames that have nonempty slot values for a given subset of all slots, i.e., $S_C \subseteq S$. Thus

$$C = \{f \in P | \forall_{s \in S_C} V(s,f) \neq \text{nil}\}.$$

Table 1 Relations used in a frame and their descriptions. (Used with permission from J. Fan, D. Ferrucci, D. Gondek, and A. Kalyanpur, "PRISMATIC: inducing knowledge from a large scale lexicalized relation resource," *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pp. 122–127, Los Angeles, CA, Association for Computational Linguistics, June 2010.)

Relation and description	*Example*
subj	Subject
obj	Direct object
iobj	Indirect object
comp	Complement
pred	Predicate complement
objprep	Object of the preposition
mod_nprep	*Bat Cave in Toronto is a tourist attraction*
mod_vprep	*He made it to Broadway*
mod_nobj	The object of a nomialized verb
mod_ndet	*City's budget was passed*
mod_ncomp	*Tweet is a word for microblogging*
mod_nsubj	*A poem by Byron*
mod_aobj	*John is similar to Steve*
isa	Subsumption relation
subtypeOf	Subsumption relation

Table 2 Frames extracted from dependency parser in Figure 1. (Used with permission from J. Fan, D. Ferrucci, D. Gondek, and A. Kalyanpur, "PRISMATIC: inducing knowledge from a large scale lexicalized relation resource," *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pp. 122–127, Los Angeles, CA, Association for Computational Linguistics, June 2010.)

	Frame01
verb	receive
subj	Einstein
type	PERSON/SCIENTIST
obj	Nobel prize
mod_vprep	in
objprep	1921
type	YEAR
mod_vprep	for
objprep	Frame02
	Frame02
noun	work
mod_ndet	his/Einstein
mod_nobj	on
objprep	effect

System overview

PRISMATIC is built using a suite of natural-language processing (NLP) tools that include a dependency parser, a rule-based NER, and a co-reference resolution component [5]. The PRISMATIC creation process consists of three phases.

1. *Corpus processing*—Documents are annotated by a suite of components that perform dependency parsing, co-reference resolution, NER, and relation detection.
2. *Frame extraction*—Frames are extracted on the basis of the dependency parses and associated annotations. This phase implements the first stage of our approach.
3. *Frame projection*—Frame projections of interest (e.g., S-V-O projections) are identified over all frames, and frequency information for each projection is tabulated. This phase produces the aggregate statistics from the extracted frames used to infer additional semantics.

Figure 2 shows the three phases of PRISMATIC, which are described in detail in the following subsections.

Corpus processing

The key step in the corpus processing phase is the application of a dependency parser that is used to identify the frame slots (as listed in Table 1) for the frame extraction stage. We use ESG (English Slot Grammar) [5], which is a Slot Grammar-based parser, in order to fill in the frame slots. Sentences frequently require co-reference in order to precisely identify the participating entity. In order to not lose that information, we apply a simple rule-based co-reference resolution component [5] in this phase. The co-reference information helps to enhance the coverage of the frame projections, which is particularly valuable in cases of sparse

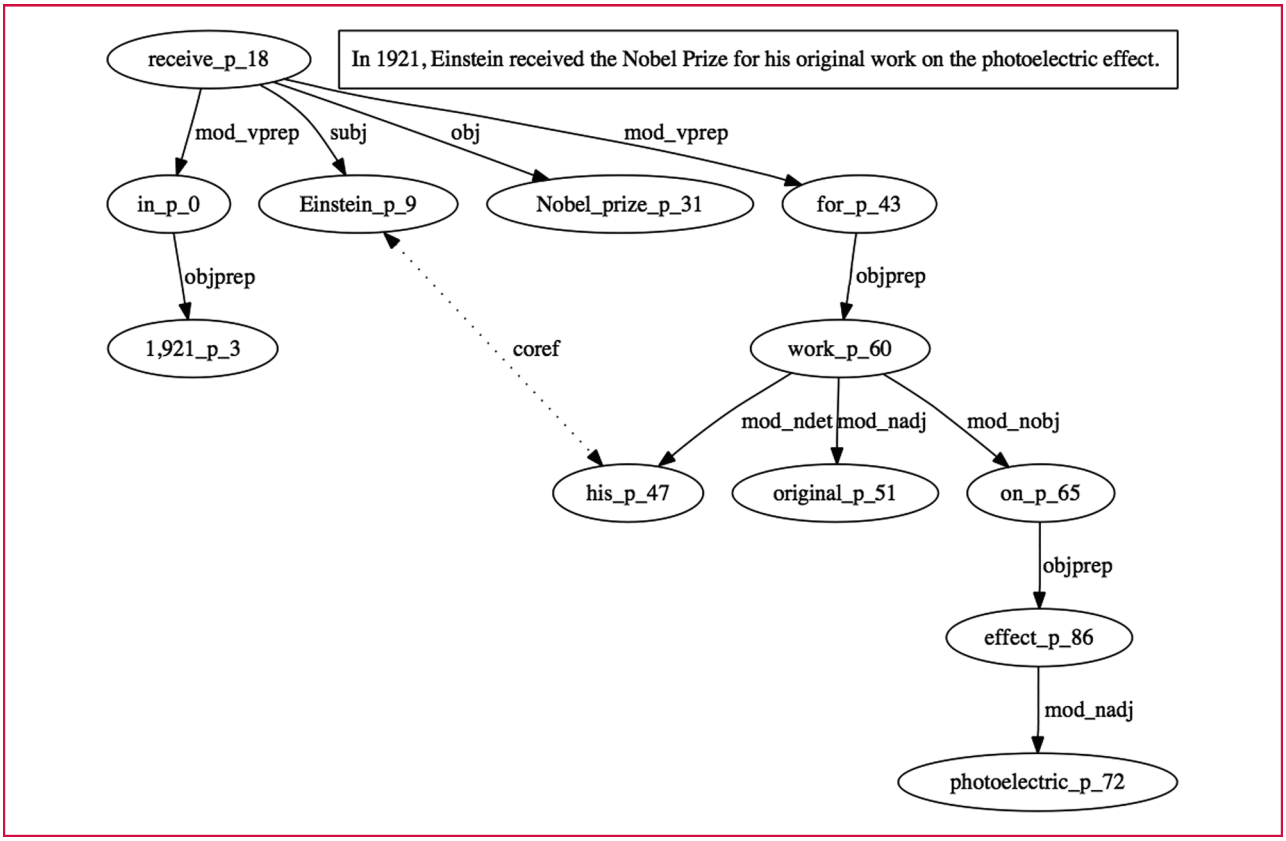

Figure 1

Parse tree of the sentence "In 1921, Einstein received the Nobel Prize for his original work on the photoelectric effect". (Used with permission from J. Fan, D. Ferrucci, D. Gondek, and A. Kalyanpur, "PRISMATIC: inducing knowledge from a large scale lexicalized relation resource," *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pp. 122–127, Los Angeles, CA, Association for Computational Linguistics, June 2010.)

data. Relation detectors are also applied during the corpus processing phase to fill frame slots not captured by ESG (e.g., *isa* slot).

Semantic annotation

The parse trees produce axioms on entities similar to the assertional axioms in description logic, but, in order to include terminological axioms in PRISMATIC, ontology-based generalization is needed. For this purpose, a rule-based NER [5] is used to identify the types of slot-fillers. The types of slot-fillers can also be recognized by any other independently developed NERs (rule-based or statistical). This type information is then registered in the frame extraction stage to construct intentional frames.

Frame extraction

The next step of PRISMATIC is to extract a set of frames from the parsed corpus. In order to capture the relationship we are interested in, frame elements are limited to those that represent the participant information of a predicate. Slots consist of the ones listed in Table 1. Furthermore, each frame is restricted to be two levels deep at the most; therefore, a single large parse tree may result in multiple frames. Table 2 shows how two frames are extracted from the complex parse tree in Figure 1. The depth restriction is needed for two reasons. First, despite the best efforts from parser researchers, no parser is perfect, and big complex parse trees tend to have more incorrect parses. By limiting a frame to be only a small subset of a complex parse tree, we reduce the chance of parse error in each frame. Second, by isolating a subtree, each frame focuses on the immediate participants of a predicate.

In addition to parser relations, semantic relations may also be added to a frame. For example, the *isa* slot in Table 1 is based the *isa* semantic relation detected by a syntactic pattern-based relation detector [6].

Nonparser information may also be included in a frame. For example, the type annotations of a word from an NER

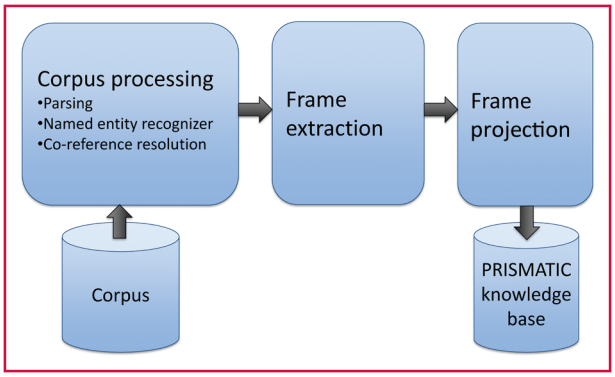

Figure 2

Main stages of PRISMATIC construction. (Used with permission from J. Fan, D. Ferrucci, D. Gondek, and A. Kalyanpur, "PRISMATIC: inducing knowledge from a large scale lexicalized relation resource," *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pp. 122–127, Los Angeles, CA, Association for Computational Linguistics, June 2010.)

are included, and such type information is useful for the various applications described in the section "Applications in Watson and beyond." We also include a flag to indicate whether a word is a proper noun. These two kinds of information allow us to easily separate the intentional and the extensional parts of PRISMATIC.

The result of frame extraction is a large set of frames representing shallow knowledge extracted from the input corpus. These frames are used as inputs for the frame projection phase described next.

Frame projection

One of the main reasons for extracting a large amount of frame data from a corpus is to induce interesting knowledge patterns by exploiting redundancy in the data. For example, we would like to learn that things that are *annexed* are typically regions, i.e., a predominant object type for the noun phrase *annexation of* is *Region*. To do this kind of knowledge induction, we first need to isolate and analyze specific portions of the frame—in this particular case, the noun-phrase → object-type relationship. Then, given multiple frames containing only the above relationship, we hope to see the frame $\{\langle noun, \text{“}annexation\text{”}\rangle, \langle preposition, \text{“}of\text{”}\rangle, \langle object\text{-}type, \text{“}Region\text{”}\rangle\}$ occur very frequently.

To enable this induction analysis, we define frame projections, which specify a projection operation on a frame. For example, we define an N-P-OT frame projection, which, when applied to a frame, keeps only the noun (N), preposition (P), and object-type (OT) slots and discards the rest. Similarly, we define frame projections such as S-V-O, S-V-O-IO, S-V-P-O, N-Isa, N-Isa-Mod, etc. (where S is subject, V is verb, O is object, IO is indirect object, N is noun, Isa is *isa* slot, and Mod is modifier), which all dissect frames along different dimensions. We use V-OT and N-Isa to illustrate the usefulness of frame projections. Continuing with the *annexation* example, we can use the V-OT frame projection to learn that a predominant object type for the verb *annex* is also *Region*, by seeing frames of the form $\{\langle verb, \text{“}annex\text{”}\rangle, \langle object\text{-}type, \text{“}Region\text{”}\rangle\}$ multiple times in our data. Such intentional frames are terminological axioms that can be used to learn entailment rules. The N-Isa frame projection shows the lexical types of nouns independent of any predefined ontology. For example, by seeing multiple frames of the form $\{\langle noun, \text{“}Bill\ Clinton\text{”}\rangle, \langle isa, \text{“}politician\text{”}\rangle\}$, we can learn that *Bill Clinton* is often referred to as a *politician*.

To make frame projections more flexible, we allow them to specify optional value constraints for slots. For example, we can define an S-V-O frame projection, where both the subject (S) and object (O) slot values are constrained to be proper nouns, thereby creating strictly extensional frames, i.e., frames containing data about instances, e.g., $\{\langle subject, \text{“}United\ States\text{”}\rangle\langle verb, \text{“}annex\text{”}\rangle \langle object, \text{“}Texas\text{”}\rangle\}$. The extensional frames are assertional axioms that can be used to provide factual evidence in tasks such as question answering.

Frame projections allow us to precompute and look up aggregate statistics quickly. In the next section, we describe some commonly used aggregate statistics.

Aggregate statistics

There are a variety of useful aggregate statistics that can be gleaned from the resulting frame projections. We list three examples here.

1. *Frequency*—The frequency of a particular frame (or slot–value pairs) can be defined as $\#(f) = |\{f_i \in P | \forall_{s \in S} V(s,f) = V(s,f_i)\}|$. It computes the number of frames whose slot values match with all the slot–value pairs in a given frame. Frequency gives us an estimate of the popularity of fillers for a given set of slots. For example, the frequency of $\#(\{\langle subj, \text{“}Einstein\text{”}\rangle, \langle verb, \text{“}win\text{”}\rangle, \langle obj, \text{“}award\text{”}\rangle\}) = 142$ may suggest that *Einstein winning awards* is a set of popular fillers in the corpus. However, frequency value is also affected by the size of the corpus and the popularities of individual fillers, e.g., *Einstein* may frequently occur throughout the corpus, and hence, *Einstein winning award* may often appear.
2. *Conditional probability*—The conditional probability of a particular frame can be estimated by $p(f|f') = [\#(f)/\#(f')]$, where $f' \subseteq f$. It estimates the probability of a specific set of fillers given a subset of such slot-filler pairs.

For example, let $f = \{\langle subj, \text{“}Einstein\text{”}\rangle, \langle verb, \text{“}win\text{”}\rangle, \langle obj, \text{“}award\text{”}\rangle\}$, $f' = \{\langle subj, \text{“}Einstein\text{”}\rangle, \langle verb, \text{“}win\text{”}\rangle\}$, then $p(f|f') = 0.32$ means that 32% of the time when Einstein wins something, he wins an award. Unlike frequency, it is easier to compare one conditional probability with another because conditional probability is less affected by the corpus size.

3. *Normalized pointwise mutual information*—The normalized pointwise mutual information (NPMI) of two frames is defined as

$$npmi(f, f') = \frac{pmi(f, f')}{-\ln \frac{(\max[\#(f), \#(f')])}{N}},$$

where $pmi(f, f') = \ln N(\#(f \cup f')/\#(f) \times \#(f'))$ (N is the size of a particular frame projection). NPMI ranges from -1 (for no co-occurrence) to 1 (for complete co-occurrence). For example, let $f = \{\langle obj, \text{“}award\text{”}\rangle\}$ and $f' = \{\langle subj, \text{“}Einstein\text{”}\rangle, \langle verb, \text{“}win\text{”}\rangle\}$; if $npmi(f, f') = 0.7$, then it indicates a high degree of co-occurrence between *Einstein winning* and *award*. Compared with conditional probability, NPMI takes into consideration the popularities of different subsets of a frame and scores them accordingly.

Evaluation

There are two kinds of metrics for evaluating a knowledge base: component- and system-level metrics. The component-level evaluation judges the quality of the knowledge base independently of the specific applications. Typically, a component-level evaluation uses metrics such as precision and recall to estimate the correctness and coverage of the knowledge base. However, neither precision nor recall is easy to obtain in the case of PRISMATIC because it is difficult to judge the correctness of a frame. Other machine learning systems, such as TextRunner [7], report precision and recall using cross validation on their training data, and these results reflect the performance of their underlying relation detection and parsing components. In contrast, PRISMATIC performance is gated by its NLP stack whose parser is reported to have an accuracy of 89.3% [5]. None of these numbers reflects the portion of information in a corpus that is captured by PRISMATIC. In the next section, we present some evaluation results on the coverage of PRISMATIC.

Coverage

PRISMATIC contains a large amount of information. The corpus we used to produce the initial PRISMATIC is based on a selected set of sources, such as encyclopedias, news articles, and other sources. After the corpus was cleaned and HTML tags removed, there is 30 GB of text. From these sources, we extracted approximately 995 million frames. From these frames, we produced the most commonly used projections such as S-V-O, S-V-P-O, and N-P-O.

Table 3 PRISMATIC's size, coverage, and impact.

Measure	*Value*
Corpus size	30 GB
Number of frames extracted	995 millions
Number of frames per sentence	1.4
Percentage of named entities that are extracted as part of a frame	94.4%
Watson Improvement with PRISMATIC TyCor	+2.4% in accuracy

Although we can measure the size of PRISMATIC, it does not show the coverage. Since directly measuring the recall is difficult, we have conducted the following study to gauge the coverage of PRISMATIC. First, we randomly sampled 140,000 documents, out of which PRISMATIC generates 1.3 frames per sentence. This suggests that there is at least one frame generated for each sentence on average. We also measured the coverage of PRISMATIC frames on the named entities in the sample. It produced frames for 94.4% of the named entities detected by the NLP components we used. Both results suggest that PRISMATIC is able to gather useful information on most of the entities from most of the sentences in the input corpus.

Table 3 shows the coverage evaluation results of PRISMATIC. The coverage experiments give us a rough idea of the quality of PRISMATIC; however, system-level evaluation, or how much impact PRISMATIC has on a variety of applications, is more important. In the next section, we discuss applications of PRISMATIC in Watson and their impact.

Applications in Watson and beyond

There are many potential applications that can utilize PRISMATIC, such as type inference, relation extraction, textual entailment, and more. We present applications of PRISMATIC in Watson in the following sections.

Type coercion

PRISMATIC is used to determine whether a candidate answer is of the LAT expressed by the question in Watson. Typically, a Jeopardy! question indicates the type of the correct answer through words in the clue. For example, the question “Senator Obama attended the 2006 groundbreaking for this man's memorial, 1/2 mile from Lincoln's” suggests that the answer type is *man*. Whether a candidate answer is of the correct LAT is an important feature of the candidate answer. We can utilize the aggregate statistic of $\{\langle noun, candidate\ answer\rangle\langle isa, lat\rangle\}$ frame to determine the likelihood that a candidate string is of the correct LAT. By including the PRISMATIC type coercion component,

the Watson question-answering system's overall accuracy improved by 2.4%, and the PRISMATIC type coercion component consistently ranks in the top 3 among 17 different type coercion components in Watson. More details on how PRISMATIC is used to judge the type correctness of a candidate can be found in [3].

Candidate generation

PRISMATIC is also used to generate candidate answers. Some Jeopardy! questions contain modifiers to the LAT, such as "While Maltese borrows many words from Italian, it developed from a dialect of this Semitic language." Notice that the LAT *language* has a modifier *Semitic*. PRISMATIC aggregate statistics on the isa slot and modifier slots are used to determine the top 20 most common instances of Semitic language, which include the correct answer Arabic. Although PRISMATIC candidate generation generates only up to 20 candidates per question, its candidates are more likely to be correct than other candidate generation techniques. On a test set of 3,508 Jeopardy! questions, the PRISMATIC candidate generator produces answers for 42.6% of them (1,495 questions). For example, given the question "To: junglechick. From: mr_simian. Subject: pronouns. Fyi, I do use "I" when referring to myself in a 1914 novel" with category "LITERARY CHARACTERS' E-MAILS". The question analysis component recognizes the LAT is *character* with *literary* as the modifier. Because of distracting terms such as *junglechick*, document/passage search on large corpora is unlikely to find the correct answer *Tarzan*. However, PRISMATIC candidate generation produces *Tarzan* as a candidate answer because it is the sixth most popular literary character based on PRISMATIC's aggregate statistics. In addition, the PRISMATIC candidate generator produces far fewer wrong candidate answers than other candidate generation techniques. On average, 1 out of 57 PRISMATIC candidates is the correct answer to a given question, compared with 1 out of 134 candidates from the rest of the candidate generation pipeline. Details on how PRISMATIC is used to generate candidate answers can be found in [2].

Missing link

PRISMATIC can also be used to provide the semantic relations between entities. Some Jeopardy! questions require players to find a hidden entity whose connections to parts of the clue are not explicitly stated. For example, given the question "On hearing of the discovery of George Mallory's body, he told reporters he still thinks he was first", the entity *Mount Everest* is not stated, but it provides the common link between *George Mallory* and the correct answer *Edmund Hillary*. PRISMATIC can be used to find such missing links and estimate the degree of closeness between two concepts. For the above example, one may find the following frames: $\{\langle subj,$"*George Mallory*"$\rangle\ \langle verb,$"*perish*"$\rangle\langle verb\ prep,$"*on*"$\rangle\ \langle objprep,$"*Mount Everest*"$\rangle\}$ and $\{\langle subj,$"*Edmund Hillary*"$\rangle\ \langle verb,$"*climb*"$\rangle\ \langle obj,$"*Mount Everest*"$\rangle\}$, which illustrates the connection between *Mallory* and *Hillary* through *Mount Everest*. Details on how PRISMATIC is used to find missing links in Jeopardy! questions can be found in [4].

Type inference and its related uses

As noted in the section on frame projection, we use frame projections to dissect frames along different slot dimensions. The projections are used to produce aggregate statistics across the entire data set to induce relationships among the various frame slots, e.g., learn the predominant types for subject/object slots in verb and noun phrases. Given a new piece of text, we can apply this knowledge to infer types for named entities. For example, since the aggregate statistics show the most common type for the object of the verb *annex* is Region, we can infer from the sentence "Napoleon annexed Piedmont in 1859" that *Piedmont* is most likely to be a Region. Similarly, consider the sentence "He ordered a Napoleon at the restaurant". A dictionary-based NER is very likely to label *Napoleon* as a Person. However, we can learn from a large amount of data that in the frame $\{\langle subject\ type,$"*Person*"$\rangle\langle verb,$"*order*"$\rangle\langle object\ type,$"?"$\rangle\ \langle verb\ prep,$"*at*"$\rangle\langle obj\ prep,$"*restaurant*"$\rangle\}$, the object type typically denotes a Dish and, thus, correctly infer the type for *Napoleon* in this context. Learning this kind of fine-grained type information for a particular context is not possible using traditional handcrafted resources such as VerbNet or FrameNet. Unlike previous work in selectional restriction [8, 9], PRISMATIC-based type inference does not depend on a particular taxonomy or previously annotated training data; it works with any NER and its type system.

One of the areas in DeepQA where the above type inference can be used is during question analysis. In [10], we describe a set of heuristics to detect the LAT from the question text. In some cases, the LAT in the question may be meaningless (e.g., *this* or *it*), as in the following example: "In the billiards game named for this black object, you must sink it last". In such cases, we can infer a meaningful LAT by looking at lexical relations involving the focus of the question and using PRISMATIC to find the most frequent argument types for the relation, which are in the focus position. In the above example, since the focus *it* is the object of the verb *sink*, we can issue the frame query $\{\langle verb,$"*sink*"$\rangle\langle object,$"?"$\rangle\}$ against the PRISMATIC knowledge base to find the most frequent lexical type values for the object slot and use these values as potential LATs. However, a key issue is taking into account the larger context (or topic) of the relation, without which the system can generate incorrect or noisy results. For example, given the aforementioned query, PRISMATIC finds the

most frequent object of the verb *sink* to be a *ship* (whereas the correct LAT expected is *ball*). The problem lies in the lack of context-aware knowledge in PRISMATIC; although the question mentions the keyword *billiards*, there is no direct dependency parse link between *billiards* and the focus, and hence, it is not used as part of the frame query.

As a workaround, we utilize type inference knowledge from PRISMATIC in an answer-scoring component. The idea is that because candidate answers are typically generated on the basis of the context of the question, checking whether a given candidate fits into the frame of the question using PRISMATIC is an easier task than predicting the correct LAT for the question, which requires considering context within PRISMATIC. Continuing with our example question, it is highly unlikely that an instance of a ship would be generated as a candidate answer given the context of the question, whereas entities related to billiards such as *table*, *black ball*, and *cue stick* are more likely candidate answers. Thus, in this case, we can use PRISMATIC knowledge to compare candidates on the basis of how well they fit into the object slot for the verb *sink* (e.g., *sink ball* is a more common frame occurrence than *sink table*, *sink cue stick*, and *sink pocket*). Results for an answer scorer that does PRISMATIC-based type inference along these lines are in [3].

The automatically induced type information can also be used for co-reference resolution. For example, given the sentence "Netherlands was ruled by the UTP party before Napoleon annexed it", we can use the inferred type constraint on *it* (Region) to resolve it to *Netherlands* (instead of the *UTP Party*).

Finally, typing knowledge can be used for word-sense disambiguation. From the sentence "Tom Cruise is one of the biggest stars in American Cinema", we can infer, using our frame-induced type knowledge base, that the word *stars* in this context refers to a Person/Actor type and not in the sense of *star* as an astronomical object.

Related work

Manually created resources

Many knowledge bases have been manually built since the beginning of artificial intelligence research. Among them, CYC** [11] is the best known. More recently, other large-scale knowledge bases such as DBpedia [12] and YAGO [13] that utilize crowd source results from the web to create large amounts of knowledge have become popular, and they have been shown to be useful for different applications.

Several lexical resources have been manually built, most notably WordNet** [14], FrameNet [15], and VerbNet [16]. WordNet is a lexical resource that contains individual WordNet word synset information, such as definition, synonyms, and antonyms. However, the amount of predicate knowledge in WordNet is limited.

FrameNet is a lexical database that describes the frame structure of selected words. Each frame represents a predicate (e.g., eat and remove) with a list of frame elements that constitute the semantic arguments of the predicate. Different words may map to the same frame, and one word may map to multiple frames on the basis of different word senses. Frame elements are often specific to a particular frame, and even two frame elements with the same name, such as "Agent", may have subtle semantic meanings in different frames.

VerbNet is a lexical database that maps verbs to their corresponding Levin [17] classes, and it includes syntactic and semantic information of the verbs, such as the syntactic sequences of a frame (e.g., *NP V NP PP*) and the selectional restriction of a frame argument value.

Compared with these resources, in addition to being an automatic process, PRISMATIC has three major differences. First, unlike the descriptive knowledge in WordNet, VerbNet, and FrameNet, PRISMATIC offers only numeric knowledge of the aggregate statistics of different predicates and their argument values throughout a corpus. The statistical profiles are easy to automatically produce, and they allow additional knowledge, such as type restriction, to be inferred from PRISMATIC easily.

Second, the frames are differently defined. The frames in PRISMATIC are not abstract concepts generalized over a set of words. They are defined by the words in a sentence and the relations between them. Two frames with different slot values are considered different although they may be semantically similar. For example, the two sentences "John loves Mary" and "John adores Mary" result in two different frames, although semantically, they are very close. By choosing not to use frame concepts generalized over words, we avoid the problem of determining which frame a word belongs to when processing text automatically. Importantly, if there is enough redundancy in a corpus, then valid values for different synonyms and variations will be produced.

Third, PRISMATIC uses only a very small set of slots (see Table 1) defined by parser and relation annotators to link a frame and its arguments. By using parser relations as slots directly, we avoid the problem of mapping parser relations to frame elements.

Automatically created resources

The notion of utilizing implicit knowledge from text can be traced to KNEX [18]. Schubert proposed to derive general world knowledge in the form of possible propositions from text. He applied a set of predefined syntactic patterns to parse trees to extract propositions, such as "behavior can be strange" or "a female individual may have an arm". DART (Discovery and Aggregation of Relations in Text)

[19] successfully applied such propositions to a variety of tasks, such as improving parsing and providing support for textual entailment rules.

TextRunner [7] is an information extraction system that automatically extracts relation tuples over massive web data in an unsupervised manner. TextRunner contains more than 800 million extractions [7] and has proven to be a useful resource in a number of important tasks in machine reading such as hypernym discovery [20] and scoring interesting assertions. TextRunner works by automatically identifying and extracting relationships using a conditional random field (CRF) model over natural-language text. Because this is a computationally inexpensive technique, it allows rapid application to web-scale data.

DIRT (Discovering Inference Rules from Text) [21] automatically identifies inference rules over dependency paths that tend to link the same arguments. The technique consists of applying a dependency parser over 1 GB of text, collecting the paths between arguments, and then calculating path similarities between paths. DIRT has been extensively used in recognizing textual entailment (RTE).

Never-Ending Language Learner (NELL) [22] is another system that acquires knowledge automatically. It starts with a set of seed categories and relations with a handful of examples, and it extracts new instances of categories and relations from millions of web pages.

Another recent system, Background Knowledge Base (BKB) [23], also extracts parts of the parsed structure from a domain-specific corpus as a knowledge source to improve machine reading systems.

PRISMATIC is similar to KNEX and DART in the way knowledge is extracted. All of them use parse trees as input, although PRISMATIC includes semantic relations, such as *isa*, in addition to parse relations. The frames and frame projections in PRISMATIC are similar to the syntactic patterns in KNEX and DART. The main difference between KNEX/DART and PRISMATIC is the kind of knowledge produced. KNEX and DART are interested in terminological axioms by generalizing named entities to their types. This allows them to use a relatively small corpus. PRISMATIC produces both assertional and terminological axioms. In order to obtain impactful coverage for the assertional axioms, PRISMATIC requires a large corpus.

PRISMATIC is similar to TextRunner and DIRT in that it may be automatically applied over massive corpora. At a representational level, it differs from both TextRunner and DIRT by storing full frames from which *n*-ary relations may be indexed and queried. PRISMATIC differs from TextRunner in that it applies a full dependency parser in order to identify dependency relationships between terms. In contrast to DIRT and TextRunner, PRISMATIC also performs co-reference resolution in order to increase coverage for sparsely occurring entities and employs a named entity detector (NED) and relation extractor on all of its extractions to better represent intentional information. PRISMATIC is also different from TextRunner in terms of the annotation efforts required. Because TextRunner needs to train a CRF model, it requires supervised training data. PRISMATIC,
on the other hand, does not require any training data.

PRISMATIC is similar to NELL in the sense that it also extracts instances of categories and relations automatically from a large corpus. However, the categories and relations are open-ended for PRISMATIC; they are not limited to the set of given examples for NELL.

PRISMATIC is similar to BKB because both use parser outputs to extract knowledge. They differ in terms of the corpora they use and their applications. BKB uses a domain-specific corpus to focus on aggregate statistics relevant to that domain to improve machine reading system performance. PRISMATIC uses an open-domain general-purpose corpus to obtain overall aggregate statistics of text usage.

Discussion and future work

We are improving PRISMATIC in three major areas: context, coverage, and confidence. First, the current version of PRISMATIC does not take context into consideration when producing the aggregating statistics. Because of the different usage of words under different contexts, the aggregate statistics often are a mixture of different contexts. As illustrated in the section "Applications in Watson and beyond," the most likely object of the verb *sink* is *ship*, and *ball* is among the top objects of *sink*. Clearly, this is the result of two senses of *sink*. If future versions of PRISMATIC include context information, it will give us more precise aggregate statistics for different situations. The second area of improvement is coverage. Although we have used a large corpus as input for PRISMATIC, there are still gaps in its coverage. There are different ways to improve this. We may be able to use better co-reference components; we may incorporate even larger corpora. Finally, we are looking to improve confidence. Because none of the NLP components is perfect, future versions of PRISMATIC may include confidence values in its frames that incorporate the scores of each of the NLP components' outputs. The confidence score may help create better aggregate statistics than the ones based only on the frequency count.

In this paper, we presented PRISMATIC, a large-scale lexicalized relation resource that is automatically built over massive amounts of text. It provides users with knowledge about predicates and their arguments. It will be useful for a variety of artificial intelligence applications, some of which have been already demonstrated as part of the IBM Watson system. We plan to pursue the others in the near future.

*Trademark, service mark, or registered trademark of International Business Machines Corporation in the United States, other countries, or both.

**Trademark, service mark, or registered trademark of Jeopardy Productions, Inc., Cycorp, Inc., or Trustees of Princeton University in the United States, other countries, or both.

References

1. D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, N. Schlaefer, and C. Welty, "Building Watson: An overview of the DeepQA project," *AI Mag.*, vol. 31, no. 3, pp. 59–79, 2010.
2. J. Chu-Carroll, J. Fan, B. K. Boguraev, D. Carmel, D. Sheinwald, and C. Welty, "Finding needles in the haystack: Search and candidate generation," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 6, pp. 6:1–6:12, May/Jul. 2012.
3. J. W. Murdock, A. Kalyanpur, C. Welty, J. Fan, D. A. Ferrucci, D. C. Gondek, L. Zhang, and H. Kanayama, "Typing candidate answers using type coercion," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 7, pp. 7:1–7:13, May/Jul. 2012.
4. J. Chu-Carroll, E. W. Brown, A. Lally, and J. W. Murdock, "Identifying implicit relationships," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 12, pp. 12:1–12:10, May/Jul. 2012.
5. M. C. McCord, J. W. Murdock, and B. K. Boguraev, "Deep parsing in Watson," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 3, pp. 3:1–3:15, May/Jul. 2012.
6. C. Wang, A. Kalyanpur, J. Fan, B. K. Boguraev, and D. C. Gondek, "Relation extraction and scoring in DeepQA," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 9, pp. 9:1–9:12, May/Jul. 2012.
7. T. Lin, O. Etzioni, and J. Fogarty, "Identifying interesting assertions from the web," in *Proc. 18th CIKM*, 2009, pp. 1787–1790.
8. J. Carroll and D. McCarthy, "Word sense disambiguation using automatically acquired verbal preferences," *Comput. Human.*, vol. 34, no. 1/2, pp. 109–114, Apr. 2000.
9. P. Resnik, "Selection and information: A class-based approach to lexical relationships," Ph.D. dissertation, Univ. Pennsylvania, Philadelphia, PA, 1993.
10. A. Lally, J. M. Prager, M. C. McCord, B. K. Boguraev, S. Patwardhan, J. Fan, P. Fodor, and J. Chu-Carroll, "Question analysis: How Watson reads a clue," *IBM J. Res. & Dev.*, vol. 56, no. 3/4, Paper 2, pp. 2:1–2:14, May/Jul. 2012.
11. D. B. Lenat, "CyC: A large-scale investment in knowledge infrastructure," *Commun. ACM*, vol. 38, no. 11, pp. 33–38, Nov. 1995.
12. C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann, "Dbpedia—A crystallization point for the web of data," *J. Web Semant.—Science, Services and Agents on the World Wide Web*, vol. 7, no. 3, pp. 154–165, Sep. 2009.
13. F. Suchanek, G. Kasneci, and G. Weikum, "Yago—A core of semantic knowledge," in *Proc. 16th Int. WWW*, 2007, pp. 697–706.
14. C. Fellbaum, *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press, 1998.
15. C. F. Baker, C. J. Fillmore, and J. B. Lowe, "The Berkeley framenet project," in *Proc. 36th Annu. Meeting ACL*, 1998, vol. 1, pp. 86–90.
16. K. Schuler, "VerbNet: A broad-coverage, comprehensive verb lexicon," Ph.D. dissertation, Univ. Pennsylvania, Philadelphia, PA, 2005.
17. B. Levin, *English Verb Classes and Alternations: A Preliminary Investigation*. Chicago, IL: Univ. Chicago Press, 1993.
18. L. Schubert, "Can we derive general world knowledge from text?" in *Proc. HLT*, 2002, pp. 94–97.
19. P. Clark and P. Harrison, "Large-scale extraction and use of knowledge from text," in *Proc. 5th KCAP*, 2009, pp. 153–160.
20. S. Soderland, A. Ritter, and O. Etzioni, "What is this, anyway: Automatic hypernym discovery," in *Proc. AAAI Spring Symp. Learn. Read. Learn. Read*, 2009, pp. 88–93.
21. D. Lin and P. Pantel, "Dirt—Discovery of inference rules from text," in *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2001, pp. 323–328.
22. A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, Jr., and T. M. Mitchell, "Toward an architecture for never-ending language learning," in *Proc. AAAI*, 2010, pp. 1306–1313.
23. A. Penas and E. Hovy, "Semantic enrichment of text with background knowledge," in *Proc. 1st Int. Workshop FAM-LbR NAACL*, 2010, pp. 15–23.

Received July 22, 2011; accepted for publication December 7, 2011

James Fan *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (fanj@us.ibm.com).*

Dr. Fan is a Research Staff Member in the Semantic Analysis and Integration Department at the T. J. Watson Research Center. He joined IBM after receiving the Ph.D. degree at the University of Texas at Austin, in 2006. He is a member of the DeepQA Team that developed the Watson question-answering system, which defeated the two best human players on the quiz show *Jeopardy!*. Dr. Fan is author or coauthor of dozens of technical papers on subjects of knowledge representation, reasoning, natural-language processing, and machine learning. He is a member of the Association for Computational Linguistics.

Aditya Kalyanpur *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (adityakal@us.ibm.com).*

Dr. Kalyanpur is a Research Staff Member at the IBM T. J. Watson Research Center. He received his Ph.D. degree in computer science from the University of Maryland in 2006. His research interests include knowledge representation and reasoning, natural-language processing, statistical data mining, and machine learning. He joined IBM in 2006 and worked on the Scalable Highly Expressive Reasoner (SHER) project that scales ontology reasoning to very large and expressive knowledge bases. Subsequently, he joined the algorithms team on the DeepQA project and helped design the Watson question-answering system. Dr. Kalyanpur has over 25 publications in leading artificial intelligence journals and conferences and several patents related to SHER and DeepQA. He has also chaired international workshops and served on W3C Working Groups.

David C. Gondek *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (dgondek@us.ibm.com).*

Dr. Gondek is a Research Staff Member and Manager at the T. J. Watson Research Center. He received the B.A. degree in mathematics and computer science from Dartmouth College, Hanover, NH, in 1998 and the Ph.D. degree in computer science from Brown University, Providence, RI, in 2005. He subsequently joined IBM, where he worked on the IBM Watson Jeopardy! challenge and now leads the Knowledge Capture and Learning Group in the Semantic Analysis and Integration Department.

David A. Ferrucci *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (ferrucci@us.ibm.com).*

Dr. Ferrucci is an IBM Fellow and the Principal Investigator for the DeepQA Watson/Jeopardy! project. He has been at the T. J. Watson Research Center since 1995, where he leads the Semantic Analysis and Integration Department. Dr. Ferrucci focuses on technologies for automatically discovering meaning in natural-language content and using it to enable better human decision making. He graduated from Manhattan College with the B.S. degree in biology and from Rensselaer Polytechnic Institute, Troy, NY, in 1994 with the Ph.D. degree in computer science specializing in knowledge representation and reasoning. He has published papers in the areas of artificial intelligence, knowledge representation and reasoning, natural-language processing, and automatic question answering.