YAGO-QA: Answering Questions by Structured Knowledge Queries

Peter Adolphs*, Martin Theobald‡, Ulrich Schäfer†, Hans Uszkoreit† and Gerhard Weikum‡
*†German Research Center for Artificial Intelligence (DFKI), Language Technology Lab
*Alt Moabit 91c, D-10559 Berlin, Germany / †Campus D3.1, D-66123 Saarbrücken, Germany
‡Max-Planck-Institut für Informatik (MPI-INF), Campus E1.4, D-66123 Saarbrücken, Germany
Emails: {peter.adolphs,ulrich.schaefer,hans.uszkoreit}@dfki.de, {mtb,weikum}@mpi-inf.mpg.de

***Abstract*—We present a natural-language question-answering system that gives access to the accumulated knowledge of one of the largest community projects on the Web – Wikipedia – via an automatically acquired structured knowledge base. Key to building such a system is to establish mappings from natural language expressions to semantic representations. We propose to acquire these mappings by data-driven methods – corpus harvesting and paraphrasing – and present a preliminary empirical study that demonstrates the viability of our method.**

I. Introduction

Motivation. Natural-language question answering, QA for short, has a long history in NLP and IR research. The Web has proved to be a valuable resource for answering fact-oriented questions. State-of-the-art methods [1], [2], [3] cast the user's question into a keyword query to a Web search engine (perhaps with phrases for location and person names or other proper nouns). Key to finding good results is to retrieve and properly rank sentences or short text passages that contain all or most keywords and are likely to contain good answers. Together with trained classifiers for the question type (and thus the desired answer type), this methodology performs fairly well for both factoid and list questions. Unfortunately, approaches along these lines cannot handle complex questions that involve relationships between entities and can be answered only by composing information pieces from different text passages.

IBM's Watson project [4] impressively demonstrated a new level of *deep QA*, by winning the popular quiz show Jeopardy against two human champions. A key element in Watson's approach is to decompose complex questions into several cues and sub-cues, with the aim of generating answers from matches for the various cues (tapping into the Web and Wikipedia but also using knowledge bases like DBpedia [5], Freebase [6], and YAGO [7]). Notwithstanding its success, the Watson QA system failed to correctly answer a substantial fraction of the questions in the competition. One of these was: "A big US city with two airports, one named after a World War II hero, one named after a World War II battle field." Watson answered "Toronto"; the correct answer would have been Chicago. Interestingly, it would seem very easy to answer this question by executing a structured query over available knowledge bases or the existing and rapidly increasing set of *Linked Data* sources [8]. All these sources are represented in the RDF format, and can be queried in the SPARQL language. The above question can be expressed as follows:

```
Select ?c Where { ?c hasType City .
?a1 hasType Airport . ?a2 hasType Airport .
?a1 locatedIn ?c . ?a2 locatedIn ?c .
?a1 namedAfter ?p . ?p hasType WarHero .
?a2 namedAfter ?b . ?b hasType BattleField . }
```

where each search condition terminated by a dot is a *triple pattern* with three components – subject, predicate, object – some of which can be variables (starting with a question mark). When the same variable appears in multiple triple patterns, the semantics is that of a relational join: results are bindings for the variable that simultaneously satisfy all predicates.

Contribution. SPARQL-like languages provide expressive access to knowledge bases and linked-data sources. However, only programmers would be able and willing to use such formal interfaces, whereas most users would prefer natural-language QA. This paper addresses this very mismatch, by proposing methods for mapping natural-language questions into structured SPARQL queries. This is a major departure from traditional QA, based on keyword queries. Prior work related to our approach includes [9], [10], [11]. Although these projects pursued aims similar to ours, their techniques were quite different and did not lead to robust results.

Our approach is based on decomposing questions into smaller units, each of which is then mapped to a SPARQL triple pattern. For this mapping, we utilize knowledge-harvesting methods for gathering surface-text *patterns* that are likely to express certain relations, and we have employed a crowdsourcing approach for gathering *paraphrases* for formulating questions. For example, for the namedAfter relation, we have harvested Web and news sources for textual patterns like "was named after" or "got its name from". This meta-knowledge is the key to matching verbal phrases in the question against the available relations in the knowledge base. Similarly, the semantic classes in the knowledge (e.g., city) are associated with synonyms and paraphrases (e.g., "agglomoration" or "urban area"), and for named entities (e.g., O'Hare International Airport) the knowledge base already provides a rich repository of surface forms (e.g., "O'Hare airport", "ORD", "Chicago international airport").

In summary, our contribution in this paper is a new approach to QA over knowledge bases, with new techniques for mapping questions into structured queries.

II. System Overview

The YAGO-QA prototype consists of three major components. First, the YAGO knowledge base provides the contents for answering questions in the form of RDF triples extracted from Wikipedia. The *YAGO server* provides a SPARQL query interface to this fact collection, encapsulated into a Web Service. Second, we use a *repository of text patterns and question paraphrases* compiled by two complementary approaches: automatically processing a very large Web corpus (ClueWeb, used in TREC) to gather patterns that may denote relations in YAGO, and collecting paraphrases for questions by crowdsourcing to humans. Third, the actual *question processing* decomposes natural-language questions and maps the constituents onto SPARQL triple patterns.

The following sections explain the system components of YAGO-QA in more detail.

III. Gathering Paraphrases by Crowdsourcing

A key problem for QA on structured data is how to efficiently arrive at mappings from natural language expressions to semantic representations close to the knowledge base. These mappings should not only represent the meanings conveyed by natural-language expressions faithfully but should also cover all relevant expressions used in the domain. Community wisdom says that high coverage can hardly be achieved with purely introspective methods, but should rather be data-driven. However, a QA corpus with sufficient coverage and variance on the domain expressions required for our data does not yet exist. And even if such a corpus existed, we would also need meaningful annotations or at least semantic clusters that group similar questions.

In order to discover all different means that users use to express a certain fact or question, we employ crowd-sourcing methods for creating an initial pattern base. To this end, we built a Web-based platform for acquiring question paraphrases from multiple human annotators. We manually compiled a list of questions centered around biographical information of people that can be answered with our data. Questions could be simple, i.e., involve just one triple pattern, such as "How old is Brad Pitt?", or more complex, involving several triple patterns, such as "Which French singer won a Grammy award?" These questions are used as seeds for the annotator's job. Annotators were asked to form paraphrases of the seed questions, using whatever means they could think of. They were explicitly encouraged to be creative and to provide short paraphrases (e.g., "Chicago-born actor") or paraphrases with unusual expressions (e.g., "Which actor saw the light of the day in Chicago?") alongside all natural paraphrases ("Which actor was born in Chicago?") to avoid too much priming by the seed questions' structure and wording. The main requirement for a paraphrase to be valid is that the same answer as for the seed question should also be acceptable for the paraphrase. This of course implies that the arguments of the triple patterns must be the same for all paraphrases.

IV. YAGO Server

The YAGO project [7] and the parallel work on DBpedia [5] and WikiTaxonomy [12] have shown how to massively extract facts from semistructured elements of Wikipedia (infoboxes, categories, lists), to build large knowledge bases. YAGO has pursued the philosophy of high – near-human-quality – precision by employing database-style consistency checking on fact candidates. YAGO primarily gathers its knowledge by rule-based information extraction from categories and infoboxes in Wikipedia, and integrates the resulting facts with the taxonomical class system of WordNet [7]. Consistency checks include type constraints (e.g., $isMarriedTo$ has type signature $Human \times Human$, $graduatedFrom$ has type $Human \times University$) and functional dependencies (e.g., for relation $isCapitalOf$, $City \rightarrow Country$ is a function). YAGO can be publicly downloaded (www.mpi-inf.mpg.de/yago-naga/), and is part of the Linked Data initiative [8].

Recently, we have completed a major revision of the YAGO architecture for automatic extraction, and built a new edition of the knowledge base [13]. We integrated *spatial* and *temporal* facts to a much larger extent, including information from Geonames. Moreover, we harvested *multilingual* information from more than a hundred Wikipedia editions to capture entity and concept names as well as taxonomic relations in many languages [14]. YAGO currently contains more than 80 million facts, including meta-facts about time, location, and provenance. Moreover, it contains a huge number of surface names, short names, and even paraphrases for named entities. For example, YAGO knows that "JFK" is an abbreviation for the John F. Kennedy International Airport and that "Big Apple" is a nickname for New York City.

The facts in knowledge bases like YAGO also serve as seeds for *pattern-based information extraction* from arbitrary natural-language texts and Web sources. Our tools SOFIE and PROSPERA combine statistical evidence from pattern occurrences with logical reasoning over consistency constraints [15]. PROSPERA uses MapReduce-style parallelism to scale up these computationally expensive procedures on distributed platforms. Its typical use case is massive-scale harvesting of new facts about known entities. In this process, it keeps the patterns that lead to fact extractions, along with their statistical confidence. This way, it can provide for each relation, a *ranked list of surface phrases* that are likely to denote that relation. We utilize this in our QA system, for mapping verbal phrases in the question onto relations in the knowledge base.

For querying the knowledge base, YAGO offers both an API for programmed access and a UI for interactive exploration. Our search engine NAGA evaluates SPARQL predicates on the underlying RDF data. We also support keyword search over text-augmented RDF triples and flexible combinations of structured and keyword conditions [16], [17]. The API used for YAGO-QA provides two Web Service functions which can be accessed via SOAP to query the knowledge base:

- `getYagoEntitiesByNames(String[] names)`
  Returns a list of mappings from names onto their most

likely meanings (i.e., their YAGO entity ids and respective frequencies), each by returning the most frequent target entity among all link anchors that match the queried name.

- `processQuery(String query)`
  Triggers the processing of a SPARQL query against the NAGA search engine. It returns a ranked list of RDF tuples. Each tuple contains a set of RDF facts that matches the SPARQL query pattern.

V. Question Processing

The QA on-line system realizes a classical pipeline architecture (see Fig. 1).

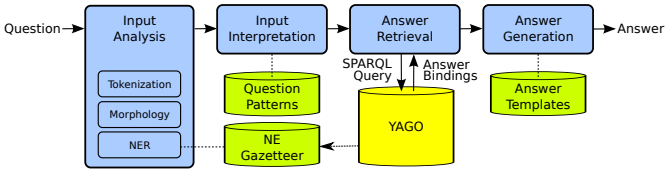

Fig. 1. Architecture

The user's input is first linguistically analyzed. We use the SProUT system [18] for tokenization, morphological analysis and named-entity recognition (NER). SProUT recognizes named entities by two different means: i) recognition of unknown names using name grammars and ii) lookup in 'gazetteers', i.e., large lists of known names. Whereas the former strategy is suitable for named-entity classes which follow regular patterns such as person names with at least some standard name elements (like regular given names, typical last names, etc.) or time and date expressions, the latter strategy is useful if the names do not follow a pattern and are already known for a domain. As the names of the entities in YAGO are already known, we create a gazetteer associating all names in the knowledge base with their NE types and identifiers.

Multiple, overlapping analyses for the same span are allowed as results of the input analysis. Analyses do not have to be disambiguated, i.e., they may provide alternative interpretations for the same linguistic expression. Rather than disambiguating analyses early, we rely on the input interpretation phase to choose suitable local interpretations of parts of the input in order to find the interpretation of the full input sequence. This is especially useful for movie, book and song titles, as they may span multiple words and form general phrases, that can hardly be distinguished without considering the full context.

Input interpretation is performed by matching the analyzed input against a set of interpretation patterns, which are derived from the collected paraphrases for given semantic relations. At this early stage of system development, we use surface patterns that provide morphosyntactic abstractions for tokens. These are case-insensitive and allow for some flexibility w.r.t. the presence or absence of functional categories such as determiners. They contain NE placeholders that are filled if the semantic types of the NE analyses match. As there might be several referents of the same semantic type for a given

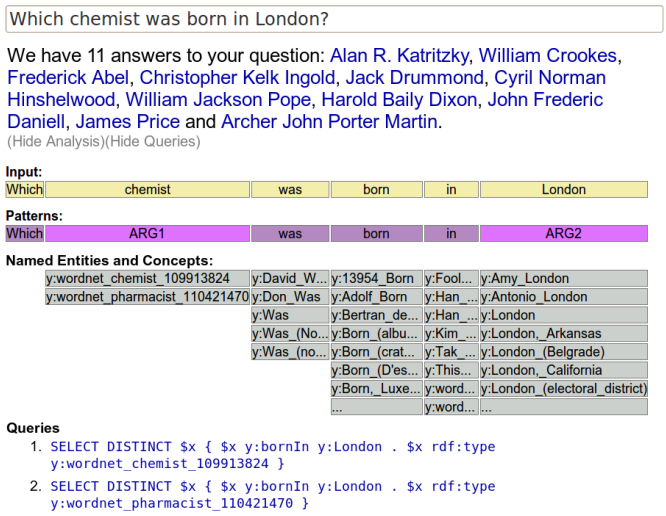

Fig. 2. Screenshot of the YAGO-QA system: user question and system answer, as well as linguistic analysis details and SPARQL queries

name, we use Wikipedia inlink counts as an approximation of the referent's salience. For instance, there are 27 referents for the string "Berlin" in the question "Which entertainer died in Berlin?". Even after narrowing down the semantic type to 'Location' through the interpretation pattern, there are still 17 potential referents left. Using the referent with the highest inlink count allows us to pick up the referent with the highest prominence, which is presumably the desired interpretation.

Interpretation patterns are associated with SPARQL query templates. Query templates are instantiated by filling all NE slots with the highest ranked referent and then issued to the knowledge base. If several patterns with different associated query templates match the input, all queries (of course, with the same referents in the NE slots) will be used for answer retrieval, and the results will finally be merged. The query results are turned to the generator module, which wraps them into a natural-language answer using pre-factured templates.

VI. Preliminary Results

The crowd-sourcing approach to pattern collection forms an efficient way to acquire relevant and semantically clustered expressions for a domain. The task is easy to understand, both for the person to create the initial seed list as well as for the annotators. At the time of the writing, we acquired 4,620 paraphrases for 254 seed questions with 7 annotators. Total annotation time sums up to approximately 49 hours, i.e., roughly one working day per annotator.

The YAGO-QA prototype is implemented in Java, following a service-oriented architecture. Access to YAGO and the linguistic analysis module as well as the actual QA processing module are realized as Web services. A web interface allows users to ask questions, see the system's answers and inspect the analyses, interpretations and queries which led to the results.

Fig. 2 shows a user's question about all London-born chemists. As can be seen, almost every part of the case-normalized input is interpretable as a reference to some entity

if no context is taken into consideration. For instance, "born" and "london" can refer to people and locations alike. The final interpretation, where "born" is seen as a verb form of "(to) bear" and "london" as a location named-entity, is found through the fuzzy match of the input to a pattern "which ARG1 was born in ARG2". This pattern provides two slots, namely an expression for classes as its first argument (ARG1) and a location name as its second argument (ARG2). Considering all semantically compatible referents for ARG1 and ARG2, we arrive at 18 different SPARQL queries, which are ranked based on the prominence of their referents, as described before.

The previous example demonstrated a query involving two simple constraints on the queried entities. The system also answers more complex questions where the queried entities are not directly related to the entities given in the question but are linked to them through a chain of relations. For instance, rather than asking "Who invented x-bar theory?" (A: "Noam Chomsky") and then using the result of that question in a follow-up question "Where does Noam Chomsky work?" ("MIT"), we can directly find the result with the question: "Where does the person work who invented x-bar theory?" Our data-driven methods ensure that variants of these questions (e.g., "Where does the scholar behind x-bar work?") are mapped to the same SPARQL queries.

VII. Conclusion

The arrival of huge structured knowledge repositories has opened up opportunities for answering complex questions, involving multiple relations between queried entities, by operating directly on the semantic representations rather than finding answers in natural-language texts. Query languages for accessing these repositories are well established; however, they are too complicated for non-technical users, who would prefer to pose their questions in natural language. We have presented the YAGO-QA system that gives access to the accumulated knowledge of Wikipedia via the automatically acquired structured knowledge base YAGO. The main novelty of YAGO-QA lies in its mapping from natural language expressions in user questions to semantic representations (relations, classes, entities) in the knowledge base. We have shown how to harness surface-text patterns from a large-scale knowledge-harvesting machinery, and complemented this with a repository of question paraphrases gathered by a crowdsourcing approach.

Our ongoing and future work addresses the following issues: 1) As structured knowledge bases cover only a subset of the information that users are interested in, our approach has to be complemented with "open QA" techniques that tap into textual content sources as well. 2) Fuzzily mapping user questions against a set of paraphrases is not sufficient for dealing with the productive use of language. We plan to learn constructions from the acquired paraphrases, utilizing limited grammatical knowledge, and allow the system to combine them freely during input interpretation in order to decompose meaning of unforeseen input. 3) We plan to conduct a systematic evaluation study of YAGO-QA with real user questions.

Acknowledgments

The authors thank the DFG Cluster of Excellence on Multimodal Computing and Interaction (M2CI); project TAKE, funded under contract 01IW08003 by the German Federal Ministry of Education and Research; project KomParse, funded under contract 1014 0149 by the ProFIT program of the Federal State of Berlin and the EFRE program of the European Union; and projects Theseus Alexandria and Alexandria for Media, funded under contract 01 MQ 07 016 by the German Federal Ministry of Economy and Technology.

References

[1] L. Hirschman and R. Gaizauskas, "Natural language question answering: the view from here," *Natural Language Engineering*, vol. 7, no. 4, pp. 275–300, 2001.

[2] C. Kwok, O. Etzioni, and D. S. Weld, "Scaling question answering to the web," in *Proc. of WWW 2001*, Hong Kong, 2001.

[3] Z. Zheng, "AnswerBus question answering system," in *Human Language Technology Conference*, San Diego, CA, 2002.

[4] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. 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 Magazine*, vol. 31, no. 3, pp. 59–79, 2010.

[5] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, and Z. Ives, "DBpedia: A Nucleus for a Web of Open Data," in *In 6th International Semantic Web Conference, Busan, Korea*. Springer, 2007, pp. 11–15.

[6] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *ACM SIGMOD International Conference on Management of Data, Vancouver, Canada*, 2008, pp. 1247–1250.

[7] F. Suchanek, G. Kasneci, and G. Weikum, "YAGO: A Core of Semantic Knowledge - Unifying WordNet and Wikipedia," in *Proc. of WWW 2007*, Banff, Canada, 2007, pp. 697–706.

[8] T. Heath and C. Bizer, Eds., *Linked Data*. Morgan & Claypool Publishers, 2011.

[9] A. Copestake and K. Sparck Jones, "Natural language interfaces to databases," *Knowledge Engineering*, vol. 5, no. 4, pp. 225–249, 1990.

[10] I. Androutsopoulos, G. D. Ritchie, and P. Thanisch, "Natural language interfaces to databases—an introduction," *Journal of Natural Language Engineering*, vol. 1, no. 1, pp. 29–81, 1995.

[11] A. Frank, H.-U. Krieger, F. Xu, H. Uszkoreit, B. Crysmann, and U. Schäfer, "Question answering from structured knowledge sources," *Journal of Applied Logics, Special Issue on Questions and Answers: Theoretical and Applied Perspectives*, vol. 5, no. 1, pp. 20–48, 2007.

[12] S. P. Ponzetto and M. Strube, "Deriving a Large-Scale Taxonomy from Wikipedia," in *Proc. AAAI'07*, 2007, pp. 1440–1445.

[13] J. Hoffart, F. M. Suchanek, K. Berberich, E. Lewis-Kelham, G. de Melo, and G. Weikum, "YAGO2: Exploring and Querying World Knowledge in Time, Space, Context, and Many Languages," in *Proc. of WWW 2011*, Hyderabad, India, 2011.

[14] G. de Melo and G. Weikum, "MENTA: Inducing Multilingual Taxonomies from Wikipedia," in *Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM 2010)*. Toronto, Canada: ACM, 2010, pp. 1099–1108.

[15] N. Nakashole, M. Theobald, and G. Weikum, "Scalable Knowledge Harvesting with High Precision and High Recall," in *4th ACM International Conference on Web Search and Data Mining (WSDM)*. Hong Kong: ACM, 2011, pp. 227–236.

[16] S. Elbassuoni, M. Ramanath, R. Schenkel, M. Sydow, and G. Weikum, "Language-Model-Based Ranking for Queries on RDF-Graphs," in *CIKM 2009 : the 18th ACM Conference on Information and Knowledge Management*. Hongkong, China: ACM, 2009, pp. 977–986.

[17] S. Elbassuoni, M. Ramanath, and G. Weikum, "Query relaxation for entity-relationship search," in *ESWC'11: Proceedings of the 8th Extended Semantic Web Conference*, Heraklion, Greece, 2011.

[18] W. Drożdżyński, H.-U. Krieger, J. Piskorski, U. Schäfer, and F. Xu, "Shallow processing with unification and typed feature structures – foundations and applications," *Künstliche Intelligenz*, pp. 17–23, 2004.