Post-print of: Boiński T., Szymański J., Dudek B., Zalewski P., Dompke S., Czarnecka M. (2018) DBpedia and YAGO Based System for Answering Questions in Natural Language. In: Nguyen N., Pimenidis E., Khan Z., Trawiński B. (eds) Computational Collective Intelligence. ICCCI 2018. Lecture Notes in Computer Science, vol 11055. Springer, Cham. DOI: 10.1007/978-3-319-98443-8_35

DBpedia and YAGO Based System for Answering Questions in Natural Language

Tomasz Boiński, Julian Szymański, Bartłomiej Dudek,
Paweł Zalewski, Szymon Dompke, and Maria Czarnecka

Department of Computer Architecture
Faculty of Electronics, Telecommunication and Informatics
Gdańsk University of Technology
tobo@eti.pg.edu.pl
julian.szymanski@eti.pg.edu.pl
bdudek@eti.pg.edu.pl
pzal@eti.pg.edu.pl
szydom@eti.pg.edu.pl
mczar@eti.pg.edu.pl

Abstract. In this paper we propose a method for answering class 1 and class 2 questions (out of 5 classes defined by Moldovan for TREC conference) based on DBpedia and YAGO. Our method is based on generating dependency trees for the query. In the dependency tree we look for paths leading from the root to the named entity of interest. These paths (referenced further as fibers) are candidates for representation of actual user intention. The analysis of the question consists of three stages: query analysis, query breakdown and information retrieval. During these stages the entities of interest, their attributes and the question domain are detected and the question is converted into a SPARQL query against the DBpedia and YAGO databases. Improvements to the methods are presented and we discuss the quality of the modified solution. We present a system for evaluation of the implemented methods, showing that the methods are viable for use in real applications. We discuss the results and indicate future directions of the work.

1 Introduction

Work on automatic question answering systems started as early as the late sixties [26]. Over the years, the systems became more and more complex while often producing very good results [29, 10]. Currently, most of the approaches are based on keywords where the answer is derived directly from the specified by the user keywords and can be either an explicit answer or (usually) the set of documents containing the keywords from the query (and potentially containing the answer). In the latter case the results obtained in this way are quite good; however, they require additional user verification and lookup within the documents provided.

The situation differs when questions are formulated in natural language. In this case there are no explicitly given keywords. The nature of the natural language can make the queries further ambiguous due to indirect subjects or lack

of context. In this case many systems rely on identifying interrogative pronouns to detect the entity of interest. Such an approach however does not work well for common-sense knowledge questions like "How many legs does a dog have?".

The questions formulated in natural language vary in difficulty depending on their complexity and ambiguity. Dan Moldovan et al [22] defined five classes of question difficulty. The first class consists of factoid questions, where the answer usually can be directly found in the database ("When did Beethoven died?"). The second class requires some knowledge about the question and the database structure. In this case the answer might not be syntactically close to the question ("Who is the spouse of Grover Cleveland?"). Class 3 of questions requires reasoning that is based on multiple, not always compatible, sources; class 4 of questions are interaction-based, and class 5 questions that require expert systems able of analogical reasoning.

In this paper we present a method for answering questions formulated in natural language. In our research we focus on the first two classes. The aim of the proposed method is to answer questions of class 1 and 2, providing, wherever possible, direct answers as found in DBpedia and YAGO databases. Most of the other solutions use their own, dedicated knowledge bases and either are complicated systems, or provide a list of potential answers for the user to choose from. The YAGO and DBpedia databases are vast and constantly improving, so we decided to base our solution on these two sources. The paper also aims at evaluating the solution against the well-known TREC question database.

The structure of the paper is as follows. Section 2 describes different approaches to question answering and our previous evaluation solution that served as a baseline for our approach. Section 3 presents in detail our approach, and shows improvements introduced to the algorithm. In Section 4 evaluation of the method is given. Finally in Section 5 we discuss the results and draw conclusions.

2 Existing solutions

Over the years many interesting approaches to question answering emerged [5, 21]. The first systems, like PRECISE [25] or BASEBALL [15], focused on providing natural language interfaces to databases, mapped user queries to SQL queries.

Further, question answering systems were proposed on a selected open domain. Here the approaches were not fine-tuned towards a specified domain, and needed to give answers to general questions. Many of the solutions were prepared during the TREC conference [33]. The best of those systems could answer as many as 70% of the questions from TREC database [32], ranging from simple factoid questions to complex, indirect questions. Some like LASSO [22], used deep lexical analysis of the question, to provide the answer using an iterative process. Others, like QRISTAL [17] or QALC [12], were based on semantic similarities and usually map the question into triples/queries. These approaches differ also in terms of complexity, from knowledge-rich systems [16] to simple systems like AskMSR [2].

Another group of systems are ontology-based solutions. Such systems take queries given in natural language and, based on the used ontology they return the answer from one or more knowledge bases that are compatible with that ontology. Examples of such systems represent a very broad spectrum of solutions. Some, like SQUALL [11], SPARQL2NL [24] or ORAKEL [6] convert the question into a SPARQL query that is evaluated against the given knowledge base.

In recent years, one very complex state of the art system was created, namely IBM Watson [14]. The system was developed as part of the DeepQA Project, started in 2007. The first IBM Watson implementation was made using a cluster consisting of around 2500 CPUs, 15 TB of RAM and, without a connection to the Internet. The quality of the system was so good that it managed to win the Jeopardy TV show [13]. Its strength comes from multiple algorithms that cooperate to calculate the best answer. The drawback of this solution is its complexity and limited availability for the wider audience.

In our approach we aim at providing a Wikipedia-based solution for a question answering system. Wikipedia itself is not very formalized, but previous research shows that it can be formalized [28, 30]. DBpedia [1] and YAGO [27] based solutions are viable for class 1 and class 2 questions. Other researchers also follow this route. Adel Tahri et al. proposed a Support Vector Machines-based algorithm for a DBpedia-based question answering system [31]. QASYO [23] is a YAGO based system designed to use YAGO ontology to answer questions. Mohamed Yahya et al. combined DBpedia and YAGO as sources to their approach and generated SPARQL queries based on the questions asked [34].

The aforementioned works, combined with our previous research, shows that such questions can be converted into formal SPARQL queries [4], which can be processed by the DBpedia and YAGO databases. The solution is based on the observation that dependency trees [7] of most of the queries had one or more paths leading from the root to the named entity of interest. These paths (so-called fibers) are candidates for representation of actual user intention. The goal of this step is to retrieve minimal fiber [4]. The question analysis consists of three stages: query analysis, query breakdown and information retrieval. The general architecture of the proposed solution was described in detail in our previous work [4]. In general the algorithm steps are as follows:

The first stage focuses on query retrieval and grammatical parsing. During this stage a Stanford NLP Parser [9] is used to detect structure of the query and convert it into an ordered tree [19]. Next step of this stage is concept retrieval where concepts are detected in the query, usually the entities and their properties. The sentence is tagged then using Penn-Treebank notation, and supplied with a list of detected entities and their properties.

The second stage aims at determining the entity and properties in question. During this stage we analyze dependencies within the query and perform minimization of the dependency tree. The created tree is then analyzed and fibers are detected within its branches. Finally the fibers detected are minimized.

The last stage, information retrieval, orders the fibers based on their score of relevancy and converts them to SPARQL queries which are used to retrieve the information from the database (YAGO and DBpedia).

The approach was tested using a subset of TREC-8[1] questions that were of class 1 and class 2 and gave precision of value 0.67 and recall equal to 0.36 resulting in F-measure equal 0.47. The specificity was 0.82 and the accuracy was 0.59 [4]. The results were not satisfactory, partially because of the algorithm's drawbacks and partially due to the quality of the databases, especially DBpedia. However, after a closer look many of the unanswered or wrongly answered questions had correct answers within the databases so we decided to extend our solution.

3 Extended fiber based solution

Both the original and extended solutions are looking for answers to the queries based on paths in dependency trees representing the user's query. Each such tree usually has one or more paths leading from the root to a named entity of interest. These paths (so called fibers) are candidates for representation of actual user intention. The original evaluation implementation had several drawbacks limiting its quality. During recent works we identified and tried to eliminate these drawbacks which allowed us to improve the quality of the proposed methods. The solution allows one to submit queries and retrieve answers along with full description of the analysis process. We are not focused on the performance of the algorithms as we use online YAGO and DBpedia endpoints which significantly influence the performance.

The first problem we identified in the original solution was the metric used to match question elements to attribute names. The original method used the Levenhstein [18] metric which gave high similarity for words with matching parts. Sequences like 'death place' were thus matched to attributes like 'deathPlace' and 'birthPlace'. This resulted in additional, usually wrong answers, e.g. for question *What is death place of Mohammad Khaled Hossain?* it produced six answers: *Mount Everest, Nepal, 2013-05-21, 2013-5-20, Bangladesh, Munshiganj*. We decided to use the difflib Python module[2] for the matching which allowed us to eliminate the additional answers. After changes, the system gave only 2, proper answers: *Mount Everest, Nepal*.

We also extended the algorithm for attributes matching. In the original solution the query parts were matched to the label of the attributes found in the used databases. After a closer look it occurred that the actual name of the attribute is often almost a direct match (whereas the label can be misleading).

Further, we also included redirection in possible matches for query attributes. If the attribute could not be matched but low scored candidates included redirection we followed those. This way it is possible to match entities like alternative names of cities or countries (e.g. *Ulan Bator* and *Ulanbaatar*).

[1] `http://trec.nist.gov/data/qa/T8_QAdata/topics.qa_questions.txt`
[2] `https://docs.python.org/2/library/difflib.html`

The original solution did not analyse synonyms when looking for entities or attributes matches. In the current implementation we use WordNet [20] as a backend for the nltk Python module[3] [3] for extending the list of attributes checked. For example the original program could not answer the question *Who is partner of Donald Trump?* as the databases do not have the attribute *partner* to identify the proper entity. The database contains however an attribute *spouse*. Using synonys we can modify the question into *Who is spouse of Donald Trump?* and thus get the correct answer. We also use the same mechanism when an attribute or its synonym cannot be found in the databases. In this case we use the nltk module to find the closest words for missing attributes and use them to generate the answer. An example of a question requiring such actions is *What is decease place of Chopin?* as neither *decease place* nor its synonyms could be found in the databases for the entity *Chopin*. This way we are able to generate alternative questions that should have the same answer, and in this way reach the answer.

Attributes name matching sometimes gave misleading results, e.g. in the question *Where is birth place of Jimi Hendrix?* attribute *birth place*, due to different similarity measures used [8], was mapped to attributes *birthPlace* and *birthDate*. To bypass the limitation of similarity measures in our approach we try to detect interrogative pronouns within the question and their domain. For answer generation we analyse only those matched attributes that share the same domain as detected from the interrogative pronoun of the question. This increased the quality of the answers but introduced another problem. In some cases the attribute is not within the same domain as the interrogative pronoun but the answer is, e.g. in the question *Who is successor of George Washington?* the attribute *successor* is not the attribute of a person. However the answer to the question, like *John Adams*, is. Thus, after finding a potential answer the domain check must be performed to check whether the answer is from the same domain regardless of the attribute that led to the answer.

The same nltk module also allowed us to ask questions where entities attributes are represented as actions, e.g. *When died Jim Morrison?*. Based on the verbs in the question we generate nouns, which are in turn the most often used as attributes names within DBpedia and YAGO. In this case there is no *Jim Morrison* entity attribute called *died*. There is, however, a proper answer stored under *deathDate*, which, combined with the domain deduction described earlier is derived from noun *death*, which in turn was generated using the nltk module from the verb *died*. So the answer was 1971-07-03.

The biggest problem occurred for entities with multiple meanings. Many words or names in natural language can have more then one meaning, e.g. entity *Washington*[4] which can have high number of pages related to. Such entities are usually represented using disambiguation entities that have references to different meanings of the entity in question. We look through those referenced entities for a potential answer and then match the domain of the referenced

[3] `https://www.nltk.org/`

[4] `https://en.wikipedia.org/wiki/Washington`

entity, the attribute and the interrogative pronoun to present the best answer. This mechanism still needs further work as it generates additional, mostly wrong, answers. The list of alternative meanings can also be long.

4 Evaluation

We evaluated the solution using the aforementioned TREC-8 questions that belonged to class 1 and class 2. Out of the 40 questions 32 had answers in the DBpedia and YAGO databases, the remaining 8 did not. For all non-answerable questions the algorithm did not give any answers. Out of 32 answerable questions the algorithm answered correctly 20 questions. In 2 cases we got wrong answers and the remaining 10 questions were left unanswered. In 7 cases the algorithm gave additional, wrong answers. Those additional answers are the reason why the sum of true/false positives/negatives is higher than the number of questions used for the evaluation.

Summary in terms of answer correctness and performance evaluation using precision, recall and F-measure are presented in Table 1 and Table 2 respectively.

Table 1. Answer correctness

True positives	True negatives	False positives	False negatives
20	8	9	12

Table 2. Performance scores

Precision	Recall	F-Measure
0.69	0.625	0.66

The updated algorithm achieved much higher recall (0.625 versus 0.36 in our evaluation implementation) while keeping precision at the same level (0.69 versus 0.67). The final F-measure increased from 0.47 to 0.66.

As can be seen in Table 3 there still were issues with some types of questions. In some cases the attributes detected within the query were matched incorrectly to entity attributes in the databases. In some cases this leads to providing completely wrong answers. It can be observed e.g. in question *Who is leading actor in 'The Godfather'?* where the attribute *leading actor* was marked as a synonym to *direct actor* and as a result matched to the attribute *director*. In other cases wrong match generated additional answers, like in question *What is location of Taj Mahal?* where word *location* had borderline similarity value of 80% with attribute *caption*. The same error can be observed in the question *What is population of Tucson?*. In this case the word *population* was incorrectly matched to attributes *populationAsOf, populationMetro, populationUrban, populationBlank*

Table 3. Examples of questions with wrong answers

Question	Answer in database	Correct answer returned	Wrong answer returned	Comment
What is location of US Declaration of Independence?	yes	no	yes	'position' from 'location' and 'position' property holds a value for picture ('right')
Who is leading actor in 'The Godfather'?	yes	no	yes	'direct actor' reasoned as synonym to 'leading actor', then associated with property 'director'
What is date of Battle of the Somme?	yes	yes	yes	wrong answers got from property 'seeAlso' ('see' was obtained as a synonym to word 'date')
What is location of Taj Mahal?	yes	yes	yes	wrong answers got from property 'caption' (which has 80% of similarity with word 'location')
What is population of Tucson?	yes	yes	yes	wrong answers from properties 'populationAsOf', 'populationMetro', 'populationUrban', 'populationBlank', 'populationEst' (too similar to 'population')
What is population of Ulan Bator?	yes	yes	yes	wrong answer from property 'populationAsOf' (too similar to 'population')
What is population of Ushuaia?	yes	yes	yes	wrong answers from property 'populationAsOf' (too similar to 'population')
What is produced by Peugeot?	yes	yes	yes	returned also answer '1739000' from property 'production'
Where was Washington born?	yes	yes	yes	problems with disambiguation

and *populationEst*. Such results were observed in almost all questions with additional, usually wrong, answers. In almost all cases however, the algorithm was able to give the correct answer alongside the wrong ones. This proves that the query analysis and entity detection process is correct. In further works we will focus on algorithms for better attribute matching.

In one case (*Who is Voyager manager?*) the question is formulated in a way that, even for humans, it is not easy to determine the correct answer without additional context information. We might assume the question asks for the manager of the Voyager space program, however the algorithm decided to match the

ambiguous name *Voyager* to a TV show *Earth Star Voyager*[5] and gave as the answer the name of the show director (which from a technical point of view can be treated as correct). In this case the algorithm should be able to generate more potential answers from different domains. The system as a whole could than present those answers and ask the user to specify the question by selecting the entity domain.

5 Conclusion

Proposed solution can, in most cases, answer the questions formulated in natural language and is not domain-specific. The strength of the proposed idea lies also in utilization of widely available tools and databases. In most cases such a system is also able to give a direct answer to the question asked (e.g. the date of an event in question). With certain improvements, mainly in matching and disambiguation algorithms, the system might be able to answer questions belonging to class 3 type of questions (answering based on multiple sources). The proposed query analysis method may also be used to extract semantic data from text.

The advantage of the system is its speed and small hardware requirements. The system can be run on a standard PC and does not require any other special resources. Many other solutions (especially IBM Watson) requires dedicated and powerful hardware.

Most of the problems found with the method are related to insufficient quality of attributes matching. Such cases were observed during analysis of results given for almost all questions with additional, usually wrong, answers. The algorithm however, in most cases, was able to give the correct answer alongside the wrong ones. This proves that the query analysis and entity detection process is correct.

Most of the wrong answers can also be eliminated by formalization and harmonization of attributes used to describe concepts within the same and different domains in the databases itself. The DBpedia and YAGO databases still use very shallow and not well interlinked internal ontologies. Data linking occurs on different levels and is very domain dependent. Constant development and formalization of the databases used can thus have a positive impact on the results obtained by the algorithm, and in time should have a positive impact on the quality of the proposed method.

The problems described in the evaluation section of this paper touch a very difficult problem which is natural language disambiguation. Further improvements of the proposed solution should be thus focused on improving the matching algorithm and ability to better understand the context and domain of the questions asked. We believe, that after further improvements, the proposed method can be used as a base for a system able to answer questions formulated in natural language.

[5] `https://en.wikipedia.org/wiki/Earth_Star_Voyager`

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: The semantic web, pp. 722–735. Springer (2007)
2. Banko, M., Brill, E., Dumais, S., Lin, J.: AskMSR: Question answering using the worldwide Web. In: Proceedings of 2002 AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases. pp. 7–9 (2002)
3. Bird, S., Klein, E., Loper, E.: Natural Language Processing with Python. O'Reilly Media (2009)
4. Boiński, T., Ambrożewicz, A.: DBpedia and YAGO as Knowledge Base for Natural Language Based Question AnsweringThe Evaluation. In: International Conference on Man-Machine Interactions. pp. 251–260. Springer (2017)
5. Bouziane, A., Bouchiha, D., Doumi, N., Malki, M.: Question answering systems: survey and trends. Procedia Computer Science 73, 366–375 (2015)
6. Cimiano, P., Haase, P., Heizmann, J.: Porting natural language interfaces between domains: an experimental user study with the orakel system. In: Proceedings of the 12th international conference on Intelligent user interfaces. pp. 180–189. ACM (2007)
7. Culotta, A., Sorensen, J.: Dependency tree kernels for relation extraction. In: Proceedings of the 42nd annual meeting on association for computational linguistics. p. 423. Association for Computational Linguistics (2004)
8. Czarnul, P., Rościszewski, P., Matuszek, M., Szymański, J.: Simulation of parallel similarity measure computations for large data sets. In: Cybernetics (CYBCONF), 2015 IEEE 2nd International Conference on. pp. 472–477. IEEE (2015)
9. De Marneffe, M.C., MacCartney, B., Manning, C.D., et al.: Generating typed dependency parses from phrase structure parses. In: Proceedings of LREC. vol. 6, pp. 449–454 (2006)
10. Duch, W., Szymański, J., Sarnatowicz, T.: Concept description vectors and the 20 question game. In: Intelligent Information Processing and Web Mining, pp. 41–50. Springer (2005)
11. Ferré, S.: Squall: a controlled natural language as expressive as sparql 1.1. In: International Conference on Application of Natural Language to Information Systems. pp. 114–125. Springer (2013)
12. Ferret, O., Grau, B., Hurault-Plantet, M., Illouz, G., Monceaux, L., Robba, I., Vilnat, A.: Finding an answer based on the recognition of the question focus. In: TREC (2001)
13. Ferrucci, D., Levas, A., Bagchi, S., Gondek, D., Mueller, E.T.: Watson: beyond jeopardy! Artificial Intelligence 199, 93–105 (2013)
14. Ferrucci, D.A.: IBM's Watson/DeepQA. In: ACM SIGARCH Computer Architecture News. vol. 39. ACM (2011)
15. Green Jr, B.F., Wolf, A.K., Chomsky, C., Laughery, K.: Baseball: an automatic question-answerer. In: Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference. pp. 219–224. ACM (1961)
16. Harabagiu, S.M., Moldovan, D.I., Pasca, M., Mihalcea, R., Surdeanu, M., Bunescu, R.C., Girju, R., Rus, V., Morarescu, P.: FALCON: Boosting Knowledge for Answer Engines. In: TREC (2000)
17. Laurent, D., Séguéla, P., Nègre, S.: Cross lingual question answering using qristal for clef 2006. In: Workshop of the Cross-Language Evaluation Forum for European Languages. pp. 339–350. Springer (2006)

18. Lcvenshtcin, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet Physics-Doklady. vol. 10 (1966)
19. Marcus, M.P., Marcinkiewicz, M.A., Santorini, B.: Building a large annotated corpus of English: The Penn Treebank. Computational linguistics 19(2), 313–330 (1993)
20. Miller, G.A., Beckitch, R., Fellbaum, C., Gross, D., Miller, K.: Introduction to WordNet: An On-line Lexical Database. Cognitive Science Laboratory, Princeton University Press (1993)
21. Mishra, A., Jain, S.K.: A survey on question answering systems with classification. Journal of King Saud University-Computer and Information Sciences 28(3), 345–361 (2016)
22. Moldovan, D.I., Harabagiu, S.M., Pasca, M., Mihalcea, R., Goodrum, R., Girju, R., Rus, V.: Lasso: A tool for surfing the answer net. In: TREC. vol. 8, pp. 65–73 (1999)
23. Moussa, A.M., Abdel-Kader, R.F.: Qasyo: A question answering system for YAGO ontology. International Journal of Database Theory and Application 4(2), 99–112 (2011)
24. Ngonga Ngomo, A.C., Bühmann, L., Unger, C., Lehmann, J., Gerber, D.: Sorry, I don't speak SPARQL: translating SPARQL queries into natural language. In: Proceedings of the 22nd international conference on World Wide Web. pp. 977–988. ACM (2013)
25. Popescu, A.M., Etzioni, O., Kautz, H.: Towards a theory of natural language interfaces to databases. In: Proceedings of the 8th international conference on Intelligent user interfaces. pp. 149–157. ACM (2003)
26. Simmons, R.F.: Natural language question-answering systems: 1969. Communications of the ACM 13(1), 15–30 (1970)
27. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of the 16th international conference on World Wide Web. pp. 697–706. ACM (2007)
28. Szymański, J.: Self–organizing map representation for clustering wikipedia search results. In: Asian Conference on Intelligent Information and Database Systems. pp. 140–149. Springer (2011)
29. Szymański, J.: Words context analysis for improvement of information retrieval. In: Computational Collective Intelligence. Technologies and Applications, pp. 318–325. Springer (2012)
30. Szymański, J., Duch, W.: Semantic memory knowledge acquisition through active dialogues. In: Neural Networks, 2007. IJCNN 2007. International Joint Conference on. pp. 536–541. IEEE (2007)
31. Tahri, A., Tibermacine, O.: Dbpedia based factoid question answering system. International Journal of Web & Semantic Technology 4(3), 23 (2013)
32. TREC: TREC 2008. `http://trec.nist.gov/data/qa/T8_QAdata/topics.qa_questions.txt` (2008), [Online: 19.11.2017]
33. Trec: Text REtrieval Conference (TREC). `http://trec.nist.gov/` (2012), [Online: 12.03.2018]
34. Yahya, M., Berberich, K., Elbassuoni, S., Weikum, G.: Robust question answering over the web of linked data. In: Proceedings of the 22nd ACM international conference on Conference on information & knowledge management. pp. 1107–1116. ACM (2013)