ISOFT at QALD-5: Hybrid question answering system over linked data and text data

Seonyeong Park, Soonchoul Kwon, Byungsoo Kim and Gary Geunbae Lee

Department of Computer Science and Engineering,
Pohang University of Science and Technology, Pohang, Gyungbuk, South Korea

{sypark322, theincluder, bsmail90, gblee}@postech.ac.kr

Abstract. We develop a question answering system over linked data and text data. We combine knowledgebase-based question answering (KBQA) approach and information retrieval based question answering (IRQA) approach to solve complex questions. To solve this kind of complex question using only knowledgebase and SPARQL query, we use various methods to translate natural language (NL) phrases in question to entities and properties in knowledgebase (KB). However, converting NL phrases to entities and properties in KB many times usually has low accuracy in most KBQA. To reduce the number of converting NL phrases to words in KB, we extract clues of answers using IRQA and generate one SPARQL based on the extracted clues and analyses of the question and the semantic answer type.

Keywords: Question answering, Hybrid QA, SPARQL

1 Introduction

Question answering (QA) systems extract short and preprocessed answers to natural language questions. QA is the fundamental goal of information extraction. In the big data era, this property of QA is increasingly gaining importance. Two popular types of QAs are knowledgebase-based question answering (KBQA) and information retrieval-based question answering (IRQA).

Recently, structured and semantically-rich KBs have been released; examples include Yago [1], DBpedia [2], and Freebase [3]. As an increasing quantity of resource description framework (RDF) data are published in linked form, finding intuitive ways to access the data is becoming increasingly important. Several QAs use RDF data; examples include Aqualog [4], template-based SPARQL learner (TBSL) [5] and Parasempre [6]. Because it is structured in linked form, the semantic web inference is possible [7], and KB is curated and verified by human, KBQA can provide more accurate answers than IRQA. However KBQAs require that the user's phrase be translated to entities and properties that exist in a KB. Many previous works use PATTY[1], pattern

[1] http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/patty/

based matching, and other methods, but it is still not sufficient to achieve high accuracy. Especially in complex question such as ‘who is the architect of the tallest building in Japan?’, system needs to map the predicate and generate queries in sequential way, and errors in mappings result in propagating errors.

To solve the problem, we combine IRQA approach and KBQA approach. We extract clues from the question using sequential phrase queries which are generated segmentation of the question using NLP tools such as chunking and dependency parsing. We define the answer clue as extracted entities to find final answer. We use multi-source tagged text database which is tagged with co-reference resolved and disambiguated form using Stanford co-reference resolution tool[2] and DBPedia Spotlight[3] [8]. To generate query, we don’t need to detect entity from every text in database in runtime because the entities have been already detected in document processing time. We concatenate next query and the answer of the first query and the next rightmost phrase. We repeat this process to find the answer of the given question. If we failed to find appropriate answer clue, we generate SPARQL query for one triple.

We use semantic similarity based on explicit semantic analysis (ESA) [9] to map predicates in the NL question to uniform resource identifiers (URIs) of properties in the KB. ESA converts target strings to semantic vectors that can convey their explicit meaning as weighted vectors of Wikipedia concepts, and calculating the similarity of two vectors reveals the semantic relatedness of the two strings from which the vectors were generated. We also increase the effectiveness of mapping the NL words to URIs by concatenating additional information to the predicate. If the property is related to arithmetic measurement (e.g. length or height), we map the predicate by pattern matching and rules.

After the final sequential phrase query is processed, we extract answer candidates and select the answer by using answer types and rules in the case of questions which requires to arithmetic comparison (e.g. ‘highest mountain’ and ‘tallest building’).

- Answer clue: entities to find final answer.
- Answer clue sentence: sentence which include answer clue.
- Predicate URI: property in DBpedia.

[2] http://nlp.stanford.edu/projects/coref.shtml

[3] https://github.com/dbpedia-spotlight/dbpedia-spotlight

2 System Description

2.1 Overall system architecture

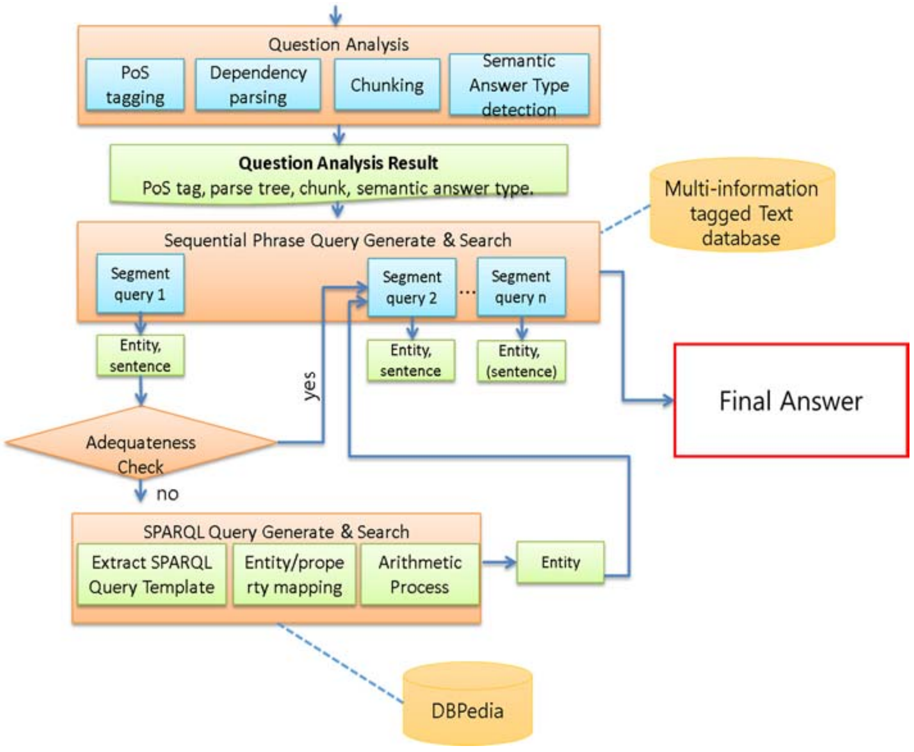
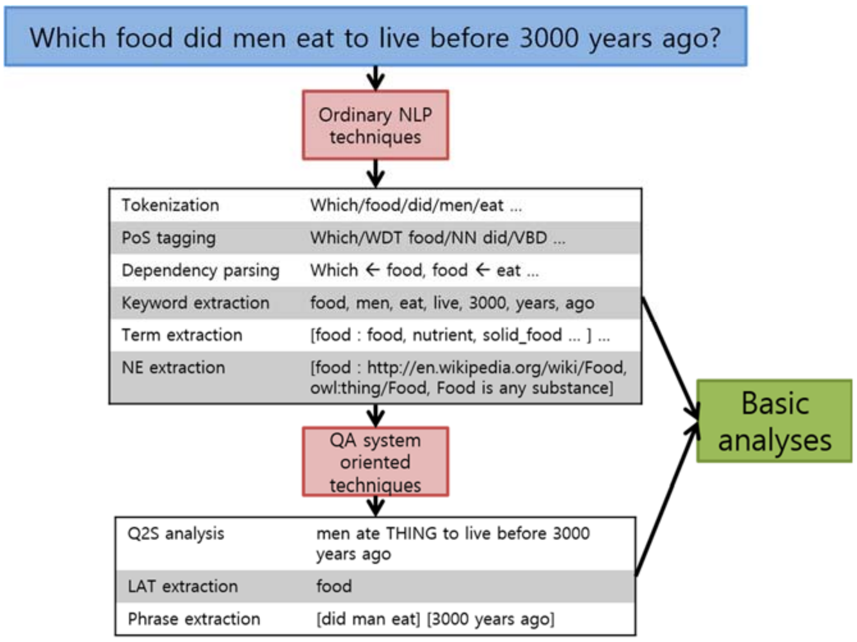


Fig. 1. Overall proposed QA system: processes are described in the text

In our system (**Fig 1**), first we analyze question, extract the sequential phrase query and classify the semantic answer type (SAT). For example, when the question is 'who is the architect of the tallest building in Japan?', the question is divided into three phrases; 'is the architect', 'of the tallest building', and 'in Japan'. Because most given hybrid questions in the QALD-5 are quite long to find answer at one time. We first search the query containing 'the tallest building' and 'in Japan' from the multi-information tagged text data. Then, we extract entities such as 'Tokyo_Skytree', 'Tokyo_Tower' and so on. If the query contains comparative form such as 'deeper' or superlative form such as 'tallest', we map these indicator to properties in KB. To extract the height of each entities, we generate SPARQL query such as *SELECT DISTINCT ?y WHERE { res:Tokyo_Skytree dbo:height ?y }*. Then, we compare the heights among the entities to get the tallest entity. If we failed to find answer, we generate SPARQL query to get answer from the tallest entity and the property which is in the KB mapped from the dependent of main verb or main verb itself. The SPARQL query is as follow, *SELECT DISTINCT ?y WHERE { res:Tokyo_Skytree dbo:architect ?y }*.


Which food did men eat to live before 3000 years ago?
Ordinary NLP techniques
QA system oriented techniques
Basic analyses

Tokenization	Which/food/did/men/eat ...
PoS tagging	Which/WDT food/NN did/VBD ...
Dependency parsing	Which ← food, food ← eat ...
Keyword extraction	food, men, eat, live, 3000, years, ago
Term extraction	[food : food, nutrient, solid_food ...] ...
NE extraction	[food : http://en.wikipedia.org/wiki/Food, owl:thing/Food, Food is any substance]

Q2S analysis	men ate THING to live before 3000 years ago
LAT extraction	food
Phrase extraction	[did man eat] [3000 years ago]

Fig. 2. Basic analyses of a question

2.2 Basic Analysis

To find the correct answer, we should analyze the input of the system, the question, carefully and thoroughly in various aspects (**Fig 2**). We use both statistical and rule-based approaches to analyze the questions. These analyses include ordinary NLP techniques such as tokenization and part of speech (PoS) tagging, and QA system oriented techniques such as question to statement (Q2S) analysis, lexical answer type (LAT) extraction, and SAT classification.

The ordinary NLP techniques include tokenization, part of speech tagging, dependency parsing, keyword extraction, term extraction, and named entity (NE) extraction. This information is not only important features for SAT extraction and answer selection, but also a basis for further question processing. We use ClearNLP[4] for tokenization, PoS tagging and dependency parsing. Keyword extraction is simply removing stop words and a few functional words, such as 'the', 'of' and 'please', from the question. Term extraction is finding nouns and verbs and their synonyms which exist in WordNet[5] dictionary. NE extraction uses Spotlight to map NEs in the question to entities in DBPedia[6]. The keywords, terms, and NEs are further used for SAT classification and query generation.

[4] https://github.com/clir/clearnlp
[5] https://wordnet.princeton.edu/
[6] http://wiki.dbpedia.org/

The QA system oriented techniques include Q2S analysis, LAT extraction and phrase extraction. The Q2S analysis is a rule-based analysis that recovers the corresponding declarative sentence from the interrogative or imperative sentence, the question. This analysis uses the result of the previous analyses, LAT, modal verbs, preceding preposition (e.g. "For whom does …"), usage of 'be' or 'do', and interrogative, to match the rules built for this system. The missing information which is asked through the question, which is called focus, is added to make a complete declarative sentence. LAT is a part of the question that limits the type of the answer, and gives a strong hint for SAT classification. LAT extraction is done along with Q2S analysis. Phrase extraction is to extract predicate phrase and prepositional phrase for query generation.

2.3 Query Generation

We have to generate Apache Lucene[7] queries to find sentences that contain the answer of the question from multi-information tagged text database. Some questions in QALD task cannot be solved with a single query. For example, a question 'who is the architect of the tallest building in Japan?' should be queried twice; 'Tokyo Skytree' from 'the tallest building in Japan' and 'Nikken Sekkei' from 'the architect of Tokyo Skytree' (**Fig 3**).

We devise sequential phrase queries to answer such questions. A unit of queries is a prepositional phrase or a predicate phrase. We generate the first query with concatenating the two rightmost phrases and find the answer. The next query is concatenation of the answer of the first query and the next rightmost phrase. We repeat this process to find the answer of the given question. The result of query: "tallest building in Japan" have many entities. If we failed to generate query in rightmost phrase, we used chunker and generate sequential query including more than one named entity. We filter many entities and select one answer cue. We can know whether the sentence including answer clues is related to query or not. We measure cosine similarity, Jaccard similarity between answer clue sentence and question statement. Also, we check whether named entities in question are in the answer clue statements or not. If the question include polarity such as "tallest", we select the entities satisfy the condition. If we failed to find answer clue, we generate SPARQL query to get answer candidates.

[7] https://lucene.apache.org/core/

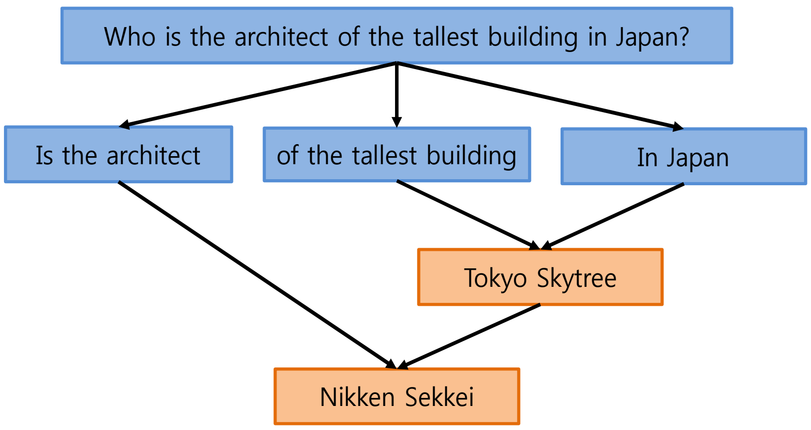


Fig. 3. Sequential phrase queries.

2.4 Semantic Answer Type Classification

Semantic answer type (SAT) is a very important feature in reducing wrong answer candidates. We can infer the SAT from the question before finding the answer candidates. For example, the answer of 'what did Bill Gates found?' can't be found before searching the database, but the SAT, 'ORGANIZATION' can be inferred from the question itself. Instead of other typesets built for SAT classification such as UIUC typeset [10], we use open typeset from DBPedia because the DBPedia typeset covers most entities properly and it can be extended as more entities are added to Wikipedia.

To classify the SAT, we used features from previous analyses such as keywords and LAT. The instances of the entities are not used because type rather than instance of each entity is important. For example, SAT of 'what is the capital of France?' is more similar than that of 'what is the capital of Germany?' than 'who is the president of France?', even though the first and the third question shares the same NE, 'France'. Thus we replaced the NE instances to type of each NE for features. We used other features such as interrogative wh-word, predicate and its arguments.

We train the SAT classifier with libSVM[8] and train data from four years of previous QALD challenges. We achieve 71.42 % accuracy for 3-level type ontology and 84.62 % for 2-level type ontology.

2.5 Multi-information Tagged Text Database

Even though most classical IR systems search the answer from plain text, we search the answer from multi-informational tagged text database. Texts on web are ambiguous and not structured, such as in 'Obama is the president of America. He is born in Hawaii.', where 'He' should be co-reference resolved to 'Obama', and 'Obama' should be disambiguated to 'Barack Obama'. We use Stanford coreference tool for co-reference resolution and DBPedia Spotlight for disambiguation.

---

[8] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

However, such processing is not a fast job at runtime: It takes more than three weeks to process all texts in Wikipedia. That is why we store such information along plain text in multi-information tagged text database. For each sentence in Wikipedia, we have stored 1. plain text, 2. tagged text with co-reference resolution and disambiguation information, 3. title from which Wikipedia page the sentence is, 4. PoS tagging, dependency parsing, and SRL result. We used Apache Lucene to store and index these attributes.

When the system throws a SPARQL query, the Apache Lucene search engine finds the related 'tagged texts'. Our system selects all NEs from all 'tagged texts' as the answer candidates. Of course most of them are irrelevant to the question, we use our SAT and answer selection module to prune out such answer candidates.

2.6 SPARQL query template generator

We detect words from each question to extract the appropriate SPARQL template.

- Questions including arithmetic information
  - If the query contains a comparative word such as 'deeper' or a superlative word such as 'deepest', we map these indicators to properties in KB based on mapping rules such as PATTY. We generate SPARQL query to find the answer candidates for further comparison. If the question contains comparative word 'deeper', we select answer candidates those are 'deeper' than the criterion in the question. If the question contains superlative word 'deepest', we sort the answer candidates searched from the query and get the candidate which is the 'deepest'. We used polarity information of adjectives. If the adjectives "highest", we sorted entities in descent order, and the adjectives "lowest", we sorted entities in ascending order.
- Yes/No questions
  - Using lexical information, we detect whether the question is a 'wh-question' or 'yes/no question'. If the question does not include 'wh-keywords' nor 'list-question keywords' (e.g., 'Give', 'List'), we regard the question as a 'yes/no question'.
- Simple question
  - If the question is not including arithmetic information nor yes/no question, we generate SPARQL query for one triple. In this case, we map predicates to properties in KB using lexical matching. If we fail to map using lexical, we try to use semantic similarity same as our previous work [11]. The proposed system extract important words which is related to predicate meaning. For example, just verb such as "start" did not assign a high score to the desired predicate URI, but concatenating additional information which make up for meaning of predicate works well. The proposed system used ESA lib which converts strings to semantic weighted vectors of Wikipedia concepts.

3 Experiment

We use the QALD-5 hybrid question test dataset for task 1 to evaluate the proposed method and use multi-information tagged text database built from Wikipedia as the text database and DBpedia 3.10 (DBPedia 2014) as the KB. We mainly compare two approaches. One is without semantic answer type and the other uses sematic answer type to filter answer candidates. We follow the QALD measurement. Correct states for how many of questions were answered with an F-1 measure of 1. Partially correct specifies how many of the questions were answered with an F-1 measure strictly between 0 and 1. Recall, Precision report the measures with respect to the number of processed questions. F-1 Global reports the F-1 measure with respect to the total number of questions.

4 Results and Discussion

To evaluate the QA system, we use global precision, recall and F-measure (**Table 1**). With S_AT yield a higher F-measure than Without S_AT. For example, semantic answer type detector extract "City" as answer type when the sentence "In which city where Charlie Chaplin's half brothers born?" is processed, so answer candidates such as United Kingdom and England can be filtered.

Table 1. Evaluation results for the QALD-5 test dataset.

Method	Total	Correct	Partially Correct	Recall	Precision	Global F-1 measure
Without S_AT	10	2	1	1.00	0.78	0.26
With S_AT	10	3	0	1.00	1.00	0.33

Error Case Analysis

1. Where was the "Father of Singapore" born?
   a. Semantic answer type: PLACE
   b. Query generation: "Father of Singapore" born/ Where
   c. Fail reason: cannot find relevant answer clue
2. Which Secretary of State was significantly involved in the United States' dominance of the Caribbean?
   a. Semantic answer type: ADMINISTRATIVEREGION
   b. Query generation: Unites' dominance of the Carbbean/involve/Secretary
   c. Which Secretary/of/State/was significantly involved in/the United States' dominance of the Caribbean
   d. Fail reason: cannot find relevant answer clue (both IR approach and KB approach)
3. Who is the architect of the tallest building in Japan?
   a. Semantic answer type: Building

   b. Query generation: “tallest building in Japan”-> result: Tokyo_Skytree/ architect + Tokyo_Skytree
   c. Success: The proposed system find tallest building in Japan is “Tokyo_Skytree” and we mapped “architect” to “architect” which is DBpedia predicate URI using only lexical information. Because the lexicals in NL predicate and predicate URI are same.
4. What is the name of the Viennese newspaper founded by the creator of the croissant?
   a. Semantic answer type: PERSON
   b. Query generation: “creator of croissant/ Viennese newspaper founded /
   c. Fail: cannot map the founded to predicate URI in DBpedia.

5. In which city where Charilie Chaplin’s half brothers born?
   a. Semantic answer type: City
   b. Query generation: Charlie Chaplin half brother/born city
   c. We find the half brother of Charlie Chaplin in multi-tagged text database using IR approach, and using KB sparql such as “select ?uri {Sydney_Chaplin birthplace?uri}, Finally we extract answer such as “England” and “London”. Using Semantic answer type, we can filter “England”.
6. Which German mathematicians were members of the von Braun rocket group?
   a. Semantic answer type:Person
   b. Query generation: member von Braun rocket group/German mathematician
   c. Fail: cannot find relevant answer clue in multi-tagged text database and KB.
7. Which writers converted to Islam?
   a. Semantic answer type: Person
   b. Query generation: writer converted Islam
   c. Fail: cannot find relevant answer clue in multi-tagged text database and KB.
8. Are there man-made lakes in Australia that are deeper than 100 meters?
   a. Semantic answer type: Place
   b. Query generation: man-made lake Australia, deeper 100
   c. Success: extract answer clues of “man -made lake Australia”. The proposed system compare the length of each named entities and check the more than one river is deeper than 100 meters. The proposed system find the river which is deeper than 100 meters.

9. Which movie by the Coen brothers stars John Turturro in the role of a New York City playwright?
   a. Semantic answer type: PLACE

   b. Query generation: role of a New York City playwright/Coen brother stars
   c. Fail: cannot find relevant answer clue in multi-tagged text database and KB.
10. Which of the volcanoes that erupted in 1550 is still active?
   a. Semantic answer type: place
   b. Query generation: volcanoes/erupted in 1550/active
   c. Fail: cannot generate appropriate query to extract answer clue.

5 Conclusion and Future work

To process the complex question with reducing error in mapping NL-predicate to KB property, we use both KBQA approach and IRQA approach. First, we search query from multi-information tagged text data. If the results are not appropriate or related to arithmetic, we generate SPARQL query and search KB. This combined approach works reduce error from mapping predicate in NL to predicate URI. Furthermore, our semantic answer type detector filter answer candidates.

However, still many questions are answered wrong because of the failure of not finding relevant answer clue, mapping NL predicate to predicate URI and query segmentation. We will focus on extending query to find relevant answer clue well and mapping NL-predicate to predicate URI to improve QA system in the future work. Also, we developed semantic parsing and open information extraction for question processing well.

Acknowledgments. This work was supported by the ICT R&D program of MSIP/IITP [R0101-15-0176, Development of Core Technology for Human-like Self-taught Learning based on a Symbolic Approach] and ATC(Advanced Technology Center) Program - "Development of Conversational Q&A Search Framework Based On Linked Data: Project No. 10048448.

References

1. Suchanek, F. M., Kasneci, G., & Weikum, G. (2007, May). Yago: a core of semantic knowledge. In Proceedings of the 16th international conference on World Wide Web (pp. 697-706). ACM.
2. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., ... & Bizer, C. (2013). DBpedia-a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web Journal.
3. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., & Taylor, J. (2008, June). Freebase: a collaboratively created graph database for structuring human knowledge. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data (pp. 1247-1250). ACM.

4. Lopez, V., Uren, V., Motta, E., & Pasin, M. (2007). AquaLog: An ontology-driven question answering system for organizational semantic intranets. Web Semantics: Science, Services and Agents on the World Wide Web, 5(2), 72-105.
5. Unger, C., Bühmann, L., Lehmann, J., Ngonga Ngomo, A. C., Gerber, D., & Cimiano, P. (2012, April). Template-based question answering over RDF data. In Proceedings of the 21st international conference on World Wide Web (pp. 639-648). ACM.
6. Berant, Jonathan, and Percy Liang. "Semantic parsing via paraphrasing."Proceedings of ACL. Vol. 7. No. 1. 2014.
7. Sintek, M., & Decker, S. (2002). TRIPLE – A query, inference, and transformation language for the semantic web. In Semantic Web – ISWC 2002 (pp. 364-378). Springer Berlin Heidelberg.
8. Mendes, Pablo N., et al. "DBpedia spotlight: shedding light on the web of documents." Proceedings of the 7th International Conference on Semantic Systems. ACM, 2011.
9. Gabrilovich, E., & Markovitch, S. (2007, January). Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis. In IJCAI (Vol. 7, pp. 1606-1611).
10. Xin Li and Dan Roth. 2002. Learning Question Classifiers: The Role of Semantic Information. Proceedings of the 19th international conference on Computational linguistics-Volume 1. 1-7
11. Park, S., Shim, H., & Lee, G. G. (2014). ISOFT at QALD-4: Semantic similarity-based question answering system over linked data. In CLEF.