How to Answer Yes/No Spatial Questions Using Qualitative Reasoning?

Marcin Walas

Faculty of Mathematics and Computer Science,
Adam Mickiewicz University, Poznań, Poland
mwalas@amu.edu.pl

Abstract. We present a method of answering yes/no spatial questions for the purpose of the open-domain Polish question answering system based on news texts. We focus on questions which refer to certain qualitative spatial relation (e.g. Was Baruch Lumet born in the United States?). In order to answer such questions we apply qualitative spatial reasoning to our state-of-art question processing mechanisms. We use Region Connection Calculus (namely RCC-5) in the process of reasoning. In this paper we describe our algorithm that finds the answer to yes/no spatial questions. We propose a method for the evaluation of the algorithm and report results we obtained for a self-made questions set. Finally, we give some suggestions for possible extensions of our methods.

1 Question Answering Task

The goal of Questions Answering (QA) is to provide the answer to a question posed in the natural language. QA can obtain answers from the variety of data sources. Open Domain QA is aimed at processing a wide domain of questions (as opposed to Closed Domain QA which focuses on a specific topic).

We aim at developing an open domain QA system for the Polish language, which retrieves answers from the collection of news texts. We mainly focus on shallow methods, which do not require deep language analysis (e.g. semantic analysis). However, in order to process spatial questions we use some more sophisticated techniques, namely automatic reasoning.

In this paper we describe a method for finding the answer to a yes/no spatial question. Yes/No questions (or polar questions) are questions with two answers possible: yes or no (e.g. *Was Baruch Lumet born in Poland?*). We define a yes/no spatial question as a yes/no question which refers to some spatial relation (e.g. is something located in a particular place). The method has been implemented in our QA system prototype Hipisek.pl (www.hipisek.pl).

1.1 Previous Work

There are several approaches to the problem of answering yes/no questions. The Webclopedia project is an example of the open domain QA system for

A. Gelbukh (Ed.): CICLing 2012, Part II, LNCS 7182, pp. 330–341, 2012.

the English language (see: [1]). In Webclopedia questions are represented in structures composed from `QTargets` and `QArgs`. QTargets can be described as an expected information type which should be provided in the answer. QArgs are the arguments of the QTarget (e.g. important parts of the question such as named entities). The system extracts answers by matching potential answer-bearing text fragments to instantiated QTargets.

Another approach consists in incorporating a reasoning system into the QA system. Such approach is taken in the SHAKEN system described in [2]. SHAKEN is a system for knowledge entry through the graphical assembly of concepts. One of its functionalities is to answer questions posed by experts. The mechanism for answer extraction includes application of the RCC-8 calculus and the Cardinal Calculus (which both are constraint calculi). The answer is found using path-consistency algorithm. The system belongs to the closed-domain QA category. SHAKEN was developed for the English language.

There exist a few QA systems for the Polish language. [3] describes the POLINT-112-SMS system. One of its components is a QA module. In the process of answering spatial questions (including yes/no questions) the system uses constraint reasoning (namely Cardinal Calculus). The system belongs to the closed-domain category, since it focuses on a narrow domain of topics, concerning public safety. For the authors knowledge there exists no open-domain QA system for the Polish language with similar capabilities for answering yes/no questions, as those implemented in our project.

The problem of answering yes/no questions can be reformulated as follows. Assuming that we are able to identify documents from which the answer should be extracted we can treat this problem as a sort of Natural Language Inference (NLI). NLI consists in determining whether a natural language hypothesis h can be reasonably inferred from a natural language premise p [4]. There are several approaches to NLI including shallow approaches (e.g. by using pattern based extraction or lexical overlap) and deep approaches (e.g. by using full semantic analysis and performing automated reasoning). In [5] it was showed that QA systems can benefit from the NLI incorporation.

1.2 Hipisek.pl Project Overview

In our project we incorporated NLI techniques into the QA system by treating the potential answer-bearing documents set as the set of premises and the question under consideration as the hypothesis.

We define three classes of the potential answer: `YES`, `NO` and `UNKNOWN`. We additionally require that the system extracts an explanation for the given answer (e.g. by showing a part of text from which the answer was extracted). In order to extract the answer the question is transformed into `QQuery` entity (which is similar to the notion of `QTarget` in the Webclopedia project).

In the **baseline version** of our system a method of answering a yes/no question was based on shallow NLI algorithms described in [4]. It includes lexical overlapping with weighting (e.g. named entities have higher weight in the process of weighting).

We process the question in the following steps:

1. Transformation into QQuery representation.
2. Retrieval of the documents and paragraphs (short extracts), which are likely to contain an answer for the question.
3. Answer extraction from the retrieved documents/paragraphs set.

One of the QQuery's elements is a question topic. **Question topic** is the main subject of the question (we assume that the question is a single non-contextual question, hence it contains only one topic). The question is transformed into QQuery representation using self-made rules set and a set of heuristics. The full description of the QQuery representation is given in [6].

In order to retrieve documents and paragraphs we use information retrieval techniques such as transformation of the question to a search engine query or lexical overlapping (similar to those described in [1]).

QQuery representation with a set of paragraphs form the input for the yes/no answering module.

1.3 The Notion of Space in Yes/No Questions

We observed that naive methods are insufficient for handling spatial questions. This is due to the fact that the process of answering a question related to some spatial entity depends on information which is not present directly in the source text. For example, consider the following part of the news article:

> Tulips named after Maria Kaczyńska, [...] will be placed in the tomb of the Presidental Couple in the Wawel castle.

One can ask the following question: *Is the tomb located in Kraków?* (for which the answer is "Yes"). The system has to determinate if the Wawel castle is located in the city of Kraków. Certainly this information is not present in the article extract, although it is obvious for the Polish citizen.

As a solution we propose the methodology of spatial reasoning. We use Region Connection Calculus in the process of answer extraction. For our purposes we have chosen the RCC-5 calculus which allows for five relations between regions. The idea is as follows: we extract a spatial relation from a question and check whether it is consistent with those extracted from the article's text. The answer is "Yes" if there are no identified inconsistencies or "No" otherwise. We use qualitative spatial knowledge database to maintain *naive* knowledge.

The paper is organized as follows: first we describe our approach to spatial knowledge representation. Next we give a short overview of our QA system and the input data for the answering algorithms. We describe algorithms and the application of the reasoning module. Next we propose a method for evaluation of algorithms and present results. Finally we formulate conclusions and discuss further work.

2 Maintaining Spatial Knowledge

In the section we briefly describe knowledge database and its representation. We present RCC-5 as a method for spatial reasoning in our system. We discuss the consistency issue in RCC-5. Finally we present our reasoning module.

2.1 RCC-5 Calculus

RCC is a topological approach to qualitative spatial representation based on a simple primitive relation of connection between regions [7]. Relationships are defined by means of the $C(a, b)$ relation (*connection relation*) which holds iff regions a and b share the common point. On the basis of relation C five base relations for RCC-5 are defined, namely: DR (*discrete*), EQ (*equal*), PP (*proper part*), PPI (*proper part inverse*) and PO (*partial overlap*) [8].

Any subset of a base relation set forms a relation in RCC-5. Thus RCC-5 contains 2^5 relations. A relation which contains all base relations is called the **universal relation**. The empty set of relations is called the **empty relation**. Following operations for the relations are possible: union, intersection, inversion and composition. RCC-5 relations are closed under composition and form a relation algebra [9]. A special compositions table is used to compute composition in RCC-5.

2.2 Reasoning with RCC-5

Using RCC-5 we can represent knowledge about entities in the form of constraints. A **constraint network** is formed from a set of variables V over the domain D and a set of constraints on the variables of V. A network is **consistent** if it has a solution which is an assignment of values of D to the variables of V in a way that all constraints are satisfied [9].

A path-consistency serves as a common approximation of the constraint satisfaction problem. A given constraint network is **path-consistent** iff for any consistent instantiation of any two variables there exists an instantiation of any third variable such that all three variables taken altogether are consistent. To enforce path-consistency, the following operation is used for all constraints between vertices i and j [9]:

$$\forall_k R_{ij} \Leftarrow R_{ij} \cap (R_{ik} \circ R_{kj})$$

The operation is used until a fixed point is reached. If an empty set results from the operation, then the network is not path-consistent. Otherwise it is path-consistent. In the general case path-consistency does not imply consistency. However, if the network is not path-consistent, it is not consistent either.

2.3 Knowledge Database

Spatial knowledge may be represented either quantitatively or qualitatively. The former way (e.g. by using absolute geographical coordinates) is certainly useful

in most engineering applications. However, for the sake of QA, the qualitative representation - storing relationships between spatial entities - is by far more useful, as this is how spatial information is represented in natural language [10].

We store entities and facts in the knowledge database. An **entity** is a notion of an individual entity in the real or abstract world. One entity belongs to exactly one **type**. A **fact** (or a **predicate**) is a triple: subject entity, predicate name and object entity. Predicate name is a name of the relation that holds between subject and object. Predicates are typed to form a taxonomy. An example of the fact that *Kraków is located in Poland* is a triple: *(city Kraków, is located in, country Poland)*.

We acquired a comprehensive qualitative spatial knowledge database by integration of several on-line data sources. Our knowledge database contains 300 000 facts concerning objects such as: world cities, countries, rivers, lakes, mountains, forests, famous buildings, administrative divisions and touristic attractions. It is used as a naive knowledge data source. Detailed description of the knowledge database and its acquisition is given in [11].

2.4 Reasoning Module

In order to reason about space we create a **constraint network**. A constraint network is created for the input fact f (for which we want to check consistency) and a set of additional knowledge facts F_e (which may be obtained during answer extraction process).

In the first step of reasoning we add all facts from F_e to the network and the subject entity of f (but not the f itself). Our aim is to obtain the object of f with constraint equal to spatial predicate of f by adding a fact from the spatial knowledge database (which introduce new entities in the network).

We make the following assumptions in the reasoning process:

- If we obtain an edge in the network, whose constraint is exactly equal to the constraint corresponding to spatial relation of f (e.g. `PP` relation is equal to *is located in* relation), then we assume that f is `TRUE`.
- If we obtain an inconsistent network, then we assume that f is `FALSE`.
- Otherwise we assume that f is `UNKNOWN`.

In each algorithm's step we retrieve all facts from the knowledge database, which correspond to the entities from the current network and we add retrieved facts to the network. The process is repeated until the network is stable.

We check if the obtained network contains an object of f. If it does then we run path-consistency algorithm on the obtained constraint network. If the network is not path-consistent then we return that f is `FALSE`. Otherwise we check if the constraint obtained between subject of f and object of f is equal to the input constraint of f. If it is, then we assume that the fact f is `TRUE`.

If the network does not contain the object of f, then we add f to the network and run path-consistency algorithm once more. If the network with f is inconsistent then the result is f is `FALSE`. Otherwise f is `UNKNOWN`. For details of the reasoning process see: [11].

For example, suppose we want to decide if the fact f: `(castle Wawel, is located in, country India)` is true. An initial constraint network is created with only two vertices: `castle Wawel` and `country India`. The network is expanded using knowledge database and the following network is obtained:

```
castle Wawel --PP--> city Krakow --PP--> country Poland
```

The entity: `country India` was not reached. Hence the fact f is added to the network. The following network is obtained:

```
castle Wawel
     ‘   ‘--PP--> city Krakow
     ‘             ‘--PP--> country Poland
     ‘                          ‘----DR --->
     ‘------------------------------PP----> country India
```

Note that `DR` relation between countries was inferred from semantics (see: [11]). This network is not path-consistent, hence p is `FALSE`.

3 Answering Algorithm

We divided yes/no questions into two groups:

- Questions which can be represented as a predicate (e.g. the question: *Is Paris located in France?*, can be represented as a predicate: `(city Paris, is located in, country France)`). Answering such a question is equal to proving that the predicate is true (or false). Check predicates are the model for the processed question.
- Questions which are represented as statements with additional constraints (e.g. the question: *Did Eric die in Africa?*, can be represented as a question: *Did Eric die?* and a constraint `(*, is located in, continent Africa)`). Answering such a question is equal to finding an answer for the statement and then checking whether the constraints are satisfied.

We introduce two properties to the QQuery representation:

- **Check predicates set** – which corresponds to the first group of the questions. All check predicates have to be proven in order to answer the question.
- **Constraint predicates set** – which corresponds to the second group of the questions. Constraint predicates have either subject or object undefined (marked as an asterisk in the example above). All constraints have to be satisfied to accept the answer.

The difference between check and constraint predicates is that a check predicate is a full representation of the question, where constraint predicates are only conditions, which have to be fulfilled by the extracted answer.

3.1 Answering a Question with the Check Predicates Set

Predicate checker processes each of QQuery's check predicates.

We assume that the answer is:

- `YES` if any of check predicates is true,
- `NO` if any of check predicates is false.
- `UNKNOWN` otherwise

The answer is **defined** if its either `YES` or `NO`. Otherwise it is **undefined**. Further question processing is suppressed if a defined answer is obtained.

Each of the check predicates p from the given QQuery with a set of paragraphs P is proven using the following algorithm:

1. Try to find predicate p in the naive knowledge database. If it is in the database then return `YES` and `STOP`.
2. Use the reasoning module on a single p predicate and create the constraint network using knowledge database. If the returned answer is defined then `STOP`.
3. Extract all predicates from paragraphs P which have equal subject or object as one of the subject or object of p.
4. For each extracted predicate p_e from P do
   (a) Use the reasoning module on p as an input predicate and p_e as an additional knowledge predicate. Create a constraint network using knowledge database.
   (b) If the reasoning module returns a defined answer then `STOP`.
5. If an undefined answer is obtained then return `UNKNOWN`.

Note that the constraint network, which was obtained during reasoning process, can serve as an explanation for the answer returned by the system.

If no answer is obtained then we move to the second answering mechanism, which uses constraint predicates.

Consider the following example: User poses questions which concern the following article extract:

> Tulips named after Maria Kaczyńska, [...] will be placed in the tomb of the Presidental Couple in the Wawel castle.

The first question is: *Is Wawel located in Poland?*. The system processes the question and transforms it into the *check predicate* structure: `(castle Wawel, is located in, country Poland)`. We perform *check predicate answering process* presented above. The system lacks the information of the desired predicate in the naive knowledge database (point 1 of the algorithm), hence it tries to use reasoning mechanisms. From the naive knowledge database the system retrieves facts that: `(castle Wawel, is located in, city Kraków)` and `(city Kraków, is located in, country Poland)`. The following constraint network is created:

```
Wawel -- PP --> Krakow -- PP --> Poland
  ‘-------------- PP ---------->
```

The network is path-consistent. We obtained the `PP` relation between *Wawel* and *Poland* vertices which is equal to *is located in* predicate. So the returned answer is: Yes.

A second exemplary question is: *Is the tomb located in Warsaw?*. The system transforms the question into the following *check predicate* structure: `(entity tomb, is located in, city Warsaw)`. Notice that the *tomb* entity has only general entity type identified. We perform *check predicate answering process*. The first two steps of the algorithm fail. Next the system extracts the fact that: `(entity tomb, is located in, castle Wawel)`. It uses this fact to obtain the following constraint network:

```
the tomb  -- PP --> Wawel -- PP --> Krakow
   ‘-------------- PP ------------>
```

The reasoning module did not prove the check predicate (the desired predicate was not obtained in the process, see section: 2.4), hence it adds check predicate to the network. The following network is obtained:

```
 the tomb
     ‘   ‘--PP--> Wawel
     ‘              ‘--PP--> Krakow
     ‘                           ‘----DR --->
     ‘-----------------------------PP----> Warsaw
```

Adding the check predicate fact to this network (`the tomb PP Warsaw`) leads to network inconsistency (since two cities are discrete, `DR` relation was inferred from semantics). Hence the system returns the answer: No.

3.2 Answering a Question with the Constraint Predicates Set

This method uses our baseline yes/no question answering module, which is based on the shallow NLI methods described in [4]. The module returns candidate answers with source sentences attached. Source sentences are sentences from the paragraphs which prove the answer chosen by the system. Candidate answers are verified by our constraint predicate verifier algorithm.

Constraint predicate verifier tries to satisfy all constraint predicates for the given candidate answer. We assume that the constraint predicate is satisfied if it is either TRUE or FALSE with reasoning using predicates extracted from the source sentence and its neighborhood (remark that falsification also satisfies constraint).

Deciding whether a given constraint predicate c is satisfied by the source sentence of the candidate answer s_a for the QQuery q is carried out using the following algorithm (we call this procedure *constraints verification procedure*):

1. Extract all predicates from s_a.
2. For each of the extracted predicates p_s do:
   (a) Attach constraint predicate c to p_s (link undefined subject/object of c with the corresponding entity of p_s).
   (b) If constraint c equals predicate p_s then return YES and STOP
   (c) Otherwise use the reasoning module on c with p_s as an additional knowledge predicate. Create a constraint network and check its path-consistency.
   (d) If the result is obtained then return it and STOP
3. If no result is obtained then return UNKNOWN (constraint is not satisfied).

Using the verification result we obtain the final answer value:

- If the verification is **positive** (all constraints are satisfied with `YES` result), then the answer is left unchanged.
- If the verification is **negative** (at least one constraint is satisfied with `NO` result and all of them are defined), then the initial answer is negated.
- If the verification is not satisfied (at least one constraint verification process returned `UNKNOWN`), then the answer is removed from the candidate set (we are unable to verify is it a correct answer or not, hence it will not be shown as a result).

Consider the following example: the user asks a question (concerning above mentioned text extract): *Will tulips be placed in Kraków?* The system transforms the question to a QQuery representation with the topic: *tulip* and one constraint predicate: `(*, is located in, city Kraków)` (note: an asterisk marks undefined subject). The baseline yes/no answerer extracts the answer *Yes* as a candidate answer (mostly due to the lexical overlapping for the topic of the question).

Now we perform constraint verification. We have one constraint to satisfy. The subject is attached to the constraint predicate obtaining constraint c equal to `(entity tulip, is located in, city Kraków)`. Next the system extracts the following predicate from the input paragraph: `(entity tulip, is located in, castle Wawel)`. For this predicate the following constraint network is created:

```
tulip  -- PP --> Wawel -- PP --> Krakow
   `------------- PP ------------>
```

The network is path-consistent. The constraint c was obtained during reasoning so it is satisfied with the `TRUE` value. The verification is positive, so the candidate answer is left unchanged: Yes.

4 Evaluation

Evaluation of question answering systems is a complex task. Several features of the QA systems can be considered in the evaluation process, such as: query language difficulty, content language difficulty, question difficulty, usability, accuracy, confidence, speed and broad domain [12]. Moreover evaluation of the

specific method leads to additional problems such as incompleteness of the indexed articles database (one can ask a question for which the answer is not present in any of the indexed articles).

In order to evaluate effectiveness of our algorithm we carried out an experiment which involved the creation of the testing corpus. Due to the lack of the QA system for the Polish language with similar capabilities as our QA project, we were unable to compare our results with the other systems.

4.1 Obtaining Data for the Evaluation

We decided to create a corpus of the questions from the predefined set of documents. From our knowledge database we randomly chose 40 articles. Each article was given to a tester, whose task was to create questions considering the article. The tester was instructed to pose only questions which contained a spatial relation and were the yes/no questions (still we did not define what the spatial relation is, leaving it to the tester's intuition). Each tester created a list which consists of questions, their correct answer and documents' ids from which the question was extracted.

4.2 Experiments and Results

We prepared two versions of our system:

- `BASE` – without the algorithms for reasoning (only the baseline version of yes/no questions answering was turned on);
- `FINAL` – with all algorithms turned on.

We carried out evaluation in two experiments:

- In *semi-supervised answering* experiment, only the answering module was run. The input to the system was a question with the corresponding document id, from which the answer should be extracted.
- In *full answering* experiment, the answering module was run as a part of the full answering process. An additional step of question processing was carried out by the system, which was a retrieval of the relevant documents, from which an answer could be obtained.

The first experiment checks whether the system is able to extract correct answer assuming that the document from which the answer should be extracted was properly retrieved (since the document was given as an input to the system). The second experiment checks the full answering process.

We computed *precision* as a fraction of the correct answers among all for which any answer was returned and *recall* as a fraction of the correct answers among all answers in the test set.

The first experiment is focused on evaluation of the answering process *in the sandbox*. This task is similar to the task of the NLI. This follows from the observation, that the document which is given as an answer source can be treated as a set of premises to the hypothesis included in the question.

Table 1. Results of evaluation experiments

	Base Supervised	Final Supervised	Base Full	Final Full
# of all questions	640	640	640	640
# of correct	218	248	127	206
# of processed questions	359	335	419	396
Precision	0.607	0.740	0.233	0.520
Recall	0.341	0.387	0.198	0.322
F-score	0.436	0.509	0.214	0.398

The second method measures the effectiveness of the developed method in the working QA system prototype. In fact results obtained in the second experiment can be misleading. When the testers prepared the test set of questions, they often assumed some context of the question posed (e.g. by using only the first name of the person, or using ambiguous phrases like: *the president* – the president of what?). Hence there exists a significant number of questions for which the expected answer can differ from the answer obtained by the system, but still both may be correct. To avoid this problem we required each answer provided by the system to be checked by the researcher (expected answers given by testers were not used). The researcher checked the answer and its explanation (e.g. paragraph retrieved or constraint network). Results of both experiments are given in Table 1.

Experiments show that our reasoning module increases precision of the system. In both experiments precision increased due to application of the reasoning module. As a consequence total number of the processed questions decreases (a processed question is a question for which a defined answer was returned). We report an increase in F-score for the system in both experiments.

Full experiment results show that the context is important issue in the process of evaluation. As expected, some answers were marked as incorrect in the automatic evaluation, but they were in fact correct. On the other hand, there were some questions which were marked as incorrect by the researcher due to bad explanation provided by the system.

5 Conclusions and Further Work

In the paper we presented a method for finding an answer for yes/no spatial questions. We applied automatic reasoning module into the state-of-art answering mechanism.

We plan to extend our method to process temporal questions. This goal can be achieved by means of application of a temporal reasoning mechanism (for which we plan to use Allen's calculus). The algorithm will be also used in other answering modules of our system (which process other types of questions, e.g. where?, when?, who? questions). It will be used to identify answers (for any spatial questions) that contradict.

References

1. Hovy, E.H., Gerber, L., Hermjakob, U., Junk, M., Lin, C.Y.: Question answering in webclopedia. In: TREC (2000)
2. Uribe, T.E., Chaudhri, V., Hayes, P.J., Stickel, M.E.: Qualitative spatial reasoning for question-answering: Axiom reuse and algebraic methods. In: AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases. AAAI, Stanford (2002)
3. Vetulani, Z., Marciniak, J., Obrebski, J., Vetulani, G., Dabrowski, A., Kubis, M., Osiński, J., Walkowska, J., Kubacki, P., Witalewski, K.: Language resources and text processing technologies. In: The POLINT-112-SMS System as Example of Application of Human Language Technology in the Public Security Area. Adam Mickiewicz University Press (2010)
4. MacCartney, B.: Natural language inference, Ph. D. Dissertation (2009)
5. Harabagiu, A., Hickl, A., Lacatusu, F.: Negation, contrast and contradiction in text processing. In: Proceedings of AAAI 2006 (2006)
6. Walas, M., Jassem, K.: Named entity recognition in a polish question answering system. In: Kopotek, M.A., et al. (eds.) Intelligent Information Systems, pp. 181–192. Publishing House of University of Podlasie (2010)
7. Randell, D.A., Cui, Z., Cohn, A.G.: A Spatial Logic based on Regions and Connection. In: Proceedings 3rd International Conference on Knowledge Representation and Reasoning (1992)
8. Bennett, B.: Spatial reasoning with propositional logics. In: Principles of Knowledge Representation and Reasoning: Proceedings of the 4th International Conference (KR 1994), pp. 51–62. Morgan Kaufmann (1994)
9. Renz, J.: Qualitative Spatial Reasoning with Topological Information. LNCS (LNAI), vol. 2293. Springer, Heidelberg (2002)
10. Renz, J., Rauh, R., Knauff, M.: Towards Cognitive Adequacy of Topological Spatial Relations. In: Habel, C., Brauer, W., Freksa, C., Wender, K.F. (eds.) Spatial Cognition 2000. LNCS (LNAI), vol. 1849, pp. 184–197. Springer, Heidelberg (2000)
11. Walas, M., Jassem, K.: Spatial reasoning and disambiguation in the process of knowledge acquisition. In: Vetulani, Z. (ed.) Proceeding of the 5th Language and Technology Conference, pp. 420–424 (2011)
12. Ferrucci, D., Nyberg, E., Allen, J., Barker, K., Brown, E.W., Chu-Carroll, J., Ciccolo, A., Duboue, P.A., Fan, J., Gondek, D., Hovy, E., Katz, B., Lally, A., McCord, M., Morarescu, P., Murdock, J.W., Porter, B., Prager, J.M., Strzalkowski, T., Welty, C., Zadrozny, W.: Towards the open advancement of question answering systems. Technical Report RC24789, IBM Research, Hawhtorne, NY (2008)