JUST.ASK, a QA system that learns to answer new questions from previous interactions

Sérgio Curto, Ana Mendes, Pedro Curto, Luísa Coheur and Ângela Costa
IST/INESC-ID
Rua Alves Redol 9
1000-029 Lisbon, Portugal
name.surname@l2f.inesc-id.pt

Abstract

We present JUST.ASK, a publicly available Question Answering system. Its architecture is composed of the usual Question Processing, Passage Retrieval and Answer Extraction components. Several details on the information generated and manipulated by each of these components are also provided to the user when interacting with the demonstration. Since JUST.ASK also learns to answer new questions based on users' feedback, (s)he is invited to identify the correct answers. These will then be used to retrieve answers to future questions.

Keywords: Question Answering, Pattern Matching, User Feedback

1. Introduction

Question Answering (QA) systems allow users to express their information needs in terms of natural language questions and to receive the exact answers to those questions.
In this paper, we present JUST.ASK, an open-domain QA system, specially tailored for factoid questions that is freely available.[1] Nevertheless, non factoid-like questions (like definitions) are also addressed. It implements several Artificial Intelligence techniques and takes advantage of different available information sources and tools. Some of its components can be easily modified/enhanced, allowing straightforward comparisons of (new) techniques. JUST.ASK attains state of the art results in Question Classification (Silva et al., 2011) and is detailed in (Mendes et al., 2013) and, to our knowledge, it is the first QA system that learns how to answer questions based on previous successful interactions. In this paper, besides the usual responses (answer plus supporting snippets), some internal data is also reported, like the result of the question classification or the generated clusters of candidate answers. Moreover, the user feedback about the correctness of the answers is also stored in order to be further applied to extract patterns and learn how to answer new questions.
This paper is organised as follows: in Section 2. we make a brief overview of JUST.ASK architecture, in Section 3. we detail its main components and in Section 4. we present the learning component. In Section 5. we describe the demo, in Section 6. we present some related work, and in Section 7. we present the main conclusions and point to future work.

2. JUST.ASK general architecture

When a new question is posed to JUST.ASK, its dataflow comprises the following steps: a) Question Processing – the question is interpreted; b) Passage Retrieval – a set of relevant passages/documents is retrieved from the available information sources; c) Answer Extraction – candidate answers are extracted and the correct ones selected. Figure 1 details JUST.ASK architecture.

[1] https://www.l2f.inesc-id.pt/wiki/index.php/Downloads

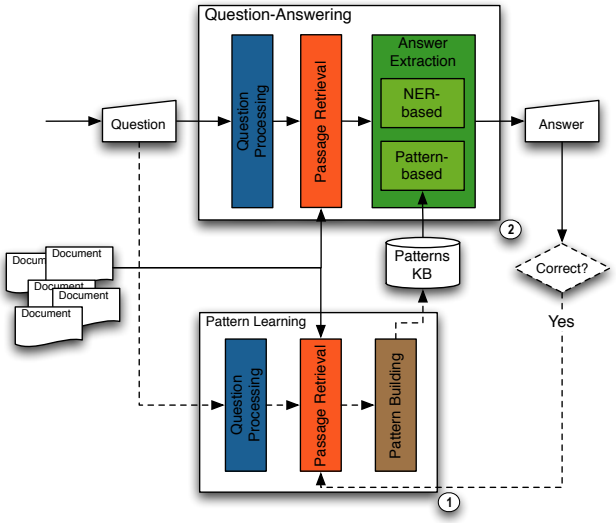

Figure 1: JUST.ASK general architecture

3. Overview of JUST.ASK components

3.1. Question Processing

The Question Processing component is responsible for two main tasks: question analysis and question classification. It receives as input a natural language question that is analysed by the Berkeley Parser (Petrov and Klein, 2007), trained on the QuestionBank (Judge et al., 2006), which returns the question tokens and the question syntactic components. Then, JUST.ASK question classifier exploits the widely used Li and Roth's (2002) two-layer question type taxonomy, consisting of 6 coarse grained categories and 50 finer grained ones, and allows the usage of different classification techniques. Thus, classification can resort to hand-built rules or machine-learning techniques, such as Support Vector Machines (SVM) or Naïve Bayes. State of the art results were attained by the classifier of JUST.ASK when modeling the task of question classification as a supervised learning classification problem (using SVM), although the most successful features used to train the model (unigrams

and semantic headword) are generated by the rule-based classifier. A detailed description of the classification task in JUST.ASK can be found in (Silva et al., 2011). Figure 2 describes the Question Processing component of JUST.ASK.

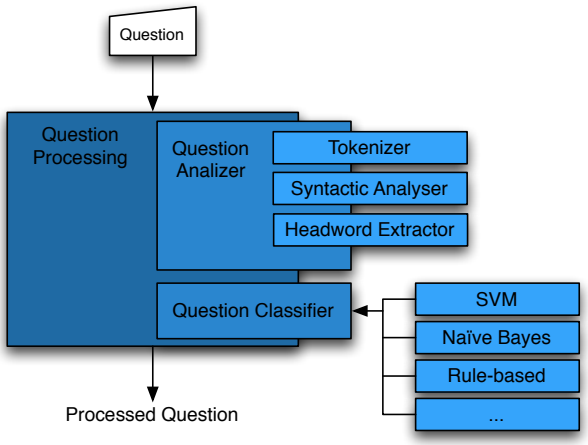

Figure 2: The Question Processing component

3.2. Passage Retrieval

The Passage Retrieval component of JUST.ASK receives as input the previously processed question and outputs the set of relevant passages for the posed question. JUST.ASK employs a multi-strategy approach to passage retrieval, with each strategy tailored to a specific group of question categories. Different strategies involve the use of different **information sources** and different **query formulations**. At the moment, JUST.ASK uses Lucene[2] search engine and the Bing search API[3] to retrieve relevant passages from unstructured sources (either specific corpora or the Web); Wikipedia[4] and DBpedia (Auer et al., 2007) are repositories of semi-structured content also used by the system. Due to their encyclopaedic nature, JUST.ASK uses the semi-structured sources to answer non factoid-like questions that require longer answers. In particular, Wikipedia is utilised to answer DESCRIPTION:DEFINITION and HUMAN:DEFINITION questions. Also, for questions whose cardinality is greater than one – i.e., list questions that require more than one answer – Wikipedia is also used. Once the information source to be used has been selected, and the queries have been formulated, the final step is to submit the queries to the information source's endpoint and retrieve the results. These queries are all submitted and processed in parallel, using multiple threads, in order to maximize the performance of the system. The results of each query are aggregated (with duplicates removed), and sent to the answer extraction component for further processing. Metadata about the results, such as the rank of each search result, is also stored and is available to be used when selecting the final answer. Figure 3 details the Passage Retrieval component of JUST.ASK.

[2] http://lucene.apache.org/.

[3] https://datamarket.azure.com/dataset/8818F55E-2FE5-4CE3-A617-0B8BA8419F65

[4] http://www.wikipedia.org/.

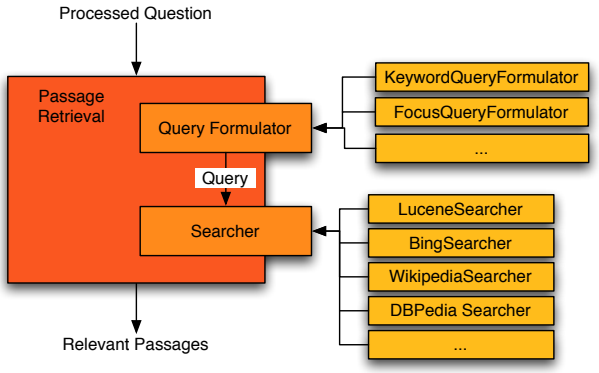

Figure 3: The Passage Retrieval component

3.3. Answer Extraction

The Answer Extraction component receives as input the data gathered in the previous steps (that is, both the processed question and the retrieved relevant passages) and is divided in two stages responsible for **candidate answer extraction** and **final answer selection**.

Considering the candidate answer extraction, strategies for each particular question category or groups of question categories are implemented. For instance, for NUMERIC type questions, we employ an extensive set of regular expressions to extract candidate answers, whereas for HUMAN type questions, we use a machine learning-based named entity recognizer. WordNet-based recognizers and Gazetteers are also used.

In what regards the final answer selection, we start by normalising candidate answers that belong to categories NUMERIC:COUNT and NUMERIC:DATE. Then, candidate answers are aggregated by lexical equivalence. The score of the new answer is the sum of the scores of all answers it comprises. A clustering step is performed afterwards, based on a measure that determines the similarity of two candidate answers. We can chose from the *overlap distance* and the *Levenshtein distance* (Levenshtein, 1966) normalized to the maximum length of the two answers being compared (other measures can be easily integrated in Just.Ask). The chosen distance is used in conjunction with a standard single-link agglomerative clustering algorithm, which works as follows: initially, every candidate answer starts in its own cluster; then, at each step, the two closest clusters, up to a specified threshold distance, are merged. The distance between two clusters is considered to be the minimum of the distances between any members of the clusters (as opposed to complete-link clustering, which uses the maximum). Finally, we filter the clusters, by discarding the ones in which an answer is contained in the original question, and the longest answer of the cluster with highest score is returned. Figure 4 shows the Answer Extraction component of JUST.ASK.

4. Learning with JUST.ASK

In a typical interaction with a QA system, a question is posed and its answer is returned. Usually, after being presented to the user, the system's answers are discarded from further processing. However, there is much information

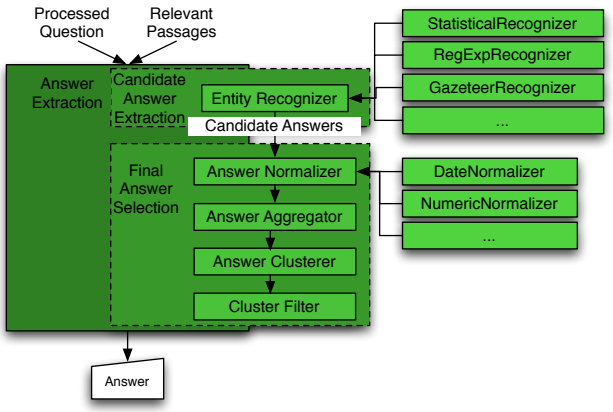

Figure 4: The Answer Extraction component

conveyed by the correct answer to a question that is simply lost in every interaction. For example, knowing that *1756* solves the question *When was Mozart born?* might be useful when answering the question *When was JFK born?*.
In a nutshell, our approach is based on (lexico-syntactic) patterns. Each one of these patterns relates a question to its answer, through the constituents of the sentence that contain (parts of) the question and the answer. To learn patterns, we use a minimally supervised approach where seeds are question/answer pairs. The process of learning patterns comprises three different sequential steps: a) the *Question Processing* step, where several actions are performed on the question, including its syntactic analysis and semantic classification; b) the *Passage Retrieval* step, where multiple queries are built from the question and the answer, and used to retrieve passages from the information sources (the Web or local corpora); and, c) the *Pattern Building* step, where the elements from the question, the answer and the retrieved passages are chosen to build the patterns. The learned patterns are then stored and indexed by the semantic category and syntactic segmentation of the seed question.
It is worth mentioning that, in this work, we define a pattern as a sequence of lexical and syntactic elements. For instance, "NP VBD [by] NP$_?$" is a pattern learned from *The Divine Comedy written by Dante* to answer the question *Who wrote The Divine Comedy?*. In the pattern, the syntactic elements[5] refer to the syntactic chunks of the question, except the one with the subscript question mark, which indicates the answer. The lexical content of the pattern is stated between square brackets.
Whenever a user gives feedback to JUST.ASK, by confirming that the answer to his/her question is correct, the system uses that question/answer pair to build new patterns. These patterns will be available in future interactions and will be used to extract candidate answers to new questions.

5. JUST.ASK demo

In Figure 5 we present a screenshot of a demo, showing the final answers to the question *What is the capital of Somalia?*.
As previously stated, the user can see some of the internal information created and manipulated by the system, namely the question headword, the question category, the POS tags of each word of the question and its syntactic constituents, the retrieved passages, the candidate answers and the resulting clusters. Also, the user has the opportunity to confirm the correctness of the answers returned from the system (Mark as correct/Mark as incorrect), which will allow JUST.ASK to create new patterns and learn how to answer new questions. In this version of the demo, however, we are only storing this information, and we do not resort to the pattern-based approach to extract candidate answers.
This demo of JUST.ASK only requires a browser and an internet connection. It can be tested in
`http://aurora.l2f.inesc-id.pt/Just.Ask-Web`

6. Related Work

Many works in the literature use questions and their correct answers in strategies for QA (several of them take advantage of the datasets of questions and answers built in the context of evaluation fora, like the Text REtrieval Conference (TREC) or the Cross-Language Evaluation Forum (CLEF)).
The use of Question/Answer (Q/A) pairs as training instances to (semi-)supervised learning machinery in QA is however not limited to the task of building patterns to candidate answer extraction. Sun et al. (2006) use Q/A pairs to learn classifiers that identify the correct answers for a question in a sentence. The first classifies a sentence as containing (or not) a correct answer to the posed question; if the sentence is classified positively, the second classifies each word as correct (or not). Moschitti and Quarteroni (2011) focus on the answer selection to definition questions, and base their work on the cross-pair similarity model (Zanzotto and Moschitti, 2006) that learns rewrite rules between two entailment pairs (*Text T*, *Hypothesis H*). They use questions and their answers as training instances and study the improvements achieved when using generalizations to syntactic/semantic structures and applying sequence/tree kernel technology in a SVM. Lita and Carbonell (2004) represent questions as points in a multidimensional space and group them in clusters according to their similarity, based on the idea that similar questions are solved by similar strategies. Different models are learned from each cluster that serve three different purposes: 1) estimate a distribution of the question's semantic category; 2) include cluster-specific content in the queries submitted to the document retrieval module; and, 3) identify if an answer is present in a text snippet. When a new question is posed, it is represented in the same space and the models of the clusters in its neighborhood are used.
Our pattern-based approach to candidate answer extraction allows JUST.ASK to learn to answer new questions based on previous successful interactions. For that, the system relies on the user feedback about the correctness of the returned answer(s). When it comes to systems that use past *interactions* to answer new questions, the approach of Harabagiu et al. (2001) is a rare example described in the literature: by using a caching mechanism, answers to previous similar questions are reused, which avoids triggering the entire QA process. The similarity is measured in terms of the number of (lexico and semantic) matches

[5] We use the Penn Treebank II Tags (Bies et al., 1995).

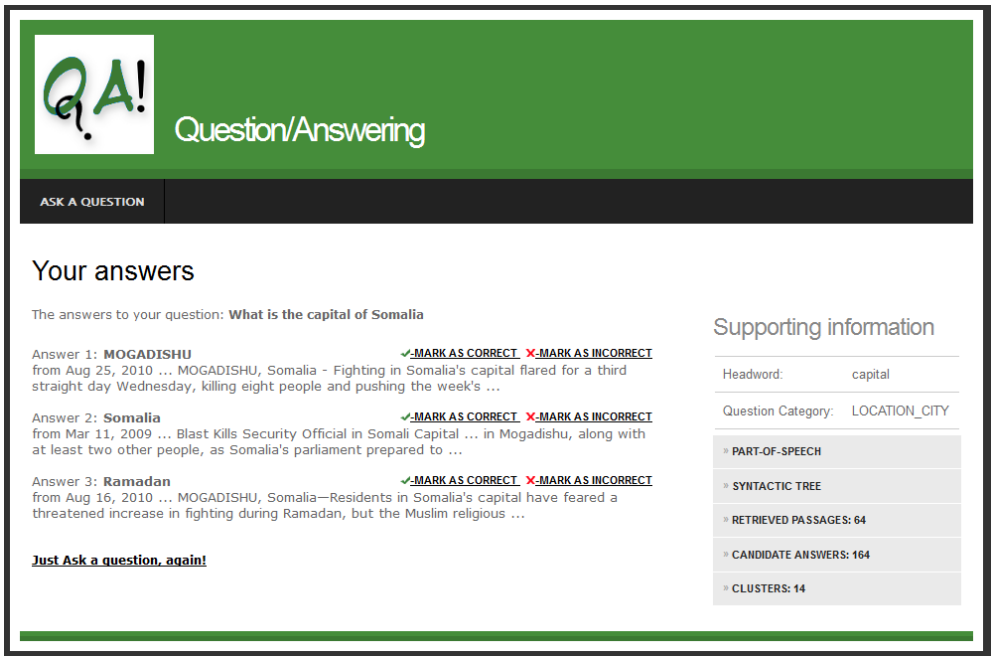

Figure 5: JUST.ASK demo

between content words of equal morphological category in both (current and cached) questions. Other works exist along the line of reusing answers from past questions in recent questions, specially applied to Community Question-Answering sites.[6] Here, the goal is to reduce to amount of unanswered questions and the main challenge is typically to identify the past question(s) that is(are) the most similar to the new question, the one(s) that convey the same information need. For example, Shtok et al. (2012) use a two-stage approach that, besides choosing the most appropriate past question through a ranking mechanism, also identifies the best answer to the new question from the list of answers to the top-ranked past questions. Finally, it should be mentioned that, although the direct user feedback is rarely used in traditional QA, its application is explicit in this type of sites, where answers are ranked based on several features, including the number of votes attributed by the community members.

7. Conclusions and Future Work

We presented JUST.ASK, a QA system that is freely available for research purposes. Users can have access to some internal information and, since JUST.ASK is able to learn from previous interactions, they can contribute to this process by indicating if an answer is correct. As future work, besides exploring the several steps of JUST.ASK, in particular the Answer Selection component, we would like to allow the user to provide the answer if the system is not able to deliver a correct one. We would also like to evaluate how robust the learning component is to wrong user feedback.

[6]In these sites, questions and answers are given by humans. Examples of such sites are Yahoo! Answers (`http://answers.yahoo.com/`), StackOverflow (`http://stackoverflow.com/`) or Quora (`http://www.quora.com/`).

8. Acknowledgements

This work was supported by national funds through FCT – Fundação para a Ciência e a Tecnologia, under project PEst-OE/EEI/LA0021/2013. Sérgio Curto and Pedro Curto scholarships were supported under project FALACOMIGO (ProjectoVII em co-promoção, QREN n 13449). Ana Cristina Mendes was supported by a PhD fellowship from Fundação para a Ciência e a Tecnologia (SFRH/BD/43487/2008).

9. References

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In *In 6th Int'l Semantic Web Conference, Busan, Korea*, pages 11–15. Springer.

Bies, A., Ferguson, M., Katz, K., MacIntyre, R., Tredinnick, V., Kim, G., Marcinkiewicz, M. A., and Schasberger, B. (1995). Bracketing guidelines for treebank ii style penn treebank project. Technical report, University of Pennsylvania.

Harabagiu, S., Moldovan, D., Pasca, M., Mihalcea, R., Surdeanu, M., Bunsecu, R., Girju, R., Rus, V., and Morarescu, P. (2001). The role of lexico-semantic feedback in open-domain textual question-answering. In *Proc. 39th Annual Meeting of the Association for Computational Linguistics*, pages 282–289. ACL.

Judge, J., Cahill, A., and van Genabith, J. (2006). Questionbank: creating a corpus of parse-annotated questions. In *ACL-44: Proc. 21st Int. Conf. on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 497–504. ACL.

Levenshtein, V. I. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707–710, February.

Li, X. and Roth, D. (2002). Learning question classifiers. In *Proceedings of the 19th international conference on*

Computational linguistics, pages 1–7, Morristown, NJ, USA. Association for Computational Linguistics.

Lita, L. V. and Carbonell, J. G. (2004). Instance-based question answering: A data-driven approach. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain*, pages 396–403.

Mendes, A. C., Coheur, L., Silva, J., and Rodrigues, H. (2013). Just.ask – a multi-pronged approach to question answering. *Artificial Intelligence Tools*.

Moschitti, A. and Quarteroni, S. (2011). Linguistic kernels for answer re-ranking in question answering systems. *Information Processing and Management*, 47(6):825 – 842.

Petrov, S. and Klein, D. (2007). Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York, April. Association for Computational Linguistics.

Shtok, A., Dror, G., Maarek, Y., and Szpektor, I. (2012). Learning from the past: answering new questions with past answers. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 759–768. ACM.

Silva, J., Coheur, L., Mendes, A., and Wichert, A. (2011). From symbolic to sub-symbolic information in question classification. *Artificial Intelligence Review*.

Sun, A., Jiang, M., and Ma, Y. (2006). A maximum entropy model based answer extraction for chinese question answering. In *Proceedings of the Third international conference on Fuzzy Systems and Knowledge Discovery*, FSKD'06, pages 1239–1248, Berlin, Heidelberg. Springer-Verlag.

Zanzotto, F. M. and Moschitti, A. (2006). Automatic learning of textual entailments with cross-pair similarities. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 401–408, Stroudsburg, PA, USA. Association for Computational Linguistics.