Biomedical Question Answering using the YodaQA System: Prototype Notes

Petr Baudiš and Jan Šedivý

Dept. of Cybernetics, Czech Technical University,
Technická 2, Praha, Czech Republic
baudipet@fel.cvut.cz

Abstract. We briefly outline the YodaQA open domain question answering system and its initial adaptation to the Biomedical domain for the purposes of the BIOASQ challenge (question answering task 3b) on CLEF2015.

Keywords: Question answering, linked data, natural language processing, bioinformatics.

1 Introduction

The YodaQA system for open domain factoid English question answering has been published recently. [1] [2] The system is a fully open source, modular pipeline inspired by the IBM Watson DeepQA system [3]. So far, the system has not been specialized for any particular domain, this represents the first such effort.

The BIOASQ challenge [4] aims at semantic indexing and question answering in the biomedical domain using a variety of knowledge bases from the given domain. The BIOASQ 2015 has tasks 3A and 3B, where 3A concerns semantic indexing and 3B is about question answering, where we participated.

The BIOASQ Task 3B is further split into phase A (information retrieval) and phase B (answer production); the phases are evaluated separately, with gold standard phase A results (documents, snippets and triples) available for phase B. Our system participated in the phase B evaluation.

The paper is structured as follows. In Sec. 2, we briefly outline the YodaQA system in its original form. In Sec. 3, we discuss the changes of the system for the biomedical domain. In Sec. 4, we review the system performance.

2 YodaQA Summary

The YodaQA pipeline is implemented mainly in Java, using the Apache UIMA framework [5]. Detailed technical description of the pipeline is included in a technical report [1].

The system maps an input question to ordered list of answer candidates in a pipeline fashion, encompassing the following stages:

– **Question Analysis** extracts natural language features from the input and produces in-system representations of the question. We currently build just a naive representation of the question as a bag-of-features. The most important characterization of the question is a set of clues (keywords, keyphrases and concept clues that exactly match enwiki titles) and possible lexical answer types.
– **Answer Production** generates a set of candidate answers based on the question, typically by performing a **Primary Search** in the knowledge bases according to the question clues and either directly using search results as candidate answers or filtering relevant passages from the result text (the **Passage Extraction**) and generating candidate answers from the filtered passages (the **Passage Analysis**). Answers are produced from text passages by a simple strategy of considering all named entities and noun phrases.
– **Answer Analysis** generates various answer features based on detailed analysis. Most importantly, this concerns lexical type determination and coercion to question type. Other features include distance from clues in passages or text overlap with clues.
– **Answer Merging and Scoring** consolidates the set of answers, removing duplicates and using a machine learned classifier (logistic regression) to score answers by their features.

3 YodaQA Domain Adaptation

We made a variety of adjustments to fit our end-to-end pipeline to the BIOASQ task. The changes are available within the public open source code base (`https://github.com/brmson/yodaqa`) in the `d/clef2015-bioasq` branch.

As a minor technical change, we enhanced our question analysis for imperative and otherwise specifically phrased questions which were uncommon in our TREC-based open domain dataset.

Our system is designed to answer just *factoid* questions, while the BIOASQ challenge also includes *list* and *yes-no* questions. For the *list* question, we simply use the top 5 answer candidates returned by the system. Since we implemented no text entailment algorithm yet and there was no easy way to skip *yes-no* questions, we simply use a fixed *yes* answer for all since it was significantly more prevalent in the training dataset.

Similarly, our system is designed to return narrow answers, not sentence-length answers that include background and justification. Therefore, we always return empty string as the *ideal* answer and supply our answers as the *exact* answer. However, the BIOASQ definition of exact answers might be more stringent than ours, which simply requires that the gold standard answer is a sub-string of the produced answers answer (e.g. "the red color" would be acceptable for gold standard "red" in our scenario). Therefore, we modified our system to require exact matches during training, and disabled a heuristic in answer analysis which attempts to find a *focus word* in the answer and run analysis (like title lookup, type coercion) on it instead of the whole answer.

As we focus on the phase B of the question answering task, we do not perform an explicit primary search on questions and instead base answer production on the search result snippets (generated by phase A) supplied along with the question on program input. These snippets are equivalent to passages our primary search would produce.[1]

To further improve the accuracy of the system, we implemented an enhanced answer production strategy (inspired by the **Jacana** QA system) that approaches the problem of identifying the answer in a text passage in a way similar to named entity recognition: as a (token) sequence tagging (by begin-inside-outside labels) that uses the conditional random field model to predict labels. [6] However, we use a significantly simplified feature set: just part-of-speech tags, named entity labels and dependency labels as token sequence unigrams, bigrams and trigrams.

A crucial feature for scoring answers is information on successful type coercion. This involves identification of Lexical Answer Type (LAT) in the question (typically an easy task using a few fixed heuristics), production of a set of LATs describing the answer and the question-answer type coercion using straight string matching and Wordnet [7] hypernymy relations. In an open domain QA system, the most useful LAT source is looking up the answer as a Wikipedia article by title and using the category information.[2] This may fail for specialized terminology of the biomedical domain, we also look up the answer in the **GeneOntology** [9] using the GOLR endpoint (as either of *bioentity, bioentity_label, bioentity_name, synonym*) and using the type field as the answer LAT; this is successful for correct identification of answers like gene and protein names.

4 Results

To evaluate the performance of our system, we split the (randomly reshuffled) reference training dataset provided for the Task 3B to a local dev/train set (100 questions) and a test set (100 questions).[3] For comparison, we also include baseline version performance[4] on the "curated" factoid open domain dataset [2]. The results are summarized in Table 1.

References

1. Baudiš, P.: YodaQA: A Modular Question Answering System Pipeline. In: POSTER 2015 - 19th International Student Conference on Electrical Engineering

[1] We did not participate in phase A due to a lack of resources on our side.

[2] In practice, DBpedia [8] `rdf:type` ontology can be leveraged for this task.

[3] The rest of the questions were unused; we opted for a smaller dataset as the GeneOntology access was causing quite a slow-down.

[4] This performance is done using the open domain metric which permits gold standard substrings, see above.

Pipeline	Recall	Accuracy-at-1	MRR
final	33.0%	10.0%	0.132
w/o GeneOntology	33.0%	8.0%	0.120
w/o G.O., CRF	33.0%	5.5%	0.114
final, ignoring yes/no q.	43.5%	10.1%	0.148
open domain	79.3%	32.6%	0.420

Fig. 1. Benchmark results of various pipeline variants on the test split of the dataset. MRR is the Mean Reciprocal Rank $|Q| \cdot \sum_{q \in Q} 1/r_q$.

2. Baudiš, P.: YodaQA: A Modular Question Answering System Pipeline. In: Sixth International Conference of the CLEF Association, CLEF'15, Toulouse, September 8-11, 2015. Volume 9283 of LNCS., Springer (2015)
3. Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A.A., Lally, A., Murdock, J.W., Nyberg, E., Prager, J., et al.: Building watson: An overview of the deepqa project. AI magazine **31**(3) (2010) 59–79
4. Tsatsaronis, G., Balikas, G., Malakasiotis, P., Partalas, I., Zschunke, M., Alvers, M.R., Weissenborn, D., Krithara, A., Petridis, S., Polychronopoulos, D., et al.: An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. BMC bioinformatics **16**(1) (2015) 138
5. Ferrucci, D., Lally, A.: UIMA: An architectural approach to unstructured information processing in the corporate research environment. Nat. Lang. Eng. **10**(3-4) (September 2004) 327–348
6. Yao, X., Van Durme, B., et al.: Answer extraction as sequence tagging with tree edit distance. In: HLT-NAACL. (2013) 858–867
7. Miller, G.A.: WordNet: a lexical database for english. Communications of the ACM **38**(11) (1995) 39–41
8. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., et al.: Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web (2014)
9. Consortium, G.O., et al.: Gene ontology consortium: going forward. Nucleic acids research **43**(D1) (2015) D1049–D1056