YodaQA: A Modular Question Answering System Pipeline

Petr Baudiš

Dept. of Cybernetics, Czech Technical University, Technická 2, 166 27 Praha, Czech Republic

baudipet@fel.cvut.cz

Abstract. ***This is a preprint, submitted on 2015-03-22.*** *Question Answering as a sub-field of information retrieval and information extraction is recently enjoying renewed popularity, triggered by the publicized success of IBM Watson in the Jeopardy! competition. But Question Answering research is now proceeding in several semi-independent tiers depending on the precise task formulation and constraints on the knowledge base, and new researchers entering the field can focus only on various restricted sub-tasks as no modern full-scale software system for QA has been openly available until recently.*

By our YodaQA system that we introduce here, we seek to reunite and boost research efforts in Question Answering, providing a modular, open source pipeline for this task — allowing integration of various knowledge base paradigms, answer production and analysis strategies and using a machine learned models to rank the answers. Within this pipeline, we also supply a baseline QA system inspired by DeepQA with solid performance and propose a reference experimental setup for easy future performance comparisons.

In this paper, we review the available open QA platforms, present the architecture of our pipeline, the components of the baseline QA system, and also analyze the system performance on the reference dataset.

Keywords

Question answering, information retrieval, information extraction, linked data, natural language processing, Apache UIMA, software engineering.

1. Introduction

We consider the Question Answering problem — a function of unstructured user query that returns the information queried for. This is a harder problem than a linked data graph search (which requires a precisely structured user query) or a generic search engine (which returns whole documents or sets of passages instead of the specific information). The Question Answering task is however a natural extension of a search engine, as currently employed e.g. in Google Search [22] or personal assistants like Apple Siri, and with the high profile IBM Watson Jeopardy! matches [10] it has became a benchmark of progress in AI research. As we are interested in a general purpose QA system, we will consider an "open domain" general factoid question answering, rather than domain-specific applications, though we keep flexibility in this direction as one of our goals.

Diverse QA system architectures have been proposed in the last 15 years, applying different approaches to information retrieval. A full survey is beyong the scope of this paper, but let us outline at least the most basic choices we faced when designing our system.

Perhaps the most popular approach in QA research has been restricting the task to querying structured knowledge bases, typically using the RDF paradigm and accessible via SPARQL. The QA problem can be then rephrased as learning a function translating free-text user query to a structured lambda expression or SPARQL query. [3] [5] We prefer to focus on unstructured datasets as the coverage of the system as well as domain versatility increases dramatically; building a combined portfolio of structured and unstructured knowledge bases is then again an obvious extension.

When relying on unstructured knowledge bases, a common strategy is to offload the information retrieval on an external high-quality web search engine like Google or Bing (see e.g. the **Mulder** system [15] or many others). We make a point of relying purely on local information sources. While the task becomes noticeably harder, we believe the result is a more universal system that could be readily refocused on a specific domain or proprietary knowledge base, and also a system more appropriate as a scientific platform as the results are fully reproducible over time.

Finally, a common restriction of the QA problem concerns only selecting the most relevant answer-bearing passage, given a tuple of input question and set of pre-selected candidate passages [23]. This Answer Sentence Selection task is certainly worthwhile as a component of a QA system but does not form a full-fledged system by itself. It may be argued that returning a whole passage is more useful for the user than a direct narrow answer, but this precludes any reasoning or other indirect answer synthesis on the part of the system, while the context and supporting evidence can be still provided by the user interface. Direct answer output may be also used in a more general AI reasoning engine.

In this paper, we present our open source Question Answering system **brmson™ YodaQA**. This is not the only open source QA framework currently available, but we found our goals not entirely compatible with any of the other systems we investigated. Specifically, we aim to build a system that (A) provides an end-to-end, universal pipeline integrating different knowledge base paradigms in a modular fashion; (B) is domain flexible and could cater even to the long tail of rarer question subjects, therefore has minimum of fixed categories and hand-coded rules.

In contrast, the classic QA system **OpenEphyra** [21] operates on the basis of fixed question categories with handcrafted rules, and puts emphasis on querying web search engines. The **OAQA** initiative [12] has developed a basic QA framework, but does not provide an end-to-end pipeline and its usage of UIMA has in our opinion severe design limitations (see below). The **WatsonSim** system [13] has begun developing independently during the course of our own work and it works on Jeopardy! statements rather than questions.

Jacana [24] [25] is a promising set of loosely coupled QA-related methods and algorithms, focused on machine learning of textual entailment. It is not meant to be a full QA framework and using it as an end-to-end pipeline is not straightforward, but integration of the Jacana implementation as modules in YodaQA is our long-term plan.

OpenQA [1] is a recently introduced end-to-end QA pipeline platform also developed independently during the course of our work, and shares our goal of a common research platform in the field. However, the approach is very different, as OpenQA is more of a portfolio-style engine with mostly independent pipelines which have their candidate answers combined, while YodaQA emphasizes modularity on the pipeline stage level, with e.g. all answer producers sharing a common answer analysis stage.

The rest of the paper is structured as follows. We outline the general structure of our framework in Sec. 2. We then describe the current reference implementation of the pipeline components in Sec. 3. We propose a common experimental setup and analyze the system performance in Sec. 4. We conclude with a summary of our contributions and an outline of future extensions in Sec. 5.

2. YodaQA Pipeline Architecture

The goals for our system **brmson™ YodaQA** are to provide an open source Question Answering platform that can serve both as scientific research testbed and a practical system. The pipeline is implemented mainly in Java, using the popular Apache UIMA framework [11]. Extensive support tooling is included within the package.

Unlike OAQA, in YodaQA each artifact (question, search result, passage, candidate answer) is represented as a separate UIMA CAS, allowing easy parallelization and easy leverage of pre-existing NLP UIMA components; we also put emphasis on aggressive compartmentalization of different tasks to interchangeable annotators rather than using UIMA just for high level flow and annotation storage.

The framework is split in several Java packages: **io** package takes care of retrieving questions and returning answers, **pipeline** contains classes of the general pipeline stages, **analysis** contains modules for various analysis steps, **provider** has interfaces to various external resources and **flow** carries UIMA helper classes.

The system maps an input question to ordered list of answer candidates in a pipeline fashion, with the flow as in Fig. 1, encompassing the following stages:

- **Question Analysis** extracts natural language features from the question text and produces QA features based on them (clues, type, etc.).
- **Answer Production** generates a set of candidate answers based on the question. Typically, this happens 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 these (the **Passage Extraction**) and generating candidate answers from picked passages (the **Passage Analysis**).
- **Answer Analysis** generates various answer features based on detailed analysis (most importantly, type determination and coercion to question type).
- **Answer Merging and Scoring** consolidates the set of answers, removing duplicates and using a machine learned classifier to score answers by their features.
- **Successive Refining** (optional) prunes the set of questions in multiple phases, interjecting some extra tasks (evidence diffusion and gathering additional evidence).

The basic pipeline flow is much inspired by the DeepQA model of IBM Watson [9]. Throughout the flow, answer features are gradually accumulated and some results of early flow stages (especially the question analysis) are carried through the rest of the flow.

3. YodaQA Reference Pipeline

The particular Question Answering problem considered in the reference pipeline is finding a precise (narrowly phrased) answer to a naturally phrased English question, based on both unstructured (English Wikipedia, *enwiki*) and structured (DBpedia [17], Freebase [4]) knowledge bases.

In our pipeline, we build on existing third-party NLP analysis tools, in particular Stanford CoreNLP (Segmenter, POS-Tagger, Parser) [18] [6], OpenNLP (Segmenter, NER)

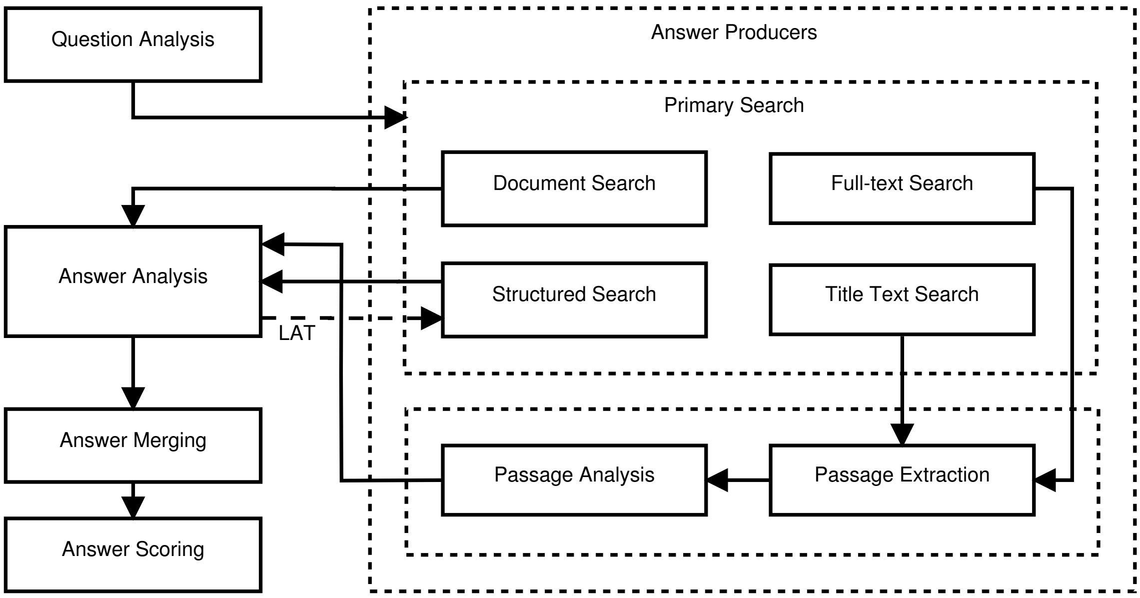

Fig.1. The general architecture of the YodaQA pipeline. Present but unused final pipeline portions not shown.

[2] and LanguageTool (Segmenter, Lemmatizer).[1 2] NLP analysis backends are freely interchangeable thanks to the DKPro UIMA interface [8]. For semantic analysis, we also rely heavily on the WordNet lexicon [19].

For a practical illustration of the pipeline processing two particular example questions, please follow Fig. 4 and 5.

3.1. Question Analysis

During question analysis, produce a part-of-speech tagging and dependency parse of the question text, recognize named entities and, roughly inspired by the DeepQA system [16], heuristically generate several QA features: Clues, Focus, and LAT.

Clues represent keywords in the question that determine its content and are used to query for candidate answers. Clues based on different question components are assigned different weight (used in search retrieval and passage extraction, determined empirically) — in ascending other, all noun phrases, noun tokens and the selection verb (SV); the LAT (see below); named entities; the question sentence subject (determined by dependency parse). If the clue text corresponds to an *enwiki* article name or redirect alias, its weight is boosted and it is flagged as a **concept clue**.

Focus is the center point of the question sentence indicating the queried object. Six simple hand-crafted heuristics extract the focus based on the dependency parse. "name of —" constructions are traversed.

LAT (Lexical Answer Type) describes a type of the answer that would fit the question. This type is not of a predefined category but may be an arbitrary English noun, like in the DeepQA system. [20] The LAT is derived from the focus, except question words are mapped to nouns ("where" to "location", etc.) and adverbs (like "hot") are nominalized (to "temperature") using WordNet relations.

3.2. Unstructured Answer Sources

The primary source of answers in our QA system is keyword search in free-text knowledge base, in our default setting the *enwiki*. While the information has no formal structure, we take advantage of the organization of the *enwiki* corpus where entity descriptions are stored in articles that bear the entity name as title and the first sentence is typically an informative short description of the entity. Our search strategies are analogous to basic DeepQA free-text information retrieval methods [7]. We use the Apache Solr[3] search engine (frontend to Apache Lucene).

Title-in-clue search [7] looks for the question clues in the article titles, essentially aiming to find articles that describe the concepts touched in the question. The first sentence of the top six articles (which we assume is its summary) is then used in passage analysis (see below).

Full-text search [7] runs a full-text clue search in the article texts and titles, considering the top six results. The document texts are split to sentences which are treated as separate passages and scored based on sum of weights of

[1] `http://www.languagetool.org/`

[2] Sometimes, different pipeline components default to different NLP backends to perform the same task, e.g. segmentation, based on empirically determined best fit.

[3] `http://lucene.apache.org/solr/`

clues occuring in each passage[4][5]; the top three passages from each document are picked for passage analysis.

Document search [7] runs a full-text clue search in the article texts; top 20 article hits are then taken as potential responses, represented as candidate answers by their titles.

Concept search retrieves articles whose titles (or redirect aliases) exactly match a question clue. The first sentence and also passages extracted as in the full-text search are used for passage analysis.

Given a picked passage, the **passage analysis** process executes an NLP pipeline and generates candidate answers; currently, the answer extraction strategy entails simply converting all named entities and noun phrases to candidate answers. Also, object constituents in sentences where subject is the question LAT are converted to candidate answers.

3.3. Structured Answer Sources

Aside of full-text search, we also employ structured knowledge bases organized in RDF triples; for each concept clue, we query for predicates with this clue as a subject and generate candidate answers for each object in such a triple, with the predicate label seeded as one LAT of the answer.

In particular, we query the DBpedia `ontology` (curated) and `property` (raw infobox) namespaces and the Freebase RDF dump. For performance reasons, we limit the number of queried Freebase topics to 5 and retrieve only 40 properties per each; due to this limitation, we have manually compiled a blacklist of skipped "spammy" properties based on past system behavior analysis (e.g. location's *people_born_here* or music artist's *track*).

3.4. Answer Analysis

In the answer analysis, the system takes a closer look at the answer snippet and generates numerous features for each answer. The dominant task here is type coercion, i.e. checking whether the answer type matches the question LAT.

The answer LAT is produced by multiple strategies:

- Answers generated by a named entity recognizer have LAT corresponding to the triggering model; we use stock OpenNLP NER models *date*, *location*, *money*, *organization*, *percentage*, *person* and *time*.
- Answers containing a number have a generic *quantity* LAT generated.
- Answer focuses (the parse tree roots) are looked up in WordNet and *instance-of* pairs are used to generate LATs (e.g. *Einstein* is *instance-of scientist*).
- Answer focuses are looked up in DBpedia and its ontology is used to generate LATs.
- Answers originating from a structured knowledge base carry the property name as an LAT.

Type coercion between question and answer LATs is performed using the WordNet hypernymy relation — i.e. *scientist* may be generalized to *person*, or *length* to *quantity*. We term the type coercion score **WordNet specificity** and exponentially decrease it with the number of hypernymy traversals required. Answer LATs coming from named entity recognizer and quantity are not generalized. We never generalize further once within the `noun.Tops` WordNet domain and based on past behavior analysis, we have manually compiled a further blacklist of WordNet synsets that are never accepted as coercing generalizations (e.g. *trait* or *social group*).

The generated features describe the origin of the answer (data source, search result score, clues of which type matched in the passage, distance-based score of adjacent clue occurences, etc.), syntactic overlaps with question clues and type coercion scores (what kind of LATs have been generated, if any type coercion succeeded, what is the WordNet specificity and whether either LAT had to be generalized).

3.5. Answer Merge-and-Score

The merging and scoring process also basically follows a simplified DeepQA approach [14]. Candidate answers of the same text (up to basic normalization, like *the-* removal) are merged; element-wise maximum is taken as the resulting answer feature vector (except for the `#occurences` feature, where a sum is taken). To reduce overfitting, too rare features are excluded (when they occur in less than 1% questions and 0.1% answers).

Supplementary features are produced for each logical feature — aside of the original value, a binary feature denoting whether a feature has *not* been generated and a value normalized over the full set of answers so that the distribution of the feature values over the answer has mean 0 and standard deviation 1. The extended feature vectors are converted to a score $s \in [0, 1]$ using a logistic regression classifier. The weight vector is trained on the gold standard of a training dataset, employing L2 regularization objective. To strike a good precision-recall balance, positive answers (which are about $p = 0.03$ portion of the total) are weighed by $0.5/p$.

3.6. Successive Refining

The pipeline contains support for additional refining and scoring phases. By default, after initial answer scoring, only the top 25 answers are kept with the intent of reducing noise for the next answer scoring classifier. Answers are compared and those overlapping syntactically (prefix, suffix, or substring aligned with sub-phrase boundaries) are subject

[4] The *about-clues* which occur in the document title have their weight divided by four (as determined empirically).

[5] We also carry an infrastructure for machine learning models scoring candidate passages, but they have not been improving performance so far.

to evidence diffusion where their scores are used as features of the overlapping answers. Another answer scoring would be then performed, and the answer with the highest score is then finally output by the system.[6]

However, while we have found these extra scoring steps beneficial with weaker pipelines (in particular without the clue overlap features), in the final pipeline configuration the re-scoring triggers significant overfitting on the training set and we therefore ignore the successive refining stage in the benchmarked pipeline.

4. Performance Analysis

As we present performance analysis of our system, we shall first detail our experimental setup; this also includes discussion of our question dataset.

Then, we proceed with the actual results — we measure the *recall* of the system (whether a correct answer has been generated and considered, without regard to its score) and *accuracy-at-one* (whether the correct answer has been returned as the top answer by the system). We find this preferrable to typical information retrieval measures like MRR or MAP since in many applications, eventually only the single top answer output by the system matters; however, we also show the *mean reciprocial rank* for each configuration and discuss the rank distribution of correct answers.

Aside of the performance of the default configuration, we also discuss scaling of the system (extending the alotted answer time) and performance impact of its various components (hold-out testing).

4.1. Experimental Setup

Our code is version tracked in a public GitHub repository `https://github.com/brmson/yodaqa`, and the experiments presented here are based on commit `f6c0cf6` (tagged as `v1.0`). The quality of full-text search is co-determined by Solr version (we use 4.6.0) and models of the various NLP components which are brought in by DKPro version 1.7.0. As for the knowledge bases, we use enwiki-20150112, DBpedia 2014, Freebase RDF dump from Jan 11 2015, and WordNet 3.1. Detailed instructions on setting up the same state locally (including download of the particular dump versions and configuration files) are distributed along the source code.

As a benchmark of the system performance, we use a dataset of 430 training and 430 testing open domain factoid questions. (For system development, exclusively questions from the training set are used.) This dataset is based on the public question answering benchmark from the main tasks of the TREC 2001 and 2002 QA tracks with regular expression answer patterns[7] and extended by questions asked to a YodaQA predecessor by internet users via an IRC interface. This dataset was further manually reviewed by the author, ambiguous or outdated questions were removed and the regex patterns were updated based on current data. We refer to the resulting 867 question dataset as `curated` and randomly split it to the training and testing sets.[8]

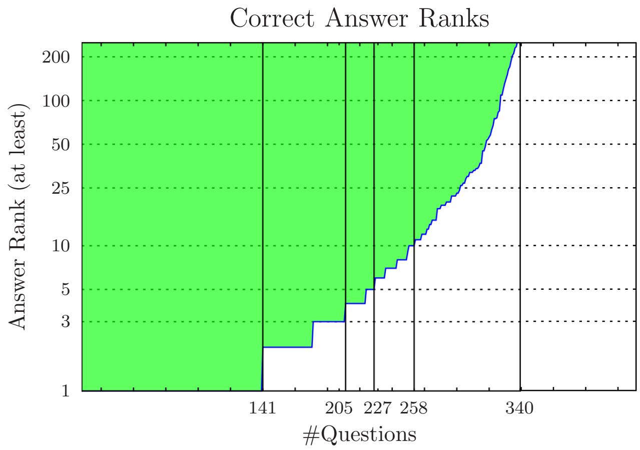

Fig. 3. Number of questions x that output the correct answer ranked at least y.

An automatic benchmark evaluation system is distributed as part of the YodaQA software package. The system evaluates the training and test questions in parallel and re-trains the machine learning models before scoring the answers. Therefore, in all the modified system versions considered below, a model trained specifically for that version is used for scoring answers.

Our benchmark is influenced by two sources of noise. First, the answer correctness is determined automatically by matching a predefined regex, but this may yield both false positives and false negatives.[9] Second, during training the models are randomly initialized and therefore their final performance on a testing set flutters a little.

4.2. Benchmark Results

Benchmark results over various pipeline configurations are laid out in Fig. 2. Aside of the general performance of the system, it is instructive to look at the histogram of answer ranks for the default pipeline, shown in Fig. 3. We can observe that while accuracy-at-one is 32.6%, accuracy-at-five is already at 52.7% of test questions.

The information retrieval parameters of the default pipeline are selected so that answering a question takes about

[6] There is also experimental support for additional evidence gathering phase, where the top 5 answers are looked up using the full-text search together with the question clues, and the number and score of hits are used as additional answer features and final answer rescoring is performed. Nevertheless, we have not found this approach effective.

[7] `http://trec.nist.gov/data/qa/2001_qadata/main_task.html` and 2002 analogically.

[8] The remaining 7 questions are left unused for now.

[9] For example numerical quantities with varying formatting and units are notoriously tricky to match by a regular expression.

Pipeline	Recall	Accuracy-at-1	MRR	time
default	79.3%	32.6%	0.420	28.8s
full-text scaling (6 $\rightarrow$ 12 fetched results)	82.3%	34.0%	0.430	50.0s
passage scaling (3 $\rightarrow$ 6 picked passages)	81.2%	31.4%	0.415	43.5s
document search scaling (20 $\rightarrow$ 40)	80.0%	31.6%	0.418	34.9s
freebase scaling (5 $\rightarrow$ 10 topics, 40 $\rightarrow$ 80 properties)	79.8%	31.6%	0.416	29.8s
full-text hold-out	49.5%	20.9%	0.277	5.8s
structured hold-out	73.5%	29.1%	0.376	23.8s
type coercion hold-out	79.3%	22.1%	0.314	30.0s
concept clues hold-out	67.9%	23.0%	0.314	25.6s
clue overlap test hold-out	79.3%	29.5%	0.390	30.1s

Fig.2. Benchmark results of various pipeline variants on the *curated* test dataset. MRR is the Mean Reciprocal Rank $|Q| \cdot \sum_{q\in Q} 1/r_q$, time is the average time spent answering one question.

30s on average on a single core of AMD FX-8350 with 24GB RAM and SSD backed databases. (Note that no computation pre-processing was done on the knowledge bases or datasets; bulk of the time per question is therefore spent querying the search engine and parsing sentences, making it an accurate representation of time spent on previously unseen questions.) By raising the limiting parameters, we can observe further recall increase, and in case of considering more full-text search results, also a solid accuracy improvement. Our system could therefore meaningfully make use of further computing resources.

We also benchmarked performance with various components of the pipeline disabled. We can see that the full-text and structured knowledge bases are complementary to a degree, but the full-text base is eventually a much stronger answer source for our system. Type coercion and detection of the concept clues in the question are both important heuristics for our QA system.

Comparison of performance across multiple systems is currently non-trivial, unfortunately, as there is no universally agreed experimental setup so far and not even published results on the TREC datasets from the years we use are readily available. OpenEphyra seemed to typically achieve accuracy-at-one of "above 25%" on the TREC datasets including our years according to [21]. In the Answer Sentence Selection task [23], Jacana and similar textual entaliement systems are reported[10] to achieve MRR around 0.750 but this task represents a significant restriction upon the general end-to-end QA pipeline.

5. Conclusion

We have described a modular question answering system which can be used for effective mixing of both structured and unstructured knowledge bases, is domain-flexible and highly amenable to further extensions in various stages of its pipeline. We put emphasis on universal, machine learned methods and employ only a very limited amount of hand-crafted heuristics.

Meanwhile, the system is already demonstrating a reasonable open domain factoid question answering performance, being able to answer a third of the testing set questions correctly, over half of the questions in top five answers, and considering the correct answer for just about 80% of the questions; we have also shown a head-room for further performance scaling by extending the available computational resources.

Our system is made available as open source under the highly permissive Apache Software Licence[11] and available for research collaboration on the GitHub social software hosting site. We hope for our system to become an universal research platform for testing of various question answering related strategies not only in isolation but also measuring their effect in a real-world high performance end-to-end pipeline. We also hope our work helps to clarify which of the numerous DeepQA contributions are most essential to a minimal working modern QA system.

5.1. Future Work

We present here just the first version of a system that could be improved in many desirable ways. The software platform itself would benefit in particular from a multi-threaded pipeline flow driver, a sophisticated user interface showing the generated answer in context and hypertext, and sped up benchmarking by caching information retrieval results (parsed picked passages) across runs.

Regarding algorithmic improvements, the most obvious candidates seem a more sophisticated answer extraction strategy (e.g. employing methods introduced in Jacana seems as a natural fit) and more reliable type coercion as a negative evidence source; we also hope that distributed rep-

[10] See the **ACL Wiki** topic *Question Answering (State of the art)*.

[11] The GPL licence applies in case Stanford CoreNLP components are employed.

resentations might improve both areas. We feel that without further large effort in feature engineering, logistic regression is inadequate for scoring answers and we are seeing promising preliminary results from employing random forests instead.

Analysis and model training would be improved with larger benchmark datasets with more sophisticated correct answer verification. Some sub-tasks like type coercion would benefit from specialized datasets, and passage extraction scoring could be tuned on the Answer Sentence Selection dataset.

A robust heuristic for additional evidencing of most promising answers remains as an open problem in our system. While the natural idea of additional fulltext search combining the question and answer has been beneficial with a less sophisticated pipeline, it does not improve performance with our current featureful pipeline.

Acknowledgements

This work was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS14/194/OHK3/3T/13. The research was co-supervised by Dr. Jan Šedivý and Dr. Petr Pošík.

References

[1] openQA: Open source question answering framework.

[2] BALDRIDGE, J. The OpenNLP project. `http://opennlp.apache.org/`.

[3] BERANT, J., CHOU, A., FROSTIG, R., AND LIANG, P. Semantic parsing on freebase from question-answer pairs. In *EMNLP* (2013), pp. 1533–1544.

[4] BOLLACKER, K., EVANS, C., PARITOSH, P., STURGE, T., AND TAYLOR, J. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data* (2008), ACM, pp. 1247–1250.

[5] BORDES, A., CHOPRA, S., AND WESTON, J. Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676* (2014).

[6] CHEN, D., AND MANNING, C. D. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2014), pp. 740–750.

[7] CHU-CARROLL, J., FAN, J., BOGURAEV, B., CARMEL, D., SHEINWALD, D., AND WELTY, C. Finding needles in the haystack: Search and candidate generation. *IBM Journal of Research and Development 56*, 3.4 (2012), 6–1.

[8] DE CASTILHO, R. E., AND GUREVYCH, I. A broad-coverage collection of portable nlp components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT (OIAF4HLT) at COLING 2014* (Dublin, Ireland, Aug. 2014), N. Ide and J. Grivolla, Eds., Association for Computational Linguistics and Dublin City University, pp. 1–11.

[9] EPSTEIN, E. A., SCHOR, M. I., IYER, B., LALLY, A., BROWN, E. W., AND CWIKLIK, J. Making watson fast. *IBM Journal of Research and Development 56*, 3.4 (2012), 15–1.

[10] 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.

[11] FERRUCCI, D., AND LALLY, A. UIMA: An architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng. 10*, 3-4 (Sept. 2004), 327–348.

[12] FERRUCCI, D., NYBERG, E., ALLAN, J., BARKER, K., BROWN, E., CHU-CARROLL, J., CICCOLO, A., DUBOUE, P., FAN, J., GONDEK, D., ET AL. Towards the open advancement of question answer systems. Tech. rep., IBM Technical Report RC24789, Yorktown Heights, NY, 2009.

[13] GALLAGHER, S., ZADROZNY, W., SHALABY, W., AND AVADHANI, A. Watsonsim: Overview of a question answering engine. *arXiv preprint arXiv:1412.0879* (2014).

[14] GONDEK, D., LALLY, A., KALYANPUR, A., MURDOCK, J. W., DUBOUÉ, P. A., ZHANG, L., PAN, Y., QIU, Z., AND WELTY, C. A framework for merging and ranking of answers in deepqa. *IBM Journal of Research and Development 56*, 3.4 (2012), 14–1.

[15] KWOK, C., ETZIONI, O., AND WELD, D. S. Scaling question answering to the web. *ACM Transactions on Information Systems (TOIS) 19*, 3 (2001), 242–262.

[16] LALLY, A., PRAGER, J. M., MCCORD, M. C., BOGURAEV, B., PATWARDHAN, S., FAN, J., FODOR, P., AND CHU-CARROLL, J. Question analysis: How watson reads a clue. *IBM Journal of Research and Development 56*, 3.4 (2012), 2–1.

[17] LEHMANN, J., ISELE, R., JAKOB, M., JENTZSCH, A., KONTOKOSTAS, D., MENDES, P. N., HELLMANN, S., MORSEY, M., VAN KLEEF, P., AUER, S., ET AL. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web* (2014).

[18] MANNING, C. D., SURDEANU, M., BAUER, J., FINKEL, J., BETHARD, S. J., AND MCCLOSKY, D. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (2014), pp. 55–60.

[19] MILLER, G. A. WordNet: a lexical database for english. *Communications of the ACM 38*, 11 (1995), 39–41.

[20] MURDOCK, J. W., KALYANPUR, A., WELTY, C., FAN, J., FERRUCCI, D. A., GONDEK, D., ZHANG, L., AND KANAYAMA, H. Typing candidate answers using type coercion. *IBM Journal of Research and Development 56*, 3.4 (2012), 7–1.

[21] SCHLAEFER, N., GIESELMANN, P., SCHAAF, T., AND WAIBEL, A. A pattern learning approach to question answering within the ephyra framework. In *Text, speech and dialogue* (2006), Springer, pp. 687–694.

[22] SINGHAL, A. Introducing the knowledge graph: things, not strings. *Official Google Blog, May* (2012).

[23] WANG, M., SMITH, N. A., AND MITAMURA, T. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL* (2007), vol. 7, pp. 22–32.

[24] YAO, X., VAN DURME, B., CALLISON-BURCH, C., AND CLARK, P. Answer extraction as sequence tagging with tree edit distance. In *HLT-NAACL* (2013), Citeseer, pp. 858–867.

[25] YAO, X., VAN DURME, B., AND CLARK, P. Automatic coupling of answer extraction and information retrieval. In *ACL (2)* (2013), Citeseer, pp. 159–165.

About Author…

Petr Baudiš has received his Masters degree in Theoretical Computer Science at the Charles University in Prague. So far, he has been working chiefly in the fields of Computer Go and Continuous Black-box Optimization. Currently, he is a PhD student at the Czech Technical University and his interests are extending also to the topics related to information extraction from unstructured text corpora.

Question Text	Who wrote Ender's Game?
Q. Analysis	**Focus:** who; **SV:** wrote; **LAT:** person
Question Clues	Ender's Game **(concept clue)**, wrote
Primary Search (DBpOnt.)	**author:** Orson Scott Card, **publisher:** Tor Books, ...
Primary Search (DBpProp.)	(ibid), **cover artist:** John Harris, **Caption:** 2005, ...
Primary Search (Freebase)	**Author** Orson Scott Card, **Characters** Valentine Wiggin, Hive Queen, ...
Primary Search (concepts)	*enwiki* Ender's Game
Primary Search (fulltext)	**Query:** `+("wrote" OR titleText: "wrote")^1.0 +("ender's Game" OR titleText:"ender's Game")^1.1 ("wrote ender's Game "~4)^2.1...` **Found:** Ender's Game (series), Ender's Game, Ender's Game (film), Ender's Game (comics), Jane (Ender's Game), List of Ender's Game series planets **Sample picked passages:** Elaborating on characters and plot lines depicted in the novel, Card later wrote additional books to form the Ender's Game series. ...Valentine wrote an essay here comparing the priestly class to the skeletons of small vertebrates some time before Speaker for The Dead.
Primary Search (titles)	Ender's Game, List of Ender's Game characters, Jane (Ender's Game), Ender's Game (short story), Ender's Game (film), Ender's Game (series) **Sample first passage:** "Ender's Game" is a 1985 military science fiction novel by American author Orson Scott Card.
Primary Search (document)	Ender's Game (series), Elisabeth Hirsch, Orson Scott Card, Ender in Exile, Worthing Inn, Jane (Ender's Game), ...
Orscon Scott Card	Structured primary search LAT *author* (Wordnet hypernyms *communicator*, *person*, *maker*, *creator*); DBpedia LAT *writer* (Wordnet hypernyms *communicator*, *person*); NER LAT *person* Successful type coercion match**!**, "sharp" (exact specific) match from NER LAT**!** **occurences:** 19**!**, **origins:** document title, concept**!**, first passage, noun phrase, named entity, multiple origins, **other:** adjecent to a concept clue mention, no clue text overlap**!**
Jane	Structured primary search LAT *character* (Wordnet hypernyms *imaginary being*, *creativity*, *person*, *message* and 36 others); NER LAT *person* Successful type coercion match**!**, "sharp" (exact specific) match from NER LAT**!** **occurences:** 4, **origins** document title, first passage, noun phrase, named entity, multiple origins, **other:** no clue text overlap**!**
Final Answers	**Orson Scott Card** (0.99), Neal Shusterman (0.96), Elisabeth Hirsch (0.96), American author Orson Scott Card (0.96), ..., List of Ender's Game series planets (0.94), Gavin Hood (0.94), Print (0.93), Jane (0.91), ...

Fig. 4. A sample of the pipeline process when (correctly) answering a training question. **!** indicates particularly distinguishing features.

Question Text	What is the name of the famous dogsledding race held each year in Alaska?
Q. Analysis	**Focus:** name; **SV:** held; **LAT:** race (by Wordnet hypernym: *contest*, *event*, *biological group*, *canal* and 9 others)
Question Clues	name, Alaska **(concept clues)**, race, held, famous, dogsledding, race, year
Primary Search (DBpOnt.)	**area Total:** 1717854.0, **country:** United States, ...
Primary Search (DBpProp.)	(ibid), **West:** Chukotka, **Income Rank:** 4, ...
Primary Search (Freebase)	**Date Founded** 1959-01-03, **Capital** Juneau, ...
Primary Search (concepts)	*enwiki* Alaska, Name **Sample picked passages:** Various races are held around the state, but the best known is the Iditarod Trail Sled Dog Race, a 1150 mi trail from Anchorage to Nome (although the distance varies from year to year, the official distance is set at 1049 mi). ...Automobiles typically have a binomial name, a "make" (manufacturer) and a "model", in addition to a model year, such as a 2007 Chevrolet Corvette.
Primary Search (fulltext)	**Query:** `("the name of the famous dogsledding race held each year in Alaska" OR titleText:...)^2.7 +("name" OR titleText:"name")^2.6...` **Found:** List of New Hampshire historical markers
Primary Search (titles)	Name of the Year, Danish Sports Name of the Year, List of organisms named after famous people, Alaska!, Alaska, Race of a Thousand Years **Sample first passage:** The 2000 Race of a Thousand Years was an endurance race and the final round of the 2000 American Le Mans Series season.
Primary Search (document)	List of New Hampshire historical markers
The 2000 Race of a Thousand Years	**Focus:** Race; DBpedia LAT *automobile race*, *auto race in australia*, *new year celebration*, *quantity* LAT Successful type coercion match**!**, "sharp" (exact specific) match**!** **occurences:** 1, **origins:** first passage, noun phrase, **other:** adjecent to an LAT clue mention**!**, containing clue text
Iditarod Trail Sled Dog Race	**Focus:** Race; DBpedia LAT *sport*, *sport in alaska*, *alaska*, *winter sport*, *attraction*; (N.B. no *race* LAT) Successful type coercion match, loose match by generalization of *attraction* to *social event***!** **occurences:** 1, **origins** passage by various clues, noun phrase, **other:** suffixed by clue text
Final Answers	The 2000 Race of a Thousand Years (0.97), –01-03 (0.94), List of New Hampshire historical markers (0.93), a binomial name, a "make" (manufacturer) and a "model", in addition to a model year, such as a 2007 Chevrolet Corvette (0.90), **the Iditarod Trail Sled Dog Race** (0.89), Various races (0.83), ...

Fig. 5. A sample of the pipeline process when (not quite correctly) answering a training question. **!** indicates particularly distinguishing features.