Systems and Approaches for Question Answering

Mgr. Petr Baudiš

June 22, 2018

Abstract

I define and explore the task of Factoid Question Answering. This involves understanding a naturally phrased question about some (often open domain) fact, looking this fact up in a knowledge base (structured database or unstructured text corpus), and scoring the candidate answers. I survey the recent scientific work and directions in this field, define some interesting sub-tasks and propose a new dataset. I then present my own baseline QA pipeline YodaQA that combines both structured and unstructured approaches. Based on experiences with this baseline and survey of the field, I propose some scientifically promising lines of further QA research.

Errata: This is a report that was finished in a bit of a rush for a deadline, and has numerous imperfections that I dislike but couldn't have avoided:

- The survey should be considered preliminary, and is further worked on at `https://github.com/brmson/qasurvey`. Many more unstructured, miscellanous and specialized systems should have been described in detail. Entity linking should be surveyed.
- The description of YodaQA v1.1 contains omissions. Gradient-boosted decision forests are currently used for answer scoring instead of logistic regression. Coarse question classification is used to generate extra features for the answer-scoring decision trees. Entity linking involves a fuzzy label lookup.
- YodaQA v1.1 itself is sub-optimal; the MRR of an improved version available at the time of this writing (v1.2) is 0.430 on curated, with precision@1 over 35%. The only change between v1.1 and v1.2 was retraining of the CRF model for answer production from passages.
- YodaQA evaluation on WebQuestions leaves much to be desired. While it uses "only structured sources", these include also DBpedia; restricting this to just Freebase actually improves F_1@1 and MRR, though it hurts AP recall. Also, F_1 (Berant) is badly missing, but this will require some infrastructure changes in YodaQA to handle answer lists in datasets.

Contents

Chapter 1

Introduction

In this report, I would like to propose a doctoral thesis to write and defend at the Czech Technical University on the topic of question answering systems.

This thesis proposal is a little unusual since I have radically changed the topic of my research in the course of my second year of study: from portfolio-based function optimization to question answering systems. Therefore, the bulk of my research results published up to now are not on topic of the proposed thesis; my work on portfolio-based optimization is summarized in App. A.

The current focus of my doctoral research is improving the state-of-art in the field of factoid question answering. My main results so far have been building an extensive question answering system **YodaQA** (Baudiš, 2015b) and proposing a high-quality dataset (Baudiš, 2015a), but I have also for example applied the system to a biological QA domain (Baudiš and Šedivỳ, 2015). This research is primarily supervised by Dr. Jan Šedivý; some of the newest results have been achieved while collaborating with intern students in our group.

1.1 Factoid Question Answering

Let us 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 (Singhal, 2012) or personal assistants like Apple Siri, and with the high profile IBM Watson Jeopardy! matches (Ferrucci et al., 2010) it has became a benchmark of progress in AI research.

My goal is ultimately building a general purpose QA system. Thus, I consider an "open domain" general factoid question answering, rather than domain-specific applications, though keeping flexibility in this direction is certainly worthwhile.

1.2 Task Outline

Diverse QA system architectures have been proposed in the last 15 years, applying different approaches to information retrieval. A full survey follows, for now let me outline at least the most basic choices I faced when designing my system.

First, the restriction to **factoid** questions means the system is essentially an extension of a search engine (rather than deducing facts logically) and should return precisely specified, relatively short text snippets as short answers. Answering happens mainly based on fact lookup. This is in contrast with different question answering tasks (like Language Comprehension Entrance Exams) where the system needs to process a text passage and answer tricky questions about the text meaning (see Sec. 2.5).

Perhaps the most popular approach in factoid 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. (Berant et al., 2013; Bordes et al., 2014) I 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 **AskMSR** system (Brill et al., 2002) or many others). I make a point of relying purely on local information sources.While the task becomes noticeably harder, I believe the outcome 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 (Wang et al., 2007). 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, an idea that I keep in sight within my design though it is clearly out of scope for the thesis I propose.

To summarize, the system I propose should produce short, clear answers based on information retrieval from both unstructured (full-text) and structured (database) knowledge bases, and not rely on any “omniscient web search” to make the task easier specifically in the open domain setting.

1.3 This Thesis Proposal

The rest of this proposal shall focus on building the case for a thesis on the topic of factoid question answering. Chapter 2 surveys the field in detail, examining different formulations of the problem as well as a variety of sub-tasks, reference datasets and approaches, and recent progress in the field. Chapter 3 describes the work I have done on the thesis so far — this revolves mainly around the question answering system YodaQA that I have built, but also progress on some of the QA sub-tasks. Chapter 4 outlines some of the specific problems of QA to focus on in the thesis. I conclude the proposal with a summary in Chapter 5.

Chapter 2

State of the Art in Factoid Question Answering

The research in Question Answering has been proceeding in several, largely independent, directions. The main division is between QA on structured knowledge bases (typically graph databases like the semantic web) and QA on unstructured text corpora (typically Wikipedia, ClueWeb, large amounts of news articles, or even a web search).

Below, we survey the recent activity in these fields. For a historical viewpoint, we refer the reader to Wang (2006), Allam and Haggag (2012) and Lopez et al. (2011).

2.1 Structured Data QA

When answering questions on structured data, we can further consider either helping users access relational databases (Natural Language Interface to Databases; NLIDB) or finding matching relationships in the linked data graph of semantic web (Ontology-based Question Answering; QALD).

We are not aware of any significant recent work in the NLIDB domain that would be based on well defined, published dataset and rigorously evaluate results — most research concerns building auxiliary systems that augment human database experts. (Bergamaschi et al., 2010; Blunschi et al., 2012) We do not consider this branch of research further.

QALD is much richer field of research, with yearly challenges (Unger, 2014) and popular reference datasets like Free917 (Cai and Yates, 2013) and WebQuestions (Berant et al., 2013).[1] The reference knowledge base is typically Freebase (Bollacker et al., 2008), which holds a large amount of open domain real-world facts.[2] QALD is one of the benchmarks for the semantic parsing task (Liang, 2015), aiming to map a naturally phrased question to a logical form (to be in this case executed as a graph query on a knowledge base). (Berant and Liang, 2014; Bordes et al., 2014)

(Bordes et al., 2014) proposed a classification of QALD methods that we adopt, outlining two general paradigms:

Information Retrieval (IR) systems first retrieve a broad set of candidate answers

[1]In the past, closed-domain datasets like GeoQuery were also popular.

[2]Freebase has been phased out into legacy state at this point. WikiData is set to replace it, but the migration of knowledge is still ongoing.

by querying the search API of KBs with a transformation of the question into a valid query and then use fine-grained detection heuristics to identify the exact answer. On the other hand, **Semantic Parsing (SP)** methods focus on the correct interpretation of the meaning of a question by a semantic parsing system. A correct interpretation converts a question into the exact database query that returns the correct answer.

These two strategies have been also compared in detail in Yao et al. (2014) and appear roughly comparable in accuracy.

2.1.1 Dataset and Evaluation

The most popular question dataset is WebQuestions, which contains 3778 training and 2032 testing questions generated automatically by a combination of Google Suggest API and Freebase[3] and with the gold standard answers manually produced by Amazon Mechanical Turk workers.[4] Answers are always Freebase entities — never relations themselves, literals like numerical quantities, metadata like counts, or whole sentences. Sometimes, the correct answers are lists of entities. Extra measurements on customized versions of TREC (see below) and WikiQuestions datasets are also sometimes published.

There is no complete consensus on benchmarking metrics in this task, owing to different strategies in the systems. In the simplest case, a single query is generated for every question and its result is matched with the gold standard (Berant et al., 2013), then we may measure just the *accuracy* (i.e. proportion of correctly answered questions).

However, most systems generate a ranked list of candidate answers. IR list measures are sometimes used (Fader et al., 2013; Bordes et al., 2014): *MRR* (Mean Reciprocial Rank) can capture the average rank the correct answer appears at, while *MAP* (Mean Average Precision) is not as intuitively interpretable, but generalizes even to a scenario with multiple expected correct answers. Aside of that, an option is to consider only correctness of the best answer, introducing measures *precision@1* as the proportion of questions that have a correct answer ranked first (and, if some questions generate no answers, *recall@1* and F_1*@1*).

However, the official WebQuestions evaluation script of (Berant et al., 2013) considers the complete set of returned answers — this fits e.g. (Yao and Van Durme, 2014), where answer correctness is treated as a binary classification problem and answers classified as true are returned (e.g. with logistic regression output larger than 0.5 as proposed in Yao et al. (2014)). We term these *answer precision* and *answer recall.* The most common measure shared by most papers is F_1 (with recall as a proportion of correct answers found). However, even this measure comes in two variations: F_1 *(Berant)* is average of precision/recall harmonic means per question,[5] while F_1 *(Yao)* is harmonic mean of answer precision and recall computed across all questions, which gives an incentive to a system not to answer some questions.

[3]Note that no fixed version of Freebase is used. Since the answers are typically not a subject of temporal drift, it is unlikely that different Freebase snapshots would yield significant differences, but we are not aware of any study confirming this.

[4]The gold standard has noise level of about 5% questions, based on manual error analysis of a Google QA test on a movies sub-sample.

[5]This is about the same as the accuracy in Berant et al. (2013) style systems. When comparing using this measure in ranked-answers systems, a single answer is often forced even if none would be produced for the question otherwise. (Yao et al., 2014)

System	F_1@1	F1 (Berant)	F1 (Yao)
Fader et al. (2014)	—	—	(35.0%)
Yao and Van Durme (2014)	35.4%	33.0%	42.0%
Bordes et al. (2014)	40.4%	39.2%	43.2%
Yao (2015)	—	44.3%	53.5%
Bast and Haussmann (2015)	—	49.4%	—
Berant et al. (2013)	35.7%	35.7%	—
Bao et al. (2014)	37.5%	37.5%	—
Berant and Liang (2014)	39.9%	39.9%	43.0%
Chang and Gao (2015)	—	52.5%	—

Figure 2.1: Benchmark results of structured KB systems on the WebQuestions dataset using Freebase. Values in parenthesis have caveats like subsampling or manual evaluating.

Ultimately, the difference between the two approaches is how strictly are answers to list-based questions evaluated and what policy does a system choose for picking answers to be evaluated (first, all with score above 0.5, etc.). We recommend that to facilitate comparison between systems that make different choices and possibly do not emphasize list performance,[6] future systems report both metrics that give credit to the best rank of at least a single list element (F_1@1, MRR), and metrics that strictly rate the appearance of all list elements and treat all generated answers as a proposed list (F_1 (Berant, Yao), MAP). (Systems which generate an answer to each question will have F_1@1 equal to precision@1 and recall@1.)

2.1.2 Information Retrieval Approach

Open Question Answering Over Curated and Extracted Knowledge Bases (OQA) (Fader et al., 2014) is a complex multi-stage system: the question is paraphrased (using mined operators), parsed to a KB query (by manual templates), the query rewritten (using mined operators again), and executed (on an ensemble of KBs including Freebase). On WebQuestions, F_1 (Yao — probably) is reported at 35%. Results on TREC were also reported, with F_1 29%. However, the answers were evaluated manually. The implementation is open source.

Information Extraction over Structured Data: Question Answering with Freebase (Jacana Freebase) (Yao and Van Durme, 2014) produces a set of fixed dependency-parse based question features. The question entity is linked to a Freebase concept using the Freebase Search API; Freebase concepts neighboring the question concept in the knowledge graph are identified; Freebase features are generated for all properties of each concept, the connecting relation and rank of the connecting relation predicted by a question-relation alignment model. The alignment model is a bag-of-words naive bayes classifier that predicts the probability of a relation (or sub-relation) component based on the question tokens; it is trained on ClueWeb sentences that contain both the question and answer. Finally, composite features are built for all tuples of question and Freebase features and a logistic regression model predicts the most likely Freebase concept to answer. As (Yao et al., 2014) also further clarifies, on WebQuestion+Freebase, this scores F_1 (Yao) 42.0%, with answer precision 38.8% and answer recall 42.0%, and F_1@1 35.4%. Without the alignment model, the system has F_1 (Yao) 36.9%. The implementation is open source.

[6]Below, we show that in unstructured QA, it is common to disregard list-based performance. Therefore, this aspect is important to hybrid systems.

Question Answering with Subgraph Embeddings (Bordes et al., 2014) considers the QA task as a problem of finding a vector embedding of the question and candidate answers such that the score — a scalar product of these embeddings[7] — is highest for the correct answer. Question embedding is produced by averaging word embeddings (as in Sec. 2.6; word embeddings are trained from scratch), the embedding matrix was also trained to produce similar vectors for question paraphrases (based on WikiAnswers). Answers are embedded from high-dimensional binary encoding of entities and relations in the subgraph surrounding the answer and the path between the question and answer. To generate candidate answers, the immediate neighborhood of the question concept is explored, and in addition directional forrays are made based on score of embeddings of particular relations. Aside of WebQuestion training set, automatically produced 2 million questions from ClueWeb were included for training. WebQuestions+Freebase precision@1 is 40.4% and full-result F_1 (Berant) is 39.2% and F_1 (Yao) 43.2%. (In ensemble with Berant and Liang (2014), F_1 is reported as 41.8% (Berant) or 45.7% (Yao).)

Lean Question Answering over Freebase from Scratch (kitt.ai) (Yao, 2015) is a simple approach that uses fuzzy matching of all question bigrams to link entities to Freebase concepts, sorts them in the Freebase Search API order, and predicts Freebase relations linking the concept with the answer by a bag-of-unigrams-and-bigrams logistic regression. On WebQuestions+Freebase, it scores F_1 (Berant) 44.3%, F_1 (Yao) 53.5%. On the training set, this approach had AP recall of 62%.

More Accurate Question Answering on Freebase (Bast and Haussmann, 2015) is an end-to-end system that first aggressively links many potential question entities to Freebase concepts, then generates answer candidates by matching three pre-defined templates in the knowledge (including one that uses entity pairs). Answers are filtered based on trivial type checking, and scored based on Freebase relation(s) alignment with the question — number of overlaping words, derived words, word vector embedding cosine similarities and indicator words in question trained by distant supervision (Wikipedia sentences that contain two entities connected with such relation in Freebase would often have the indicator word on the path between the entities in dependency parse). The answers are ranked using pairwise random forest ranker. On WebQuestions+Freebsae, it scores F_1 (Berant) 49.4%.

2.1.3 Semantic Parsing Approach

The semantic parsing field has too many publications to list them all in detail here. Therefore, we chose to restrict ourselves only to semantic parsers that report results on unabridged WebQuestions dataset, so that we can facilitate easy comparison. Most notably, we omit modern semantic parsers preceding the dataset (Kwiatkowski et al., 2013; Cai and Yates, 2013), parsers with only partial evaluation (Reddy et al., 2014) and parsers using different datasets like WikiAnswers Fader et al. (2013).

Semantic Parsing on Freebase from Question-Answer Pairs (SEMPRE) (Berant et al., 2013) was the semantic parsing system that first introduced the WebQuestions dataset and remains a popular baseline. It builds a lexicon mapping from phrases to predicates by aligning Freebase to ClueWeb and producing co-occurrence based features, then generates additional predicates *(bridging)* based on neighboring predicates. The baseline WebQuestions+Freebase accuracy is 35.7% (as revised in Yao et al. (2014); this number is equivalent to answer precision, recall and F_1 (Berant) as well as these metrics @1 for this system as it generates a single answer

[7]That is, non-normalized cosine similarity of vectors.

(or answer list) hypothesis for each question). The implementation is open source.

Knowledge-Based Question Answering as Machine Translation (Bao et al., 2014) considers converting a question to an answer as a machine translation problem, with the formal query as Meaning Representation and various partial derivations of the question converted and composed to these representations. We do not include details since accuracy is only 37.5% and involves training on several thousand hand-labelled questions.

Semantic Parsing via Paraphrasing (ParaSempre) (Berant and Liang, 2014) extends the work of Fader et al. (2013) (Paralex) on learning a rich lexicon for logical form translation rules based on the WikiAnswers dataset which contains clusters of equivalent but differently phrased questions. In this paper, five query templates are manually defined, along with eight general transformation rules. From the input question, a number of possible logical forms are generated and then scored by a paraphrase-based model, using features based on word associations and features based on averaged word vector embeddings with a generative model for embeddings alignment (similar to Yu et al. (2014) described in detail in Sec. 2.2.2). WebQuestions+Freebase accuracy is reported to be 39.9%. Bordes et al. (2014) reports that F_1 (Yao) is 43.0%.

Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base (STAGG) (Chang and Gao, 2015) produces the logical forms of questions in the form of subgraph templates to be matched against the Freebase knowledge graph, facilitating robust answering of even multiple-concept questions (such as which actor plays character X in Y); the subgraph templates also include aggregation operators. S-MART Yang and Chang (2015) is used for entity linking that has similar recall as Freebase Search API, but twice the precision. The *core inferential chain* is built as a property path between question topic and answer concept within Freebase, and expanded to a subgraph query by iteratively adding extra constraints based on matches of question content in the path neighborhood (entity mentions or aggregation words). Candidate query subgraphs are scored based on vector similarity features; vector embeddings of the question and property paths are produced by a max-pooled CNN (which is pre-trained to produce similar vectors for ClueWeb sentence pairs containing entities that are linked within Freebase). The features are used for ranking candidate query subgraphs using lambda-rank. On WebQuestions+Freebase, the system achieves answer list precision 52.8% and recall 46.6%, with F_1 (Berant) 52.5%.

2.2 Unstructured Data QA

When dealing with question answering on top of unstructured data, two research directions exist: aside of generating a crisp answer, *Answer Sentence Selection* is a popular sub-task where instead of the answer, just an appropriate answer-bearing passage is to be returned to the user.

2.2.1 Dataset and Evaluation

The main benchmarked dataset is a collection of TREC competition datasets from the turn of the century,[8] on top of the AQUAINT newswire corpus; as gold standard, regular expressions matching the answers judged correct were provided. Compared to e.g. WebQuestions + Freebase, the TREC questions are more realistic in open domain user QA setting — they are

[8]`http://trec.nist.gov/data/qamain.html`

diverse in answer type and level of detail and of quite varying difficulty. On the other hand, naive usage of this dataset hits a few major issues:

- The AQUAINT corpus is not freely available, and some of the questions are dependent on that time period (like population numbers or persons occupying public positions) or are highly specific (referring particular individual news articles).
- Regular expressions are limited when matching alternative spellings or name abbreviations, and almost break down when matching entities such as dates or dimensional quantities (possibly using different units).
- There are no well-defined train/test splits, datasets for individual years significantly vary in character, and even the years included in summary TREC datasets differ.

Therefore, measurements on TREC are very difficult to compare. Many papers declare that they subsampled the dataset in arbitrary ways or judged correct answers manually. We attempt to rectify this situation for future comparisons in Sec. 3.3.1.

The typical IR metrics of *MRR* and *MAP* are commonly used, as well as *precision*, *recall* and F_1. Notably, since TREC questions are typically single-answer instead of list-based, we consider a question answered correctly if the top-ranked answer is correct: then, precision is the proportion of correctly answered questions to all answered questions,[9] while recall would be the proportion of correctly answered questions to all questions altogether — this definition is shared by e.g. Yao et al. (2013b) and Sun et al. (2015). Ferrucci et al. (2010) uses *precision@70%* to denote a proportion of questions that are answered correctly when attempting to answer 70% of questions.[10]

TREC competitions used MRR (Voorhees et al., 1999; Voorhees and Tice, 2000, 2001), confidence-weighted score $1/Q\sum_{i=1}^{Q}$ precision-at-i/i (Voorhees, 2002), as well as requiring only single answer returned and computing simple accuracy (Voorhees, 2003).

Many unstructured QA systems are primarily IR machines that follow an overproduce-and-choose answer production strategy. To measure the answer production performance, Chu-Carroll et al. (2012) uses *candidate binary recall*, that is the number of questions where a correct answer has been generated as a candidate for scoring; we also use this measure, but prefer the term *answer production (AP) recall*.

2.2.2 Answer Sentence Selection

The task of answer sentence selection aims, given a set of passages, to identify those that answer the question — i.e. bear the answer, but ideally also in the proper context. This may be regarded as a binary classification or ranking problem, and the IR measures MRR (when we just want to position such a sentence as high as possible) and MAP (when we seek to identify all answer bearing sentences) are appropriate. A reference dataset based on the TREC questions has been established by Wang et al. (2007).

In our work, we prefer to focus on returning crisp answers. Even in such a system, a component performing answer sentence ranking is useful as part of the IR chain and feature provider, nevertheless we do not go into much detail in this survey, and instead refer the

[9]Except some outliers; Schlaefer et al. (2006) uses the "precision" term to denote AP recall.

[10]The original papers do not use the percent sign, which we introduce to distinguish from *prec@1* (Bordes et al., 2014).

reader to `http://aclweb.org/aclwiki/index.php?title=Question_Answering_%28State_of_the_art%29` which contains an overview of various approaches. We just pick two example systems below, the second one being state-of-art.

Answer Extraction as Sequence Tagging with Tree Edit Distance (Jacana) (Yao et al., 2013b) uses logistic regression for binary classification of relevant sentences. The core features are determined by aligning parse trees of the question and each candidate sentence and building a tree edit script. TREC MAP 0.631, MRR 0.748.

Deep Learning for Answer Sentence Selection (Yu et al., 2014) estimates, given vector embeddings $\mathbf{q}, \mathbf{a}$, $P(rel|q,a) = \sigma(\mathbf{q}^T \mathbf{M}\,\mathbf{a} + b)$ i.e. trains a model with parameters $\mathbf{M}, b$ that generates a likely question embedding using $\mathbf{M}\,\mathbf{a}$ and then using dot-product measures its distance to the posed question. Cross entropy over all QA pairs is used as the loss function for training. Off-the-shelf $d = 50$ distributed representations are used as word embeddings. To generate compositional embeddings, a simple unigram model that averages the embeddings is the baseline; as a small ($\Delta 0.02$) improvement, bigram model is proposed that uses a CNN on the sentence with a bigram convolutional layer and an average-pooling layer. To deal with numbers and proper nouns, a token co-occurrence counter feature is also used; the learnt method gives $\Delta 0.125$ against this baseline. TREC MAP 0.7113, MRR 0.7846.

2.2.3 Precise Answer Production

In the era when the classic TREC QA track was active, systems with large amount of handcraft Harabagiu et al. (2003) or wrapping a web search engine Brill et al. (2002) dominated (making results difficult or laborious to reproduce, and with sometimes limited scientific value). Let us cover at least a few most interesting recent systems, even though straight evaluation comparison is not possible.

First, the classic QA system **OpenEphyra** (Schlaefer et al., 2006) is the best known open source QA system. It operates on the basis of fixed question categories with hand-crafted rules, and puts emphasis on querying web search engines.

Automatic Coupling of Answer Extraction and Information Retrieval (Jacana) (Yao et al., 2013a) couples the answer sentence selection and extraction of Yao et al. (2013b) described above with IR query mechanism that constructs queries which prefer results that would match highest-weighed QA features in the followup pipeline stages. F_1 on MIT99 (200-question subset of TREC) is 23.1%

Open Domain Question Answering via Semantic Enrichment (QuASE) (Sun et al., 2015) is a web-based QA system that links question entities to Freebase concepts to generate extra features for answers. One feature is the cosine similarity of word vectors corresponding to question (and web-fetched question support) and answer's textual propreties (like *description*) in Freebase (N.B. word frequency vectors, not distributed representations). Another feature is probabilistic type matching by bag of words Bayes model with Perplexity. Generative mixture model with Dirichlet priors is used for answer scoring. Compared to naive web search baseline, TREC F1 improvement was 3.5%. (Absolute numbers are not comparable due to sub-sampling and re-evaluation.)

2.2.4 Non-TREC QA

Of course, question answering may diverge from the standard TREC setting of brief answers to short English language questions. There are no well-established datasets, so we do not survey this in detail, but we still mention at least two examples.

Jeopardy! statements are short declarative sentences that substitute the entity in question with its type. The task of identifying the entities has gained fame as tackled by the **IBM Watson DeepQA** (Ferrucci et al., 2010; Ferrucci, 2012). It uses a pipeline architecture leveraging a wide array of strategies (including information extraction to build some structured KBs (Wang et al., 2012)) and large manual rulesets. However, it also pioneered important conceptual improvements:

- Specific structure and titling of Wikipedia articles is leveraged as high quality data source. (Chu-Carroll et al., 2012)
- Aside of having a fixed taxonomy for question/answer types, these are described by Lexical Answer Types (LATs) as arbitrary English words and flexible Type Coercion methods determine answer compatibility. (Murdock et al., 2012)
- The answers are evaluated in multiple successively focused phases with additional expensive evidence gathered only for top answers. (Gondek et al., 2012)

Another QA system on Jeopardy! questions is **WatsonSim** (Gallagher et al., 2014) (open source).

Another QA-like task is the Quiz Bowl problem, where a sequence of declarative sentences describe some entity, starting from obscure facts and progressing to more obvious identifications. **QANTA** (Iyyer et al., 2014) pioneered usage of TreeRNN-based compound vector embedding models to align vector embeddings of the entities and the sentences describing them.

We must also mention that modern personal assistants like Siri and Cortana include some form of question answering. Many of the common questions in TREC and WebQuestions are also answerable by the Wolfram Alpha engine, as well as the QA component of Google Search (where an actual answer appears, not just relevant search results). Performance of these systems on any well known dataset has not been published to our knowledge, though.

2.3 Auxiliary Tasks in QA

2.3.1 Question Classification

In datasets which ask mostly uniform types of answers (e.g. WebQuestions, where the answer is always an entity), it may be possible to use the same set of features to produce and score answers across all questions. However, e.g. in the TREC dataset, types of answers vary widely and different strategies might be appropriate (even if they are to be machine learned rather than hardcoded, as was common in the early systems).

One approach is to classify an answer to a fixed set of categories. A two-level (coarse, fine) taxonomy based on the TREC set of questions and a labelled dataset[11] was introduced by **Learning question classifiers** (Li and Roth, 2002). They set a baseline of coarse $P_1 = 91.0\%$, fine $P_1 = 84.2\%$.

This sentence classification problem has been one of the standard benchmarks for semantic NLP tasks; a few systems are listed in Fig. 2.2. (Zhao et al., 2015) has more references; note that even primitive baseline would have performance 85%.

[11]`http://cogcomp.cs.illinois.edu/Data/QA/QC/`

Approach	Coarse P_1
SVM$_S$ (Silva et al., 2011)	95.0%
DCNN (Kalchbrenner et al., 2014)	93.0%
CNN (Kim, 2014)	93.6%
Skip-Thought (Kiros et al., 2015)	92.2%
AdaSent (Zhao et al., 2015)	92.6%

Figure 2.2: Selected evaluation results for question classification.

A different approach that was introduced by IBM Watson DeepQA (Murdock et al., 2012) associates the question with a Lexical Answer Type (LAT) which is an arbitrary English word that would describe the answer concept (e.g. "inventor" or "length").

2.3.2 Answers by Paraphrasing

Many questions ask essentially for a paraphrase of the term under question — for example, the question "Who is a plumber?" wants to find out a description of *plumber* without using these words, while "What is the capital of Christians?" may seek the paraphrase of *capital of Christians*, aside of database lookups.

Learning to Understand Phrases by Embedding the Dictionary (DefGen) (Hill et al., 2015) is solving the tasks of reverse dictionary and QA on crossword puzzles using word2vec with composing via RNN or an averaging baseline, embeddings of concepts are pretrained from wikipedia intros and wordnet definitions.

2.4 Hybrid QA Systems

Full-scale end-to-end systems combine structured and unstructured knowledge bases and combine a variety of approaches.

DeepQA IBM Watson (Ferrucci et al., 2010) uses a common pipeline for processing questions, in particular with common question analysis and answer analysis and scoring components, only using the different paradigms in answer production stage. **YodaQA: A Modular Question Answering System Pipeline** (Baudiš, 2015b) which is also described in Ch. 3 largely follows these ideas, reimplementing them within an open source framework from scratch.

OpenQA (Marx, 2014) is conversely more of a portfolio-style engine with mostly independent pipelines which have their candidate answers combined, rather than emphasizing modularity on the pipeline stage level (with e.g. all answer producers sharing a common answer analysis stage) as YodaQA does.

2.5 Non-Factoid QA

Some questions are not meant to be answered by factoids. One special case are yes/no questions — but these can be transformed to similar queries as factoid questions if they have factoid content (e.g. whether person X was born at place Y).

However, in general the prevailing non-factoid scenario is that the system is given a text snippet and questions about the specific content of the text snippet. Examples include entrance exams,[12] TERENCE child therapy snippets[13] or the Toy Tasks set (Weston et al., 2015).Instead of knowledge base retrieval, the system has to infer the answer from the text snippet. These tasks are typically solved by systems for Recognizing Textual Entailment[14] or, lately, Memory Neural Networks (Weston et al., 2014).

As another example, in the Cloze procedure scenario (Taylor, 1953), we consider a $(context, query)$ pair where the query sentence is entailed by the context, but a single entity name in the query is missing (e.g. in a context detailing an incident of the well-known personality *Jeremy Clarkson*, we are to find X for the query *Producer X will not press charges against Jeremy Clarkson, his lawyer says.*). **Teaching Machines to Read and Comprehend** (Hermann et al., 2015) is an attention-based model that produces vector embeddings of $(document, query)$ pairs and uses a weight matrix to judge probability of a particular word being the answer based on the composite pair embedding; the paper is not clear on the particulars, unfortunately. Three composite vector embedding models are considered, based on bi-directional LSTM and convolution-ish architectures.

2.6 Vector Embeddings of Words

Recent progress in NLP has been marked mainly by the proliferation of so-called *vector embeddings*, the most popular being called *word2vec*. (Mikolov et al., 2013b) This approach stems from the so-called *distributional semantics* hypothesis, which posits that we can derive meanings of words purely from the context they tend to appear in. Therefore, each word is associated with a list of n real numbers (i.e. coordinates of an n-dimensional vector) and these numbers are derived automatically just from the context the words appear in.[15]

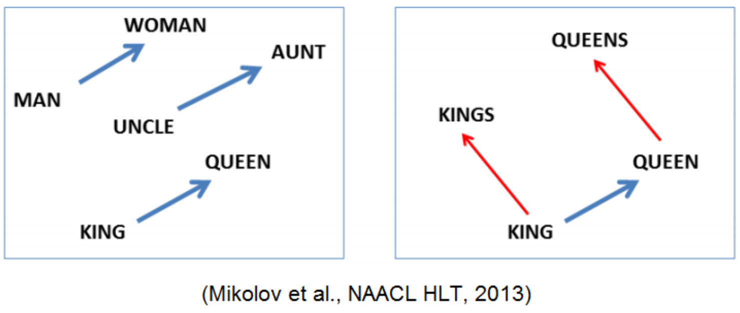

Figure 2.3: Semantic relationships between words as arrows in the vector space. (Mikolov et al., 2013c)

A popularly shown interesting property is that the automatically assigned numbers exhibit semantic properties in how they relate between words. For example, if we do arithmetics on these word vectors and try to compute e.g. $king + (woman - man)$, the nearest vector we reach is *queen*, i.e. the gender transition is represented by an arrow in our vector space

[12] http://nlp.uned.es/entrance-exams/

[13] http://datahub.io/dataset/terence-reading-comprehension-dataset

[14] http://www.aclweb.org/aclwiki/index.php?title=Recognizing_Textual_Entailment

[15] The *word2vec* method uses *multi-task learning* — the n real numbers come out from training a classifier that predicts the most likely next words to come given a context of (say, 100) preceding words; this so-called *language model* task is useful e.g. in speech recognition or OCR.

(Fig. 2.3). Fig. 2.4 shows how relationships between adjectives are represented while Fig. 2.5 shows mappings between coordinates of countries and their capitals. Let us emphasize again that these coordinates were determined purely based on the context of the words (in Wikipedia or millions of news articles); the system did not have any extra information or databases available.

A lot of the current research focuses on the best ways to build up vector representations of whole sentences and documents — ranging from simple averaging (Kim, 2014; Hill et al., 2015) (successful baselines) to recurrent neural networks (Bahdanau et al., 2014; Vinyals et al., 2014). Many applications that rely on semantic understanding of word nuances are popping up; this method became state-of-art for machine translation (Bahdanau et al., 2014), automatic image captioning (Vinyals et al., 2014) and specific types of question answering (Iyyer et al., 2014; Hill et al., 2015; Hermann et al., 2015)[16]. One open problem is efficient composition of vector embeddings for common words with entities like numbers or proper names, or for complex sentences which might require segmentation.

[16]The demo at `http://45.55.181.170/defgen/` is nice.

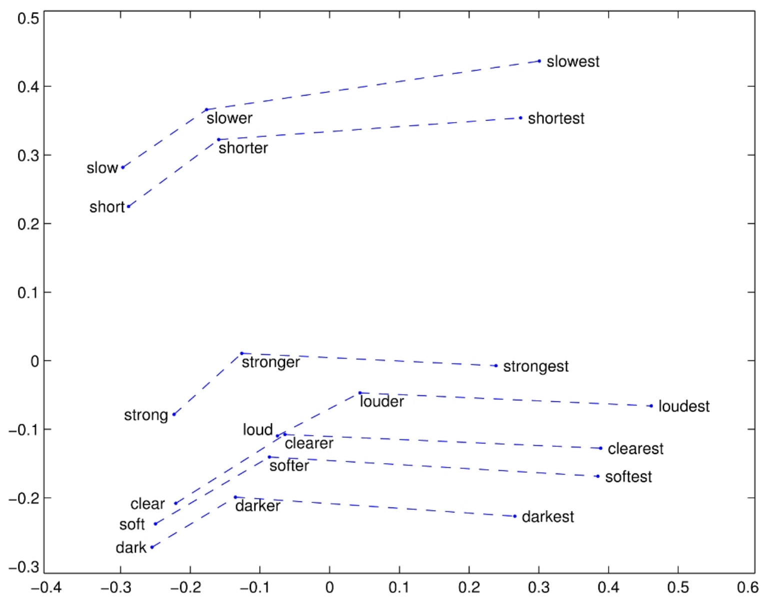

Figure 2.4: Semantic relationships between superlative adjectives as represented in the vector space. This is a 2D projection of the high-dimensional space that is designed to well preserve relative positions of the shown entities (so-called t-SNE 2D). (Pennington et al., 2014)

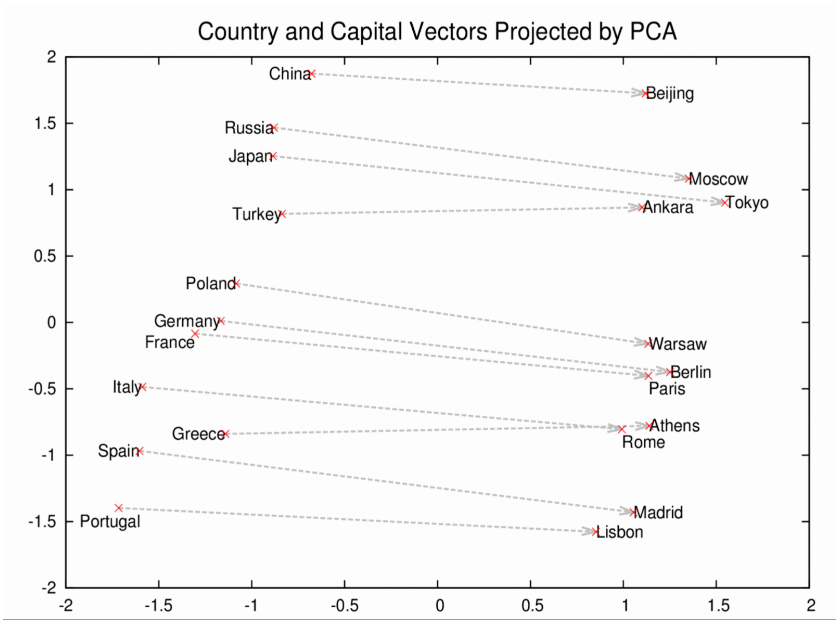

Figure 2.5: Mappings between countries and capitals, acquired entirely from the typical context of the respective words. This is again a 2D projection of the high-dimensional space, this time obtained by a PCA dimensionality reduction technique. (Mikolov et al., 2013a)

Chapter 3

The YodaQA System

My work on Question Answering so far has focused largely on building a new open source QA platform that is sufficiently generic and modular, features modern architecture and can serve both as scientific research testbed and a practical system. The outcome of this work is the **YodaQA** system.[1] (Baudiš, 2015b,a)

3.1 YodaQA Pipeline Architecture

The QA task is implemented in YodaQA as a pipeline that transforms the question to a set of answers by applying a variety of analysis engines and annotators. It is composed from largely independent modules, allowing easy extension with better algorithms or novel approaches, while as a fundamental principle all modules share a common end-to-end pipeline. The pipeline is implemented in Java

The YodaQA pipeline is implemented mainly in Java, using the Apache UIMA framework. (Ferrucci and Lally, 2004) YodaQA represents each artifact as a separate UIMA CAS, allowing easy parallelization and straightforward leverage of pre-existing NLP UIMA components; as a corollary, we compartmentalize different tasks to interchangeable UIMA annotators. Extensive support tooling is included within the package.

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 algorithms for the particular analysis steps, **provider** has interfaces to various external resources and **flow** carries UIMA helper classes and a web interface dashboard.

The system maps an input question to ordered list of answer candidates in a pipeline fashion, with the flow as in Fig. 3.1, encompassing the following stages:

- **Question Analysis** extracts natural language features from the input and produces in-system representations of the question.
- **Answer Production** generates a set of candidate answers based on the question, by performing a **Primary Search** in the knowledge bases according to the question clues and either directly using the results as candidate answers or selecting the relevant passages (the **Passage Extraction**) and generate candidate answers from these (the **Passage Analysis**).

[1]Available open source at `https://github.com/brmson/yodaqa` under the Apache Software Licence 2.0.

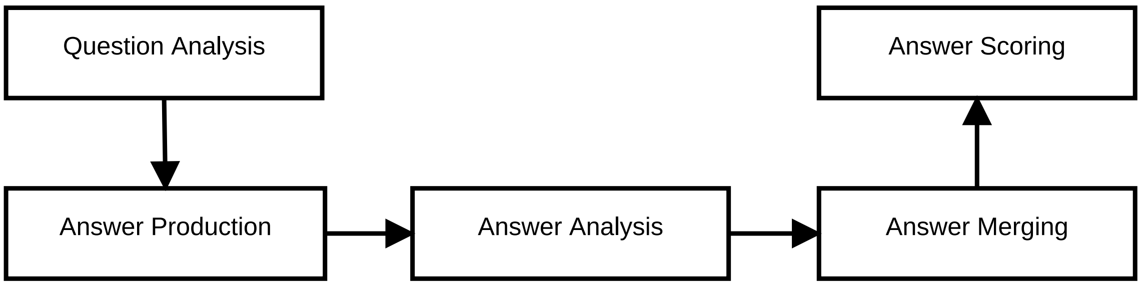

Figure 3.1: The general architecture of the YodaQA pipeline. Present but unused final pipeline portions not shown.

- **Answer Analysis** generates answer features based on detailed analysis (most importantly, lexical 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).[2]

The basic pipeline flow is much inspired by the DeepQA model of IBM Watson (Epstein et al., 2012). 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.

3.2 YodaQA Reference Pipeline

The reference pipeline currently considers an English-language task as outlined in the Introduction — answering open domain factoid questions, producing a narrowly phrased answer. We base the answers on information retrieval from both unstructured (English Wikipedia — *enwiki*) and structured (DBpedia (Lehmann et al., 2014), Freebase (Bollacker et al., 2008)) knowledge bases.

In our pipeline, we build on existing third-party NLP analysis tools, in particular Stanford CoreNLP (Segmenter, POS-Tagger, Parser) (Manning et al., 2014) (Chen and Manning, 2014), OpenNLP (Segmenter, NER) (Baldridge) and LanguageTool (Segmenter, Lemmatizer).[3][4] NLP analysis backends are freely interchangeable thanks to the DKPro UIMA interface (de Castilho and Gurevych, 2014). For semantic analysis, we also rely heavily on the WordNet lexicon (Miller, 1995).

Our key design rule is avoidance of hand-crafted rules and heuristics, instead relying just on fully-learned universal mechanisms; we use just about 10 hard-coded rules at this point, mostly in question analysis.

[2]We do not include Successive Refining in our evaluation or include further details as it is not beneficial in our current setup.

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

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

3.2.1 Question Analysis

The question analysis (Fig. 3.2) involves producing a part-of-speech tagging and dependency parse of the question text, recognizing named entities and performing entity linking[5] to concepts (as represented by *enwiki* articles). The question representation we produce is similar in spirit to DeepQA (Lally et al., 2012): a bag-of-features including a set of clues (keywords, keyphrases and linked concepts), possible lexical answer types and the selection verb.

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 and matched concepts; the question sentence subject (determined by dependency parse).

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 pre-defined category but may be an arbitrary English noun, like in the DeepQA system. (Murdock et al., 2012) 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.

SV (Selection Verb) represents the coordinating verb of the question that selects the answer with regard to other clues (like "born", "received", etc.)

3.2.2 Unstructured Knowledge Bases

The primary source of answers in our QA system is keyword search in free-text knowledge base (the *enwiki* in our default setting). While the knowledge base 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 (Chu-Carroll et al., 2012). We use the Apache Solr[6] search engine (frontend to Apache Lucene). They are shown as the non-highlighted components in Fig. 3.3.

Title-in-clue search (Chu-Carroll et al., 2012) looks for the question clues in the article titles, essentially aiming to find articles that describe the concepts touched in the question. The first sentences of the top six articles (which we assume is its summary) are then used in passage analysis (see below).

Full-text search (Chu-Carroll et al., 2012) 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 clues occuring in each passage[7][8]; the top three passages from each document are picked for passage analysis.

Document search (Chu-Carroll et al., 2012) 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

[5]Right now, entities are linked just by an exact match of the main or alias label.

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

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

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

their titles.

Concept search retrieves articles that have been linked to entities mentioned in the question. The first sentence as well as passages extracted like 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.

In addition, we implement 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. (Yao et al., 2013b) 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.

3.2.3 Structured Knowledge Bases

Aside of full-text search, we also employ structured knowledge bases organized in RDF triples; in particular, we query the DBpedia `ontology` (curated) and `property` (raw infobox) namespaces and the Freebase RDF dump. They are shown as the highlighted components in Fig. 3.3.

For each concept linked to an in-question entity, we query for predicates with this concept as a subject[9] and generate candidate answers for each object in such a triple, with the predicate label seeded as one LAT of the answer.

Furthermore, we have trained a multi-label classifier (logistic regression) that predicts *property paths* likely connecting an identified in-question concept with the answer in the knowledge base graph based on particular words in question representation. (Yao, 2015) Howeve, we consider long property paths (similar to the core inferential chain of STAGG (Chang and Gao, 2015)) as Freebase is organized such that finding related concepts often requires traversing intermediate nodes representing relationshibs (e.g. siblinghood); we also do not consider all unigrams and bigrams but just the LATs and SVs.

3.2.4 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*).

[9]All our knowledge bases are linked to *enwiki*.

- 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).

3.2.5 Answer Merge-and-Score

The merging and scoring process also basically follows a simplified DeepQA approach (Gondek et al., 2012). 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.[10] 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 $0.5/p$.

3.2.6 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 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.[11]

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

[10] An alternative gradient-boosted decision forest classifier is also available, but it is not beneficial in the default evaluation scenario yet.

[11] 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.

triggers significant overfitting on the training set and we therefore ignore the successive refining stage in the benchmarked pipeline.

3.3 Results

As we present evaluation 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 *AP recall* of the system (recall of the Answer Production component — whether a correct answer has been generated and considered, without regard to its score) and *precision@1* (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. These measures are further described in Ch. 2.

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).

3.3.1 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 tag `v1.2`. 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.

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.[12] Second, during training the models are randomly initialized and therefore their final performance on a testing set flutters a little.

As a main 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[13] 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.

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

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

We refer to the resulting 867 question dataset as `curated` and randomly split it to the training and testing sets.[14]

To further facilitate comparison of YodaQA to other systems, we also benchmark its performance on the (i) original, unrevised and unabridged TREC datasets (even though the train/test splits might not be entirely compatible), and (ii) the WebQuestions dataset (Berant et al., 2013) (which represents several thousands of open domain factoid questions tied to Freebase, popular as a semantic parsing benchmark).

3.3.2 System Evaluation

Evaluation over various pipeline configurations are laid out in Fig. 3.4. 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. 3.5. We can observe that while precision@1 is 31.4%, precision@5[15] is already at 50% of test questions.

The information retrieval parameters of the default pipeline are selected so that answering a question takes about 10s on average on a single core of AMD FX-8350 with 24GB RAM and SSD backed databases.[16] By raising the limiting parameters, we can observe further AP recall increase and a solid precision improvement. Our system could therefore meaningfully make use of further computing resources.

We also benchmarked accuracy 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 on the open domain factoid questions of the `curated` dataset. The "answer typing" hold-out represents suppression of external resources (Wordnet, DBpedia) when guessing answer types. We can see that entity linking in the question is also an important heuristic for our QA system.

3.3.3 Comparison to Other Systems

In Fig. 3.6, we compare various performance measures with the most relevant previously published systems on the TREC dataset, as reported in the respective papers, with ours benchmarked on non-curated version of the TREC 2002, 2003 dataset test split. This comparison has numerous caveats, though, as explained within the table. Our system has AP recall 60.9% on this dataset.

In Fig. 3.7, we compare ourselves with other published systems on the WebQuestions dataset (test split), as reported in their respective papers. We limit our system to use only structured KBs. Our system has AP recall 78.3% and MRR 0.431 on this dataset.

3.4 Datasets and Other Work

As part of our work, we published several other outputs besides the YodaQA system itself:

- The `curated` TREC-based dataset of factoid questions described above.[17]

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

[15]Proportion of questions where the correct answer is ranked in top five.

[16]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.

[17]`https://github.com/brmson/dataset-factoid-curated`

- The `webquestions` dataset that contains WebQuestions dataset in a more convenient format for further processing and automatically annotated by other data such as property paths between questions and answers.[18]
- The `movies` dataset that contains WebQuestions-based dataset for the movie domain, combined with questions based on web user input and feedback on an earlier version of YodaQA.[19] Made available in cooperation with Nguyen Hoang Long.
- A standalone `label-lookup` service for fuzzy lookup of labels in corpora.[20] Co-authored with Nguyen Hoang Long. (A fast Levenshtein distance based lookup library DAWG[21] is a work in progress, primarily authored by Tomáš Veselý.)
- A tool for evaluating the Google QA subsystem.[22] Primarily authored by Nguyen Hoang Long.

[18] `https://github.com/brmson/dataset-factoid-webquestions`
[19] `https://github.com/brmson/dataset-factoid-movies`
[20] `https://github.com/brmson/label-lookup`
[21] `https://github.com/brmson/dawg-levenshtein`
[22] `https://github.com/brmson/google-qa`

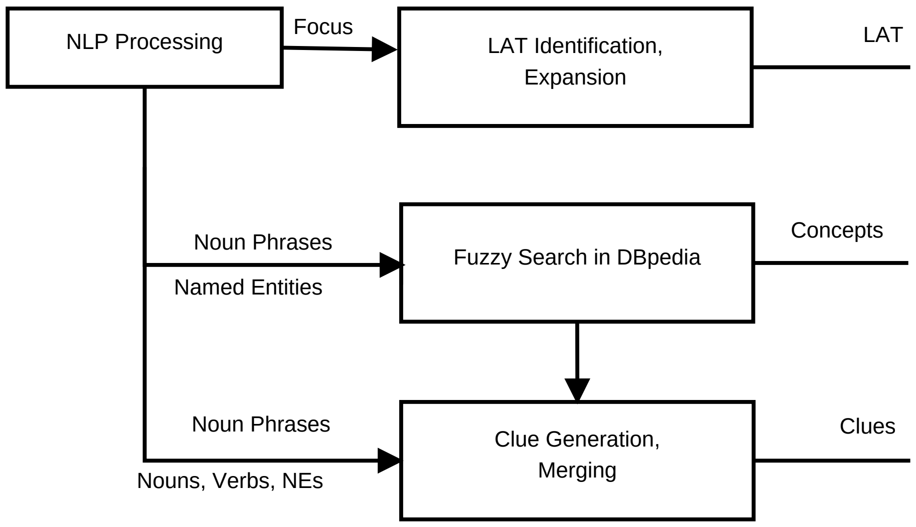

Figure 3.2: The general organization of the Question Analysis.

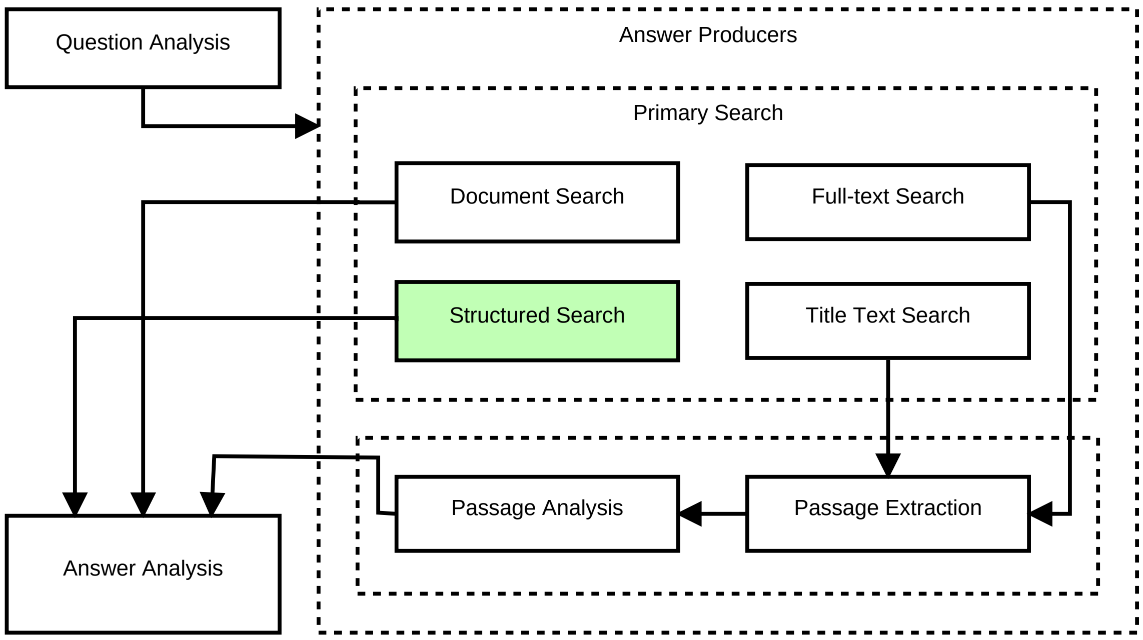

Figure 3.3: The general organization of the Answer Producer component — multiple strategies that generate large amount of answer hypotheses.

Pipeline	AP Recall	Prec@1	MRR
default	77.2%	31.4%	0.409
full-text scaling (6 → 12 fetched results)	80.0%	35.3%	0.440
enwiki KB hold-out	42.1%	19.8%	0.253
structured KB hold-out	70.7%	28.8%	0.378
answer typing hold-out	77.2%	30.7%	0.394
concept clues hold-out	68.1%	22.3%	0.318

Figure 3.4: Benchmark results of various pipeline variants on the *curated* test dataset. MRR is the Mean Reciprocal Rank $|Q| \cdot \sum_{q \in Q} 1/r_q$.

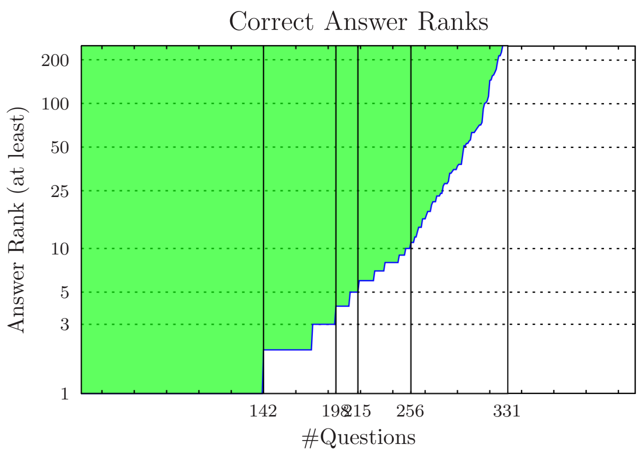

Figure 3.5: Number of questions x that output the correct answer ranked at least y.

System	Precision@1	F1	MRR
LLCpass03 (Harabagiu et al., 2003) (hand-crafted system)	68.5%	—	—
AskMSR (Brill et al., 2002) (web-search system)	61.4%	—	0.507
OpenEphyra (Schlaefer et al., 2006) (hand-crafted OSS)	“above 25%”	—	—
JacanaIR (Yao et al., 2013a) (modern fully-learned OSS)	—	23.1%*	—
OQA (Fader et al., 2014) (modern fully-learned OSS)	—	29%**	—
YodaQA v1.1	26.4%	26.4%	0.325

Figure 3.6: Benchmark results of some relevant systems on the unmodified TREC dataset.
* MIT99 subset; ** sub-sampled dataset with manual evaluation

System	F1@1	F1 (Berant)
Sempre	35.7%	35.7%
JacanaFB	35.4%	33.0%
YodaQA v1.1	34.3%	—
STAGG (state-of-art)	—	52.5%

Figure 3.7: Benchmark results of som relevant systems on the WebQuestions dataset. See also Sec. 2.1.

Chapter 4

Future Work

Ultimately, the goal of building a Question Answering system would be a machine that can answer any kind of question a human could (or more). But given the current state of art, I propose to consider this out of scope as a doctoral thesis topic. Instead, I shall propose two particular real world QA scenarios that are inspired by real world applications. Equipped with this vision and a baseline system of my own making, I shall consider the scientific questions that we need to answer in order to do well in these scenarios. I want to propose answering some of these questions to be the major focus of my thesis.

The first scenario involves answering **movie questions** about metadata of films and TV series (credits, characters, dates, awards, etc.). This represents a proxy for answering questions about product or company databases and many other business tasks. The domain is closed, but familiar to most users and with readily available knowledge bases. We already collected few hundred movie related questions as a dataset and are working on YodaQA version for answering movie questions.[1]

The second scenario aims to answer **prospectus questions** about contents of an isolated document like a product page, real estate offer or a usage manual. Unlike non-factoid QA (Sec. 2.5), the focus of this task is not answering logical riddles or following through with a story, but rather entity recognition and relationship extraction.

Both my proposals aim at realistic user interaction, which brings in the first question — **How to link question keywords to appropriate concepts?** That is, building a dataset and highly accurate solution for entity linking. Preliminary results on the movies dataset show that spelling mismatches (both typos and interpunction variants) as well as disambiguation issues (linking "Hobbit" to the best concept depending on question, or identifying that "the first time" might not be a movie reference) are non-trivial problems. Yao (2015) agrees that the current datasets like WebQuestions do not strain this issue realistically. Some aspects (like fast fuzzy searches) are mostly of engineering interest, but an example of semantically interesting scientific problem here is a general mechanism to prefer certain classes of concepts based on context, say movie instances of "The Hobbit" when asked about a director. A baseline idea (slightly inspired by Sun et al. (2015)) would be to rank concepts based on similarity of vector embeddings (Sec. 2.6) of the question and of the concept description. We are already in progress of building an entity linking dataset to properly compare various approaches.

How to map unseen question focuses to knowledge graph edges? The existing structured QA proposals (as surveyed in Sec. 2.1) rely on seeing each kind of question with

[1] `d/movies` branch in the YodaQA Git repository.

many variations to infer explicit mappings from in-question words to particular relations in the knowledge base ontology. In open domain, the long tail of exotic questions would escape this; in closed domain, amassing a large enough corpus of questions may be tricky. At the same time, a human operator has no trouble finding the appropriate relation based on its label. Therefore, seeking a general way to match questions to labels of properties (or even property paths) seems desirable, instead of simply learning a fixed mapping from individual in-question words. Again, leveraging an abstract representation of word meanings like vector embeddings would be promising.

How to pick the source sentences most likely to bear an answer? is an important question when dealing with question answering on unstructured text corpora. While Sec. 2.2.2 surveys some approaches, our preliminary research has shown that the current reference dataset is a very problematic benchmark as the test set appears to be much easier than the training set and realitsic QA setting. Moreover, the current approaches use compound vector embeddings of sentences, but it is troublesome when the sentences need to be matched based on dates, entity named and such, as explained below.

How to integrate relation extraction approaches with question answering? is a research question that would boost the more naive approaches of answer hypothesis generation by the research in relation extraction (e.g. ReVerb, RelEx, NELL, ...). One obvious approach is to first run general relation extraction on the knowledge corpus, then query such a structured database. However, I believe that the task of "reading" text with a particular question in mind makes the task of identifying pertaining relations considerably easier, and features used to extract a relation would be useful for scoring the answers too. Adapting well-known relation extraction strategies is a first step, but going further and exploiting synergies with my work on the other questions, I would like to experiment also with using embedded representation of the sentences (directly for relation extraction, or, say, as an attention mechanism in the spirit of Hermann et al. (2015)).

Many of the research questions might be in part answered by vector embeddings of words, or whole sentences. One reason we emphasize this approach so much is its high reusability potential: a well developed solution to one problem might help with another one, both on conceptual level and on the level of actual performance, e.g. applying multi-task learning to learn a common representation of source artifacts. To get there, we need to ask two important questions about vector embeddings themselves, though. **How to keep the signal level high and represent complex statements when building compound embeddings?** points out that right now, we do not know how to step from representing short and simple snippets or well known entities[2] to representing complex sentences (like in a typical Wikipedia article) or rare entities. Moreover, when we are to process sentences that contain numbers (like dates) or references to entities, we need to answer **How to embed or tie arbitrary data in addition to normal words to compound embeddings?** The current state of art does not seem to go beyond simple hacks like carrying bag-of-tokens along the embeddings and counting intersect size of that set when classifying relationships between two embeddings (Yu et al., 2014).

[2]When representing entities, we are thinking about representations built in the spirit of Iyyer et al. (2014).

Chapter 5

Conclusion

In this report, I have proposed a doctoral thesis on the topic of factoid question answering. So far, I have built a question answering system that exhibits performance more than comparable with well-known baselines and is still rapidly improving. I have also managed to reproduce state-of-art results in some of the sub-tasks and identified and proposed a few tasks on my own that are tied to building a state-of-art system in a real-world setting.

In my thesis, apart of describing my system building work, I would like to focus particularly on research connected to the Information Extraction portion of the system.

As discussed above, vector embeddings are a booming area of semantic NLP research with their ability to numerically capture meaning nuances, and based on the presented survey of the field, I believe looking into vector embedding approaches when manipulating natural text (analyzing the question, understanding answer-bearing passages and relation labels) is a highly promising area of research. Some scientific problems that I would like to tackle are connected to the current limitations of vector embeddings themselves. This includes building compound vector embeddings conveying the meaning of a whole sentence rather than individual words, and exploring ways to embed or tie named entities, numerical quantities and other arbitrary data to the conventional embeddings.

Other scientific problems related to Information Extraction that I plan to explore are in particular testing various relation extraction strategies in the unstructured knowledge base context and improving performance in this sub-task. My sight is also set at tasks beyond basic factoid QA like performing basic inference, though it is unclear if it will eventually fall in the scope of my thesis as well.

However, improving the system performance might lead to interesting results in other parts of the system, involving Entity Linking, Information Retrieval, Result Ranking etc.

In summary, I would like to propose a thesis studying **Semantic Methods in Question Answering**.

5.1 Acknowledgements

I would like to thank my main supervisor Dr. Petr Pošík for his guidance, support, but also giving me the freedom to eventually take a different direction of scientific pursuit than we originally planned. I would also like to thank my specialist supervisor Dr. Jan Šedivý for supporting me in this new topic of research, his result-focused guidance and giving me great opportunities to boost the development of my system.

Appendix A

Portfolio-Based Optimization

The initial focus of my doctoral research was developing algorithm portfolio strategies with applications particularly in continuous black-box optimization. The results of my work have been a software framework for experiments (Baudiš, 2013) and two results published at top-tier conferences (Baudiš and Pošík, 2014, 2015). This research had been primarily supervised by Dr. Petr Pošík.

Let us consider the problem of finding a minimum value of a continuous real-parameter function that has inaccessible analytical form.[1] This is a rich area of research that produced many algorithms over the last 50 years — from the venerable Nelder-Mead simplex algorithm (Nelder and Mead, 1965) to various gradient descent methods to population-based methods.

A.1 Online Black-box Algorithm Portfolios

The key question in the face of such variety of optimization algorithms is "which algorithm should I choose?" Unified comparison benchmarks (Hansen et al.) can help determining the best option for a particular function class. However, if a function is truly "black-box" and its features are hard to predict, an automated process with minimum overhead is certainly desirable.

The problem of algorithm selection is not new (Rice, 1976) and was so far popular mainly when applied to combinatorial problem solvers (Kotthoff, 2014). In my work, I adopted the prism of algorithm portfolios (Gomes and Selman, 2001) with *online* selection.[2] That is, multiple diverse optimization algorithms are applied to the given function instance simultaneously, with the best performers quickly gaining the largest time allocation (that is, the chance to perform the most optimization steps).

Deciding which algorithm to apply in each step of the portfolio optimization is essentialy an instance of the well-known Multi-Armed Bandit Problem, where a policy decides which algorithm to try next based on their empirically determined expected reward. I have built a modular Python framework **COCOpf** (Baudiš, 2013) suitable both for research and application of this problem.

[1]No analytical form implies that we do not have information e.g. on the derivatives of the function, except approximating them numerically.

[2]The concept of offline selection also occurs in the literature, when we assume a stream of function instances and apply just a single algorithm on each of the instances.

This helped me to identify fine structure of the problem (particularly, I proposed a classification of functions based on their in-portfolio behavior). Further, I have built a reference portfolio of seven well-known diverse optimization algorithms and based on performance evaluation using the popular COCO optimization benchmark (Hansen et al.), I have identified a policy that significantly improves the baseline and on well-behaved functions on average overperforms even the overally best individual algorithm. This result was presented at the PPSN 2014 conference. (Baudiš and Pošík, 2014)

A.2 Minimizing Separable Functions by a Mix of Methods

Another tier of research on how to best combine different optimization algorithms concerns speeding up optimization of continuous black-box *separable* functions in particular. I have closely cooperated on this research with my supervisor, Dr. Petr Pošík.

Separable multivariate functions can be decomposed to a sum of univariate functions, each parametrized solely by a single dimension of the input vector. For some very hard separable functions, exploiting separability is the only way to quickly find the minimum and a natural idea to optimize such functions is to use univariate optimization algorithms on individual dimensions. In (Pošík, 2009), Brent's method (Brent, 1973) and the STEP algorithm (Langerman et al., 1994) were used to separately optimize the function along each dimension. Brent's method was shown to be fast in case of unimodal functions, but due to its local nature it fails on multimodal functions. The global STEP method was able to solve both the uni- and multimodal functions, but needed much larger number of function evaluations. Moreover, their multidimensional variants were constructed inefficiently: the dimensions were optimized sequentially, one by one.

We have built on the above mentioned methods, and contributed two improvements:

1. We combined Brent's method and STEP into a single algorithm which converges faster than STEP (in many cases, it is almost as fast as Brent's method), while it preserves the global search ability of STEP (thus solving a larger proportion of functions than Brent's method, and often doing it faster).
2. We suggested a better way of making a multidimensional variant of this method. As opposed to solving the 1D problem in all dimensions sequentially, we proposed to interleave the steps in individual dimensions, updating the full coordinates of sampled points based on results obtained in other dimensions so far.[3]

Thus, we have introduced a new hybrid algorithm "Brent-STEP" combining these two methods non-trivially and demonstrated that on univariate and separable functions the hybrid algorithm in general outperforms both of them, in the univariate case often by a wide margin, and that it is behaving robustly even when one of the constituent methods is failing to converge. This result was presented at the GECCO 2015 conference. (Baudiš and Pošík, 2015)

[3]Later, we found out about a similar recent work by Ilya Loschilov Loshchilov et al. (2013), but our algorithm reaches better results.

Bibliography

Ali Mohamed Nabil Allam and Mohamed Hassan Haggag. The question answering systems: A survey. *International Journal of Research and Reviews in Information Sciences (IJRRIS)*, 2 (3), 2012. URL `http://aliallam.com/QA%20Survey%20Paper%20(IJRRIS).pdf`.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. URL `http://arxiv.org/abs/1409.0473`.

Jason Baldridge. The OpenNLP project. `http://opennlp.apache.org/`.

Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. Knowledge-based question answering as machine translation. 2014. URL `http://research.microsoft.com/en-US/people/nanduan/acl2014.pdf`.

Hannah Bast and Elmar Haussmann. More accurate question answering on freebase. 2015. URL `http://ad-publications.informatik.uni-freiburg.de/freebase-qa.pdf`.

Petr Baudiš and Jan Šedivỳ. Modeling of the question answering task in the yodaqa system. 2015. URL `http://ceur-ws.org/Vol-1391/131-CR.pdf`.

Petr Baudiš. COCOpf: An algorithm portfolio framework. In *Poster 2014 — the 18th International Student Conference on Electrical Engineering.* Czech Technical University, Prague, Czech Republic, 2013.

Petr Baudiš. 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, 2015a.

Petr Baudiš. YodaQA: A Modular Question Answering System Pipeline. In *POSTER 2015 - 19th International Student Conference on Electrical Engineering*, 2015b. URL `http://ailao.eu/yodaqa/yodaqa-poster2015.pdf`.

Petr Baudiš and Petr Pošík. Online black-box algorithm portfolios for continuous optimization. In Thomas Bartz-Beielstein, Jürgen Branke, Bogdan Filipič, and Jim Smith, editors, *Parallel Problem Solving from Nature – PPSN XIII*, volume 8672 of *Lecture Notes in Computer Science*, pages 40–49. Springer International Publishing, 2014. ISBN 978-3-319-10761-5. doi: 10.1007/978-3-319-10762-2_4. URL `http://dx.doi.org/10.1007/978-3-319-10762-2_4`.

Petr Baudiš and Petr Pošík. Global line search algorithm hybridized with quadratic interpolation and its extension to separable functions. In *Proceedings of the 2015 Conference on Genetic and Evolutionary Computation*, New York, NY, USA, 2015. ACM. Accepted for publication.

Jonathan Berant and Percy Liang. Semantic parsing via paraphrasing. 2014. URL `http://nlp.stanford.edu/pubs/berant14paraphrasing.pdf`.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, pages 1533–1544, 2013. URL `http://www.aclweb.org/anthology/D13-1160`.

Sonia Bergamaschi, Elton Domnori, Francesco Guerra, Mirko Orsini, Raquel Trillo Lado, and Yannis Velegrakis. Keymantic: semantic keyword-based searching in data integration systems. *Proceedings of the VLDB Endowment*, 3(1-2):1637–1640, 2010. URL `http://disi.unitn.it/~velgias/docs/BergamaschiDGOLV10.pdf`.

Lukas Blunschi, Claudio Jossen, Donald Kossmann, Magdalini Mori, and Kurt Stockinger. Soda: Generating sql for business users. *Proceedings of the VLDB Endowment*, 5(10):932–943, 2012. URL `http://arxiv.org/pdf/1207.0134.pdf`.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM, 2008.

Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676*, 2014. URL `http://arxiv.org/pdf/1406.3676`.

RP Brent. Algorithms for minimization without derivatives. *Prentice-Hall series in automatic computation*, 1973.

Eric Brill, Susan Dumais, and Michele Banko. An analysis of the AskMSR question-answering system. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 257–264. Association for Computational Linguistics, 2002.

Qingqing Cai and Alexander Yates. Large-scale semantic parsing via schema matching and lexicon extension. In *ACL (1)*, pages 423–433, 2013.

Wen-tau Yih Ming-Wei Chang and Xiaodong He Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. 2015. URL `http://research.microsoft.com/pubs/244749/ACL15-STAGG.pdf`.

Danqi Chen and Christopher D Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, 2014.

Jennifer Chu-Carroll, James Fan, BK Boguraev, David Carmel, Dafna Sheinwald, and Chris Welty. Finding needles in the haystack: Search and candidate generation. *IBM Journal of Research and Development*, 56(3.4):6–1, 2012.

Richard Eckart de Castilho and Iryna Gurevych. A broad-coverage collection of portable nlp components for building shareable analysis pipelines. In Nancy Ide and Jens Grivolla, editors, *Proceedings of the Workshop on OIAF4HLT at COLING 2014*, pages 1–11, Dublin, Ireland, August 2014. ACL and Dublin City University.

Edward A Epstein, Marshall I Schor, BS Iyer, Adam Lally, et al. Making watson fast. *IBM Journal of Research and Development*, 56(3.4):15–1, 2012.

Anthony Fader, Luke S Zettlemoyer, and Oren Etzioni. Paraphrase-driven learning for open question answering. In *ACL (1)*, pages 1608–1618, 2013. URL `http://homes.cs.washington.edu/~afader/bib_pdf/acl13.pdf`.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1156–1165. ACM, 2014. URL `http://homes.cs.washington.edu/~lsz/papers/fze-kdd14.pdf`.

David Ferrucci and Adam Lally. UIMA: An architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng.*, 10(3-4):327–348, September 2004. ISSN 1351-3249. doi: 10.1017/S1351324904003523. URL `http://dx.doi.org/10.1017/S1351324904003523`.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010. URL `http://www.aaai.org/ojs/index.php/aimagazine/article/download/2303/2165`.

David A Ferrucci. Introduction to “this is watson”. *IBM Journal of Research and Development*, 56(3.4):1–1, 2012.

Sean Gallagher, Wlodek Zadrozny, Walid Shalaby, and Adarsh Avadhani. Watsonsim: Overview of a question answering engine. *arXiv preprint arXiv:1412.0879*, 2014.

Carla P Gomes and Bart Selman. Algorithm portfolios. *Artificial Intelligence*, 126(1):43–62, 2001.

DC Gondek, Adam Lally, Aditya Kalyanpur, J William Murdock, Pablo Ariel Duboué, Lei Zhang, et al. A framework for merging and ranking of answers in deepqa. *IBM Journal of Research and Development*, 56(3.4):14–1, 2012.

Nikolaus Hansen et al. Comparing continuous optimisers: Coco. `http://coco.gforge.inria.fr/`.

Sanda M Harabagiu, Dan I Moldovan, Christine Clark, Mitchell Bowden, John Williams, and Jeremy Bensley. Answer mining by combining extraction techniques with abductive reasoning. In *TREC*, pages 375–382, 2003. URL `http://trec.nist.gov/pubs/trec12/papers/lcc.qa.pdf`.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *CoRR*, abs/1506.03340, 2015. URL `http://arxiv.org/abs/1506.03340`.

Felix Hill, Kyunghyun Cho, Anna Korhonen, and Yoshua Bengio. Learning to understand phrases by embedding the dictionary. *CoRR*, abs/1504.00548, 2015. URL `http://arxiv.org/abs/1504.00548`.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In *Empirical Methods in Natural Language Processing*, 2014. URL `http://cs.umd.edu/~miyyer/pubs/2014_qb_rnn.pdf`.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014. URL `http://arxiv.org/abs/1404.2188`.

Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014. URL `http://arxiv.org/pdf/1408.5882v2.pdf`.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. *arXiv preprint arXiv:1506.06726*, 2015. URL `http://arxiv.org/pdf/1506.06726v1.pdf`.

Lars Kotthoff. Algorithm selection for combinatorial search problems: A survey. *AI Magazine*, 2014.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. Scaling semantic parsers with on-the-fly ontology matching. 2013. URL `http://www.aclweb.org/anthology/D13-1161.pdf`.

Adam Lally, John M Prager, Michael C McCord, BK Boguraev, Siddharth Patwardhan, James Fan, Paul Fodor, and Jennifer Chu-Carroll. Question analysis: How watson reads a clue. *IBM Journal of Research and Development*, 56(3.4):2–1, 2012.

Stefan Langerman, Grégory Seront, and Hugues Bersini. Step: The easiest way to optimize a function. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 519–524, 1994.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 2014.

Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002. URL `http://ucrel.lancs.ac.uk/acl/C/C02/C02-1150.pdf`.

Percy Liang. Learning semantic parsers for natural language understanding. 2015. URL `http://www-cs.stanford.edu/~pliang/papers/semantic-parsing-intro.pdf`.

Vanessa Lopez, Victoria Uren, Marta Sabou, and Enrico Motta. Is question answering fit for the semantic web?: a survey. *Semantic Web*, 2(2):125–155, 2011. URL `http://oro.open.ac.uk/29573/1/swj124_3.pdf`.

Ilya Loshchilov, Marc Schoenauer, and Michèle Sebag. Bi-population cma-es agorithms with surrogate models and line searches. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, pages 1177–1184. ACM, 2013.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

Edgard Marx. openQA: Open source question answering framework, 2014. URL `http://aksw.org/Projects/openQA.html`.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b. URL `http://papers.nips.cc/paper/5021-di`.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. 2013c. URL `http://www.aclweb.org/anthology/N13-1#page=784`.

George A Miller. WordNet: a lexical database for english. *Communications of the ACM*, 38 (11):39–41, 1995.

J William Murdock, Aditya Kalyanpur, Chris Welty, James Fan, David A Ferrucci, DC Gondek, Lei Zhang, and Hiroshi Kanayama. Typing candidate answers using type coercion. *IBM Journal of Research and Development*, 56(3.4):7–1, 2012.

J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965. doi: 10.1093/comjnl/7.4.308. URL `http://comjnl.oxfordjournals.org/content/7/4/308.abstract`.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543, 2014. URL `http://llcao.net/cu-deeplearning15/presentation/nn-pres.pdf`.

Petr Pošík. BBOB-benchmarking two variants of the line-search algorithm. In *GECCO '09: Proceedings of the 11th annual conference companion on Genetic and evolutionary computation conference*, pages 2329–2336, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-505-5. doi: 10.1145/1570256.1570325. URL `http://dx.doi.org/10.1145/1570256.1570325`.

Siva Reddy, Mirella Lapata, and Mark Steedman. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2: 377–392, 2014. URL `http://www.aclweb.org/anthology/Q14-1030`.

John R Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.

Nico Schlaefer, Petra Gieselmann, Thomas Schaaf, and Alex Waibel. A pattern learning approach to question answering within the ephyra framework. In *Text, speech and dialogue*, pages 687–694. Springer, 2006. URL `http://petra.w-wie-wolf.de/tsd061a.pdf`.

Joao Silva, Luísa Coheur, Ana Cristina Mendes, and Andreas Wichert. From symbolic to sub-symbolic information in question classification. *Artificial Intelligence Review*, 35(2):137–154, 2011.

Amit Singhal. Introducing the knowledge graph: things, not strings. *Official Google Blog, May*, 2012.

Huan Sun, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang. Open domain question answering via semantic enrichment. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1045–1055. International World Wide Web Conferences Steering Committee, 2015. URL `http://www.cs.ucsb.edu/~huansun/docs/QA_paper.pdf`.

Wilson L Taylor. "Cloze procedure": a new tool for measuring readability. *Journalism quarterly*, 1953.

Christina Unger. Multilingual question answering over linked data: Qald-4 dataset, 2014.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *arXiv preprint arXiv:1411.4555*, 2014. URL `http://arxiv.org/abs/1411.4555`.

Ellen M Voorhees. Overview of the trec 2002 question answering track. In *In Proceedings of the Eleventh Text REtrieval Conference (TREC)*, 2002. URL `http://trec.nist.gov/pubs/trec11/papers/QA11.pdf`.

Ellen M Voorhees. Overview of the trec 2003 question answering track. 2003. URL `http://trec.nist.gov/pubs/trec12/papers/QA.OVERVIEW.pdf`.

Ellen M Voorhees and DM Tice. Overview of the trec-9 question answering track. In *TREC*, 2000. URL `http://trec.nist.gov/pubs/trec9/papers/qa_overview.pdf`.

Ellen M Voorhees and DM Tice. Overview of the trec 2001 question answering track. 2001. URL `http://trec.nist.gov/pubs/trec10/papers/qa10.pdf`.

Ellen M Voorhees et al. The trec-8 question answering track report. In *TREC*, volume 99, pages 77–82, 1999. URL `http://trec.nist.gov/pubs/trec8/papers/qa_report.pdf`.

Chang Wang, Aditya Kalyanpur, James Fan, Branimir K Boguraev, and DC Gondek. Relation extraction and scoring in deepqa. *IBM Journal of Research and Development*, 56(3.4):9–1, 2012.

Mengqiu Wang. A survey of answer extraction techniques in factoid question answering. In *Proceedings of the Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2006. URL `http://www.cs.cmu.edu/~mengqiu/publication/LSII-LitReview.pdf`.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*, volume 7, pages 22–32, 2007.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *CoRR*, abs/1410.3916, 2014. URL `http://arxiv.org/abs/1410.3916`.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698, 2015. URL `http://arxiv.org/abs/1502.05698`.

Yi Yang and Ming-Wei Chang. S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking. 2015. URL `http://www.anthology.aclweb.org/P/P15/P15-1049.pdf`.

Xuchen Yao. Lean question answering over freebase from scratch. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 66–70, Denver, Colorado, June 2015. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/N15-3014`.

Xuchen Yao and Benjamin Van Durme. Information extraction over structured data: Question answering with freebase. 2014. URL `http://www.aclweb.org/website/old_anthology/P/P14/P14-1090.pdf`.

Xuchen Yao, Benjamin Van Durme, and Peter Clark. Automatic coupling of answer extraction and information retrieval. In *ACL (2)*, pages 159–165, 2013a. URL `http://cs.jhu.edu/~xuchen/paper/yao-jacana-ir-acl2013.pdf`.

Xuchen Yao, Benjamin Van Durme, et al. Answer extraction as sequence tagging with tree edit distance. In *HLT-NAACL*, pages 858–867, 2013b. URL `http://cs.jhu.edu/~xuchen/paper/yao-jacana-qa-naacl2013.pdf`.

Xuchen Yao, Jonathan Berant, and Benjamin Van Durme. Freebase qa: Information extraction or semantic parsing? *ACL 2014*, page 82, 2014. URL `http://anthology.aclweb.org/W/W14/W14-24.pdf#page=94`.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. Deep Learning for Answer Sentence Selection. In *NIPS Deep Learning Workshop*, December 2014. URL `http://arxiv.org/abs/1412.1632`.

Han Zhao, Zhengdong Lu, and Pascal Poupart. Self-adaptive hierarchical sentence model. *arXiv preprint arXiv:1504.05070*, 2015. URL `http://arxiv.org/pdf/1504.05070v2.pdf`.