Advances in Cognitive Systems 6 (2018) 1-20 Submitted 6/2018; published 8/2018

Engineered AI Still Matters for Question Answering

J. William Murdock MURDOCKJ@US.IBM.COM
IBM Watson Research Center, 1101 Kitchawan Rd, Yorktown Heights, NY 10598 USA

Lin Pan PANL@US.IBM.COM
IBM, 550 King St, Littleton, MA 01460 USA

Chung-Wei Hang HANGC@US.IBM.COM
IBM, 3039 E Cornwalis Rd, Research Triangle Park, NC 27709 USA

Mary Swift MDSWIFT@US.IBM.COM
Zhiguo Wang ZHIGWANG@US.IBM.COM
IBM Watson Research Center, 1101 Kitchawan Rd, Yorktown Heights, NY 10598 USA

Chris Nolan CMNOLAN@US.IBM.COM
Prathyusha Peddi PEDDIS@US.IBM.COM
IBM, 3039 E Cornwalis Rd, Research Triangle Park, NC 27709 USA

Nisarga Markandaiah NMARKAN@US.IBM.COM
IBM Watson, 51 Astor Place, New York, NY 10003 USA

Eunyoung Ha EYHA@US.IBM.COM
Kazi Hasan KSHASAN@US.IBM.COM
Yang Yu YU@US.IBM.COM
Wei Zhang ZHANGWEI@US.IBM.COM
IBM, 550 King St, Littleton, MA 01460 USA

Abstract

Many question-answering systems rely on a significant amount of engineering effort. They often require both knowledge bases and rules, which can be very expensive to create. Even when there is significant statistical machine learning involved in these systems, there is also an enormous amount of effort spent on identifying what features are useful for the machine learning and implementing capabilities (often using knowledge bases and rules) to assign values to those features. However, in recent years an alternative approach has been growing in popularity: single-strategy systems in which one statistical model is used to address the entire task. These models take in questions and content and produce answers from that content. Of course, configuring a statistical learning system (e.g., deciding on the structure of a neural network) requires some amount of engineering work, but it can be vastly less. As a result, people building practical question-answering systems for commercial use have an incentive to prefer this kind of technology: it is cheaper to develop. Furthermore, there is mounting evidence in the field of reading comprehension to suggest that

single-strategy statistical models are extremely effective at answering questions. However, we believe that this evidence is a misleading artifact of the way that the reading comprehension task has been formalized. We explore two types of data: for conventional reading comprehension and for end-to-end factoid question answering. We show that the former is extraordinarily well suited to pure statistical methods and there is little additional benefit from engineered knowledge and rules. However, we also show that this result does not generalize to factoid question answering, where engineered knowledge and rules have substantially more impact. We conclude that the dramatic success of purely statistical methods on conventional reading comprehension reflect the artificially constrained nature of the problem and that engineered knowledge and rules remain useful for more realistic and open-ended tasks.

1. Introduction

A comprehensive question-answering system might be able to answer questions such as "What is the capital of New Jersey?" or "Which is more important, honesty or beauty?" Factoid question answering is a subproblem of question answering, in which questions (like the latter) that do not have concise factual answers are out of scope. Reading comprehension is also a constrained form of question answering: a specific piece of text is provided to the system along with a question and the system is required to answer the question using the text. In principle, a highly effective reading comprehension system should be similarly effective as a component of a factoid question-answering system: if a system is good at producing answers from pieces of text, then the factoid question-answering task breaks down into two simple steps: find some relevant text and then use your reading comprehension system to produce an answer using that text. In this paper, we explore this possibility by examining and synthesizing two radically different approaches to identifying answers to questions. We compare their effectiveness on a set of factoid question answering data and a set of (mostly factoid) reading comprehension data.

The two radically different approaches that we are examining and synthesizing are:

- A massively multi-strategy system consisting of many components that use manually engineered rules and/or knowledge plus a few components that use statistical machine learning
- A single-strategy system that directly maps question/passage pairs to answers without using any manually engineered features

We consider these approaches in the context of two different data sets. SQuAD 1.1 (Rajpurkar et al., 2016) is a reading comprehension data set that is widely used to train and assess statistical models. It was developed by showing humans articles from Wikipedia and asking them to create questions and for each question to highlight a chunk of text in the article that is the answer to the question. The questions range from prototypical factoid questions such as "What year was Super Bowl 50?" (answer: "2016") to questions with somewhat longer answers such as "Why are coastal species tough?" (answer: "to withstand waves and swirling sediment particles") that might also be considered factoid questions for a sufficiently broad definition of the term. Our second data set is an open-domain factoid question answering set in which questions seeking short factual answers were written without being tied to any particular source. We use a combination of Wikipedia and Wiktionary to find text containing answers for these questions.

The massively multi-strategy system that we present in the paper is named Discovery DeepQA (DDQA). This is a revised and updated version of the original IBM Watson 1.0 question-answering system that participated in the Jeopardy! challenge (Ferrucci, 2012). It embodies the key theoretical commitments of this original system, but it also includes significant differences in implementation details. It incorporates new technological innovations that post-date the 2011 Jeopardy! challenge, and it has been re-engineered to be less computationally expensive so it can scale out to many simultaneous users with relatively moderate cost.

We also present several different single-strategy statistical reading comprehension systems. All of them are deep neural networks that have been documented extensively in other publications that we cite, so we provide only a brief introduction to each. The key theoretical commitment that each of these systems makes is that they use a single statistical model that jointly learns the entire reading comprehension task (as formalized in SQuAD and similar data sets). These systems did require *some* engineering effort to build as they involve complex combinations of neurons arranged in looping graphs to facilitate memory and attention focusing. However, the amount of effort needed to configure the neural network is extremely small compared to the countless person-years spent creating all the knowledge and logic in DDQA (and its predecessor, IBM Watson 1.0).

We compare the effectiveness of DDQA and single-strategy statistical reading comprehension systems. We also show how the two can be integrated, by treating each single-strategy statistical reading comprehension system as an additional strategy within DDQA that can complement the other strategies. On the SQuAD reading comprehension data, single-strategy statistical systems are spectacularly effective and adding all of DDQA to them provides only very minimal benefit over the statistical systems alone. That result alone might suggest that all the time and effort needed to build this multi-strategy system with engineered rules and knowledge provides very little net value. However, when we test on the factoid question-answering system we see a dramatically different result: the single-strategy statistical systems are still reasonably effective but the integrated combination of DDQA and the statistical systems is substantially more accurate than either alone.

The next section of this paper describes DDQA: it explains the key theoretical commitments of the DeepQA framework (which are unchanged from IBM Watson 1.0) and presents some incremental revisions to components within that framework. The following section describes a set of single-strategy statistical reading comprehension systems. The next section then describes how we integrated those statistical reading comprehension systems as additional strategies within DDQA. Next we discuss related research. After that, we show experimental results for DDQA, the statistical reading comprehension systems, and both integrated. Finally, we provide conclusions in response to the experimental results.

2. Discovery DeepQA

Discovery DeepQA (DDQA) is a massively multi-strategy system that uses manually-engineered reasoning strategies and knowledge bases to identify candidate answers and provide scores that are relevant to whether those answers are correct and then uses a statistical model to combine those scores into a final conclusion about how to rank those candidate answers by likelihood of being correct.

DeepQA is the technological framework behind both the IBM Watson 1.0 system and DDQA. Key theoretical commitments of DeepQA include:

Massively multistrategy reasoning: DeepQA is designed to enable many different reasoning strategies to coexist and complement each other. In some cases, distinct components are employing different strategies and/or resources to perform the same task. For example, the IBM Watson 1.0 had a suite of passage scoring components (Murdock et al., 2012a) that all had roughly the same goal (determine if the passage supports some answer contained within it). In other cases, these are strategies with radically different inputs and goals: for example, knowledge-based reasoning components (Kalyanpur et al., 2012) are using radically different kinds of information to analyze answers than the passage scoring components. Multiple kinds of components and multiple instantiations of each kind of component complement each other and outperform what any one component can do alone.

Statistical machine learning to integrate results: In earlier question-answering systems, a common approach to combining complementary reasoning strategies was for the developers of those systems to make informed decisions about how to prioritize components. For example, COGEX (Moldovan et al., 2003) used semantic theorem proving as the highest priority method of ranking answers and would fall-back to more superficial methods only when the deep semantic theorem proving failed. This approach can be effective for combining a few strategies, but the massively multistrategy reasoning approach of DeepQA makes it infeasible for developers to manually determine what strategies should have priority over other strategies and how the conclusions of multiple strategies should be integrated.

Minimal commitment to inferences: One key to enabling massively multistrategy reasoning is to ensure that components do not make decisions that constrain the other components. For example, many older question-answering systems such as PIQUANT (Chu-Carroll et al., 2005) generate candidate answers using semantic types. If a question asks for a "city", they only produce candidate answers that they decide are cities. This requires the system to make a firm commitment to a binary decision of whether some string is or is not a city. In contrast, DeepQA systems generate many candidate answers that may or may not have the correct type and employ many competing strategies to determine if the answer has the correct type and also many other strategies to determine if the answer meets the other requirements of the question. The machine learning component then combines all these insights into a final conclusion. This approach requires much more computation than older approaches because the number of candidate answers considered is much larger. However, it also provides opportunities to find correct answers in cases where there is little or no evidence that some candidate answer has the correct type but sufficient other evidence that the answer is correct to offset that lack.

Deep semantic analysis: Many early efforts to find information using semantic analysis had little impact on effectiveness (Sanderson & Croft, 2012). Instead superficial methods like matching keywords and weighing by inverse frequency have consistently outperformed semantic analysis, which tends to be more brittle. DeepQA is able to make effective use of semantics because it uses a learning approach to integrate complementary reasoning strategies. For example, an ablation study of

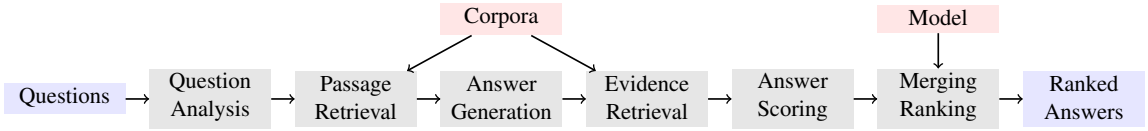

Figure 1. The DDQA factoid question answering pipeline.

IBM Watson 1.0 passage scorers (Murdock et al., 2012a) shows two key observations. First, if you can only include any single one of the passage scoring components, you are most effective if you select a particularly simple and superficial one, because those methods are highly robust and are frequently very effective. Second, if you exclude any single one of the passage scoring components, you lose the most effectiveness if you exclude the one that performs analogical reasoning over a deep analysis of semantic relations. At first these results seem contradictory: are the superficial strategies most important or are the deep semantic approaches most important? The evidence suggests that superficial strategies have more value for a single-strategy approach, but also that a multi-strategy approach with multiple superficial strategies benefits more from adding in a deep semantic strategy than from adding additional superficial strategies.

Figure 1 provides an overview of the components of DDQA. It is a slightly simplified version of the DeepQA architecture presented in (Ferrucci, 2012). The pipeline consists of the following steps:

- *Question Analysis*: Analyze the text of the question and determine what it is asking for.
- *Passage Retrieval*: Find passages that are relevant to the question.
- *Answer Generation*: Identify candidate answers from passages or from other sources.
- *Evidence Retrieval*: Find additional evidence such as more passages.
- *Answer Scoring*: Compute features of the candidate answers that are relevant to determining if they are correct.
- *Merging and Ranking*: Combine equivalent candidate answers and rank/score them.

The following subsections describe incremental revisions in each of these components that distinguish DDQA from IBM Watson 1.0. However, they key novelty for the purposes of this paper comes in later sections, where we describe some statistical reading comprehension systems, explain how they are integrated into DDQA, and show experimental results for DDQA alone, the statistical reading comprehension systems alone, and the integration of both.

2.1 Question Analysis

Question analysis in DDQA includes a core natural-language processing pipeline plus additional components that are specific to analyzing questions. The key element of the natural-language pro-

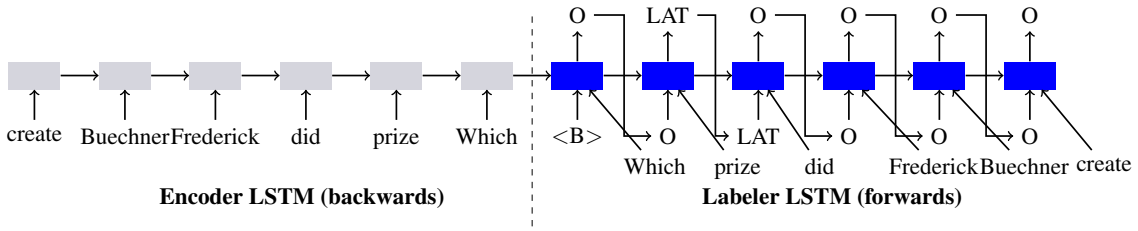

Figure 2. Encoder-labeler Long Short-Term Memory (LSTM) network for Lexical Answer Type (LAT) detection. A question is encoded backwards and then decoded back to itself with each token classified as Lexical Answer Type or not.

cessing pipeline for DDQA is the IBM Watson Relation Extraction service[1], which provides grammatical parses and detects entities and relations. The most important question-specific analysis is the detection of the *Lexical Answer Types*. Lexical Answer Types are question terms that refer to which type of answer is being asked for (Lally et al., 2012). For example, in question *Which prize did Frederick Buechner create?*, "prize" is the Lexical Answer Type indicating the answer to this question should be a kind of "prize." In DDQA, Lexical Answer Types are used by a set of *type coercion* scorers (Murdock et al., 2012b) that determine if a candidate answer is an instance of the answer types.

The original IBM Watson 1.0 system detects Lexical Answer Types using a rule-based approach with machine learning used only to compute confidence scores (Lally et al., 2012). While the approach detects Lexical Answer Types accurately and requires relatively little labeled data, it has several drawbacks. First, the approach requires handcrafted, comprehensive, language-dependent rules. Second, the rules are defined on top of parsing information such as part-of-speech tags and lemmas. As a result, its performance is highly sensitive to accurate parses.

DDQA adopts the encoder-labeler Long Short-Term Memory network (Kurata et al., 2016) for Lexical Answer Type detection. This supervised approach requires no handcrafted rules, no parsing or pretrained word embeddings. Figure 2 shows how the encoder-labeler Long Short-Term Memory network works. The encoder Long Short-Term Memory encodes a question into a fixed length vector by reading the question backwards word-by-word. Then the labeler Long Short-Term Memory initializes its hidden state as the vector encoded by the encoder, reads the question word-by-word, and predicts the label (a Lexical Answer Type or not) of each word, where its label is predicted by considering both the previous hidden state of the labeler and the previous label.

Our encoder-labeler Long Short-Term Memory network is trained on an internal question-answer set of 4,153 English questions with human labeled Lexical Answer Types. We manually pick the hyperparameters that yield reasonable results on a few confidential question sets. We set the learning rate to 0.001, the length of the hidden state to 256, the size of word embedding to 64 with context window size 7, the gradient clipping to 1.0, the dropout to 0.5, and the number of epochs to 15.

1. `https://www.ibm.com/blogs/insights-on-business/government/relationship-extraction/`.

2.2 Passage Retrieval and Supporting Evidence Retrieval

Most question-answering systems make use of a passage-retrieval component as a way to extract specific portions of documents within the corpus that potentially include the candidate answers (Ferrucci, 2012). In general, passage-retrieval is a multi-stage information retrieval process that includes document search and scoring to create a set of top documents, passage extraction from the document pool to create a set of passages for evaluation, and then a passage scoring stage to determine the best passages out of the passage pool. The DDQA system makes use of a custom library based on (Gómez et al., 2007), specifically designed for question answering. The document search component is built on top of Lucene[2] and makes use of Dirichlet Similarity (Zhai & Lafferty, 2001) for document scoring. The passage scoring component uses the document context in a Distance Density n-gram Model as described by Gómez et al..

When we apply DDQA to reading comprehension data sets, passage retrieval is not required because (by definition) the data sets provide the passages from which the answers should be derived. We do use passage retrieval when we work on question answering data sets that do not provide text for each question. In addition to the primary passage retrieval used to find answers, the DeepQA architecture allows supporting evidence retrieval to find additional evidence for each candidate answer. In principle, we believe that supporting evidence retrieval could be used for reading comprehension tasks, in much the same way that human readers rely on their own background knowledge to enrich their understanding of some text they read. We intend to work more on this topic in future research. For the results reported in this paper, we do not use supporting evidence retrieval.

2.3 Answer Generation and Scoring

DDQA employs a variety of additional mechanisms to identify and score candidate answers. Some are very similar to capabilities in IBM Watson 1.0. For example, DDQA uses an assortment of manually-engineered knowledge bases to determine if some candidate answer is consistent with the Lexical Answer Type using many of the strategies from (Kalyanpur et al., 2012). Others are novel but based on logic (like many of the Watson 1.0 components). For example, DDQA has a scorer that assesses the *grammatical coherence* of a candidate answer using the results of our core natural-language processing; "the tall boat tied up near the dock" is an example of a grammatically coherent answer because it is a complete noun phrase while "boat tied up near" is grammatically incoherent. This scorer is particularly useful in a pipeline that includes answer generation statistical methods, because such components are more likely to produce incoherent answers that are likely to be wrong (and the Ranker component described earlier can learn an appropriate amount to discount those answers).

2.4 Merging and Ranking

IBM Watson 1.0 uses a sequence of logistic regression models to independently compute the confidence in each answer. The models are aligned in a sequence to allow candidate answers to be

2. `https://lucene.apache.org/`.

scored and then transformed in various ways and then scored again. For example, in one phase of the merging and ranking process, IBM Watson 1.0 uses a knowledge base to determine whether candidate answers are semantically equivalent and when it does so, it uses the ranking from the previous phase as one indicator of which of the equivalent candidate answers emerges as the final answer (Gondek et al., 2012).

DDQA uses a similar (but smaller) cascade of multiple ranking phases (e.g., ranking both before and after merging) but instead of logistic regression, it uses the Ranker module from IBM Watson Retrieve and Rank[3] on the IBM Cloud. The Ranker module is an implementation of several learning-to-rank approaches including the LambdaMART algorithm (Burges, 2010) and traditional logistic regression, as well as ensembles of both. Data augmentation and subsampling techniques are also applied to improve robustness and performance of the module. At training time, the ranker receives numerical feature vectors representing question-answer pairs along with relevance labels (ground truth). The cross-validation method is used to determine an optimum approach and configuration for the training data. At runtime, given a set of unlabeled feature vectors corresponding to a list of candidate answers to a question, the ranker generates scores that can be used to sort the feature vectors to optimize for the relevance ranking in that list.

3. Single-Strategy Reading Comprehension Models

A variety of reading comprehension models have been proposed for extracting a single entity or a span of text from a given passage. We have conducted experiments with three models, all of which are deep neural networks:

- *Attention Sum Reader* (Kadlec et al., 2016) was originally designed to extract single word answers from passages that contain correct answers. For each question/passage pair, the model encodes question words and passage words with Bidirectional Gated Recurrent Unit networks. Each resulting vector representation of a passage word is scored against the question vector representation. The final score for a passage word is the sum of scores for all its occurrences in the same passage, and the word with the maximum score is identified as the answer.
- *Bi-Directional Attention Flow* (Seo et al., 2017) is a more complex model that addresses the task of extracting a span of text from a given passage. The model consists of six layers. Character and word embedding layers are responsible for generating combined embedding vectors for question and passage words. The contextual embedding layer models the temporal interactions between words by using Long Short-Term Memory Networks on the embedding vectors. Context-to-query and query-to-context attention are calculated in the attention flow layer to form query-aware representations of passage words. These representations are fed into Long Short-Term Memory Networks again in the modeling layer. The final output layer calculates begin and end probabilities of each phrase in the passage.
- *Multi-Perspective Context Matching* (Wang et al., 2017) is another model for extracting a span of text from a passage. The model bears a lot of similarity to the Bi-Directional Attention

3. https://www.ibm.com/watson/services/retrieve-and-rank/.

Flow model, with the key difference being how attentions are calculated to form query-aware passage representations. This is done by matching each passage word with the given question from multiple perspectives, based on the assumption that a span in a passage is more likely to be the correct answer if its context is more similar to the given question.

In this paper we report results only on Bi-Directional Attention Flow, because our experiments with the other two models are incomplete. The results that we have gotten from the other two models are all consistent with the conclusions that we present in Section 7.

4. Integrating Single-Strategy Models into DDQA

We integrate statistical reading comprehension models into DDQA as answer generation components that assign a score to each answer they generate. In the experiments in Section 6, we describe results on two different tasks: reading comprehension and factoid question answering. The former involves answering questions from a given piece of text while the latter involves answering questions from any available source. For reading comprehension we use the passage provided in the data set for each question and do not use the passage retrieval component in DDQA for any purpose. For factoid QA, we use the passage retrieval component in DDQA to find passages even when running experiments on the single-strategy statistical components, since they do require passages as input.

For both tasks, one challenge that we face with the integrated systems is the fact that the statistical reading comprehension model and the final ranking component in DDQA both need training data. As noted above, the outputs of the former are inputs to the latter. Consequently, if we were to train both using the same data, the model that the final merging and ranking component in DDQA learned would be distorted by the fact that its inputs were outputs from a model trained on that data. In practice, the statistical reading comprehension models we use tend to fit their training data much more closely than they fit other data, so the distortion that this would cause would be quite severe (i.e., DDQA would give far too much weight to the statistical reading comprehension models and not nearly enough weight to the scores of its other components).

We have two alternative solutions for this challenge: subdividing the training data for each purpose or folding the training data. The former solution involves designating some fraction of the training data as training for the statistical reading comprehension models and some fraction of the training data as training for the DDQA final model. The latter divides the entire train set into a fixed number of folds; for each fold we train the statistical reading comprehension models on the training data outside the fold and apply it to the training data inside the fold to provide model outputs that we then use as inputs to train the final ranking model. Subdividing the training data is much quicker and easier. In contrast, folding the training data requires more time and effort (since we need to train both the reading comprehension models once for each fold), but it allows both the statistical reading comprehension models and the DDQA ranking models to both use all of the data.

We present results in Section 6 in which we subdivide the data into 90% for the statistical reading comprehension model and 10% for the DDQA ranking models. Our statistical reading comprehension models are deep neural networks and thus benefit more from larger volumes of training data than the relatively simple model used for DDQA ranking, which is why we give them

a larger share of the data. We believe our results would be slightly better if we folded the training data instead, and we hope to conduct those experiments in future work.

Regardless of whether the training data is folded or subdivided, the statistical reading comprehension models need to be trained on question/passage pairs with some number of answers being labeled as correct. For reading comprehension data sets, that requirement is trivial to meet because the training data includes all that information. For factoid QA data sets, no passages are provided. Thus for each training query, we use the DDQA passage retrieval mechanism (described in Section 2.2) to retrieve passages and then construct training instances for the statistical reading comprehension model that pairs the question with all of the retrieved passages that contain at least one correct answer (according to the factoid ground truth for that training query).

5. Related Work

The work we describe in this paper involves fusing IBM Watson 1.0 with statistical models for reading comprehension. Here we compare our work with those to antecedents and then with other existing approaches to question answering.

5.1 IBM Watson 1.0

The original IBM Watson 1.0 system for Jeopardy! embodies essentially the same core theoretical commitments as DDQA, as described at the beginning of Section 2. One substantive difference between IBM Watson 1.0 and DDQA is that IBM Watson 1.0 retrieves multiple supporting passages for every candidate answer and combines the evidence scores for each of those answers (Murdock et al., 2012a). Another is that IBM Watson 1.0 had more more question analysis components, more primary search components, and more phases of scoring. Both of these differences reflect the fact that DDQA is designed to eventually be a cloud service than can handle many simultaneous requests without requiring a prohibitively expensive quantity of computing hardware; some of the most computationally expensive aspects of IBM Watson 1.0 were excluded from DDQA.

In addition, there are some new capabilities that are in DDQA and not in IBM Watson 1.0, as described in the subsections of Section 2. These capabilities represent relatively incremental enhancements because they fit within the existing architecture. They help DDQA to be effective despite the cuts from IBM Watson 1.0 that were needed to make DDQA cost effective to run.

5.2 Statistical Reading Comprehension

Reading comprehension is a task in which a system is given a passage and a question to be answered from that passage. It can be thought of as a slice or aspect of the more general question-answering task. Reading comprehension data sets include MCTest (Richardson et al., 2013), CNN and Daily-Mail (Hermann et al., 2015), bAbi (Weston et al., 2015), the Children's Book Test (Hill et al., 2016), NewsQA (Trischler et al., 2016), and SQuAD (Rajpurkar et al., 2016). Most systems that achieve high accuracy on this task use deep neural networks. Given a passage and a question asking about the passage, these systems "read" through the question and the passage and pinpoint the answer span within the passage.

In a later section of this paper, we present results on version 1.1 of the SQuAD data set. When we answer factoid questions using passages found by an information retrieval system is that some retrieved passages contain a correct answer but others do not; in contrast, all questions in SQuAD 1.1 contain the correct answer and provide sufficient evidence to conclude that the answer is correct (by design). This contradiction represents a defect in SQuAD 1.1 as a proxy for the task of extracting answers from found passages. SQuAD 2.0 (Rajpurkar et al., 2018) addresses this defect by adding in more than 50,000 additional questions where the correct answer is not available in the text, to test the ability of systems to recognize when no correct answer is available. We believe that this is a substantial improvement and we intend to use SQuAD 2.0 data in future work. However, we also believe that there are other crucial limitations of SQuAD, which we discuss in more detail in the concluding section of this paper.

5.3 Other Question-Answering Systems

Baseball (Green et al., 1961) is an early question-answering system that answers questions about baseball statistics. It works by parsing the question, determining the semantic relationships among the entities in the question, using these relationships to form a query against a structured knowledge base, and then retrieving the answer from that structured knowledge base. The Answer Lookup component in IBM Watson 1.0 (Kalyanpur et al., 2012) uses this same approach, with a much bigger and more broadly applicable knowledge base and much more powerful parsing and relationship detecting components. However, Answer Lookup was only a very marginal contributor to answer finding in IBM Watson 1.0, providing far fewer correct answers than answer generators based on text search, and the component is not included in DDQA.

IBM Watson's Answer Lookup, like Baseball and other predecessors using this approach, has limited effectiveness for two main reasons: existing knowledge bases cover only a small fraction of all questions that can be asked and even when the knowledge base does have the answer, reliably converting the question into a structured query is very difficult. As knowledge-bases such as DB-Pedia (Bizer et al., 2009) and Freebase (Bollacker et al., 2008) get larger, the former problem gets less severe but the latter problem gets more severe: very large and complex knowledge bases have an enormous variety of relation types among entities and as a result converting a question into a structured query becomes increasingly difficult. Aqqu (Bast & Haussmann, 2015) is remarkably effective at dealing with this challenge because it uses the contents of the knowledge base to constrain and guide the mapping of the question to a structured query: it looks at the entities in the questions, sees what relationships those entities participate in (and what relationships the related entities participate in) and then tries to match those relationships to the text of the question (using statistical methods). We have begun experimenting with integrating Aqqu into DDQA; none of the results reported in this paper include Aqqu, but we expect to have results with this or similar technology in the future. In the implementation reported in this paper, knowledge-bases are only used for scoring and merging answers found in text.

SAM (Lehnert, 1977) is able to answer reading comprehension questions in which the answer is not explicitly stated in the text but rather can be inferred indirectly from the context. For example, given a story about a bus ride and a question about how the passenger was able to ride the bus, SAM replies by saying that the passenger probably had a ticket, even though no ticket is mentioned in the

story. It does this using a "script" for bus riding in which a ticket is used to enable the bus riding. This is a very impressive result, and Lehnert makes a reasonable argument for the plausibility of the method used as a coarse model of human cognition. However, nobody has managed to engineer script knowledge of this sort at the scale needed to make SAM effective at answering a large percentage of real-world questions in a broad domain. Even if this knowledge were available, systems using it would face the challenge that systems answering factual questions from large factual knowledge bases do: mapping language to knowledge base queries is very hard. The Elemental Cognition Pipeline Model (Chu-Carroll, 2017) for reading comprehension also attempts to answer questions in which the answer is not explicitly stated but can only be indirectly inferred. Like SAM, it uses background knowledge and like the work reported in this paper, it also uses deep learning to provide broad coverage for answering easier questions too. The creators of this system have not provided extensive details of how it works, but the preliminary results provided seem promising.

6. Experimental Evaluation

In this section, we describe experiments on DDQA along with Bi-Directional Attention Flow, which is one of the single-strategy (deep learning) reading comprehension models described in Section 3. We compare the effectiveness of that single-strategy model alone to the effectiveness of the integrated system. The single-strategy model has an enormous practical advantage in that it requires no engineered knowledge, no engineered decomposition of the task (other than what is implicit in how their neural networks are structured), and no engineered reasoning strategies beyond forward and backward propagation. As commercial practitioners trying to build useful systems with minimal time and effort, we would prefer to just train a single-strategy model if that model can be as effective as an integrated system that includes both types of capabilities. We consider this opportunity first in the context of a reading comprehension data set, where single-strategy deep learning models are known to be highly effective. We then reconsider the question in the context of the general factoid question-answering task.

6.1 Data Sets

We evaluate DDQA and reading comprehension models on two data sets: **SQuAD 1.1** (Rajpurkar et al., 2016), a public reading comprehension data set, and **Factoid-1527**, a confidential factoid question answer set.

SQuAD 1.1 contains 107,785 question-answer pairs (87,599 training questions and 10,570 development questions which we use for evaluation). Each question-answer pair consists of a passage from a Wikipedia document, a crowd-sourced question based on the passage content, and an answer that is a span of the passage. Compared to other reading comprehension data sets, we choose SQuAD because (a) reasoning is required to answer the questions, (b) the passage span answers are easier to evaluate, (c) the questions are written by real crowdworkers, and (d) the answer types are more diverse beyond numbers and entities.

We also have a data set we refer to as Factoid-1527, which consists of 1,527 general-knowledge factoid trivia questions including topics like geography, politics, popular culture, etc. We use 1,272 of these questions for training and 255 for evaluation. This data set is confidential so we are not

Table 1. Discovery DeepQA (DDQA) and single-strategy model (Bi-Directional Attention Flow) results on the SQuAD 1.1 data set.

Configuration	Exact Match	Precision	Recall	MRR
Single-strategy model (full train)	66.90	77.61	80.45	71.67
Single-strategy model (90% train)	66.03	76.98	80.31	70.74
DDQA (10% train) + Single-strategy (90% train)	67.15	77.97	77.69	73.20

able to share it with the community and it is much smaller than SQuAD. However, it has a key trait that SQuAD does not: the questions were not written to be specific to a particular piece of text so they are useful for testing a complete system that finds relevant text and extracts answers from that text. In our experiments, we use a recent download of Wikipedia and Wiktionary as the corpora of text from which we find candidate answers.

As noted in Section 4, for both data sets we divide up our training data into 90% that we use to train the single-strategy model and 10% that we use to train the DDQA model that combines the single-strategy model with the engineered answer generation and scoring methods in DDQA. For both data sets, this subdivided training data is disjoint from the evaluation data; no data is used both to train and to evaluate the system.

6.2 SQuAD 1.1

Table 1 shows results for the single-strategy statistical model (Bi-Directional Attention Flow) trained on the full train set, the results of that model trained on 90% of the train set, and the results of the combined DDQA and single-strategy statistical model in which 90% of the training data is used to train the statistical model and 10% of the data is used to train the final ranking model that combines the output of the single-strategy statistical model with the values of all of the other scores produced by DDQA to rank answers. If you compare just the last two lines in this table, you see the direct impact of adding DDQA to the single-strategy system. However, this comparison seems somewhat incomplete to address the question of how much value the engineered knowledge and reasoning strategies in DDQA are contributing, because the benefits of these features are partially offset by the cost of having to set aside some training data to train the integration. So we believe that comparing the first line of this table to the third gives a more fair estimate of the net cost and benefit of adding DDQA and consequently allocating some training data for training the integration.

The exact match, precision, and recall metrics are used to assess the single top-ranked answer that each system outputs. The exact match score is percentage of queries for which the top-ranked answer exactly matches one of the ground-truth answers. The precision and recall measures treat the top-ranked answer and the ground-truth answers as predicted and expected bags of tokens; precision is the percentage of top-ranked answer tokens that are ground-truth answer tokens, and recall is the percentage of ground-truth answer tokens that are top-ranked answer tokens.

We also present the mean reciprocal rank (MRR) metric (Voorhees, 1999), which is not limited to assessing the top-ranked answer for each query. Instead, it looks at all the answers that the system returns and gives full credit when the top answer is correct and partial credit for when a

Table 2. Discovery DeepQA (DDQA) and single-strategy model (Bi-Directional Attention Flow) results on the Factoid-1527 data set.

Configuration	Exact Match	MRR
Upper bound for the retrieved passages	89.02	89.02
Single-strategy model (full train)	14.90	20.91
Single-strategy model (90% train)	15.29	20.98
DDQA (full train)	50.59	58.59
DDQA (10% train) + Single-strategy model (90% train)	47.45	55.66

lower ranked answer is correct (if the highest ranked correct answer is in second place then half credit is awarded, if it is in third place then one third credit is awarded, etc.). We believe that this metric is a useful complement to the recommended metrics for SQuAD because it measures the ability of the system to get a correct answer close to the top rank. This is capability is useful in practice for question-answering systems that present multiple alternative answers to users.

What we observe across all the numbers in Table 1 is that the single strategy model is quite effective on this data set, and that there is little or no measurable benefit to adding in all the engineered knowledge and reasoning that DDQA provides. However, we will see in the next section that we do not get the same result on our other data set.

6.3 Factoid-1527

Table 2 shows results on the Factoid-1527 data set. As with SQuAD, we show the single-strategy model data with both 90% train (to show the direct effect of adding engineered knowledge and reasoning) and with full train (to show the net effect of adding engineered knowledge and reasoning and allocating some training data to enable the integration). We also show the results with DDQA alone trained on the full data and the integrated system. The numbers in the top line of the table reflect the percentage of queries for which a correct answer appears in any of the retrieved passages; this is an upper bound for how well any system can do in finding answers in these passages. Of course, it is possible for a system to get even better results if it had more and better sources (including text, knowledge bases, etc.) from which to find answers.

For Factoid-1527, we do not report the token-level precision and recall numbers that we report for SQuAD. We believe that these numbers are useful for the SQuAD task because each question has a single passage from which to draw answers, so there is a small pool of tokens from which both ground-truth answers and system answers are drawn. In contrast, in the general factoid question-answering task embodied in Factoid-1527, the pool of available tokens are all possible tokens in all possible sources from which answers could be drawn. Consequently, assessing the results of the task as if it were trying to classify the tokens in some source makes less sense. Instead, the creators of the Factoid-1527 data set did their best to enumerate as many possible ways of expressing each correct answer as possible, and we just assess whether the answers that we produce exactly match one of those known ways to express a correct answer. As in the previous experiment, the exact

match scores reflect the percentage of queries for which the top-ranked answer exactly matches a ground-truth answer, and the mean reciprocal rank scores generalize those results by also providing partial credit for lower-ranked answers that exactly match a ground-truth answer.

The key observation from Table 2 is that the single strategy model is not very good at finding answers to questions in this data set, and severely under-performs both DDQA alone and the integrated system with both DDQA and the single strategy model. Consequently, the conclusion that one might draw from SQuAD alone (that single-strategy statistical learning systems can solve the complete task very well and engineered knowledge and features provide little additional value) do not seem to generalize to this task. We discuss this contradiction in more detail in the concluding section of this paper.

6.4 Limitations of the Evaluation

We have some concerns regarding the results reported above that we would address immediately if we had more time. We intend to provide more complete results in future work. Noteworthy limitations of the current set of results include the following:

- For both SQuAD and the Factoid-1527 data, we used a fixed train and test set that had been pre-determined by the original creators of this data. An alternative design would have been a cross-fold validation in which all the available data was split into multiple folds and for each fold the system was trained on data outside the fold and tested on data inside the fold. That design would produce test results on all the data instead of just a fraction of the data, and consequently, we would have more accurate estimates of how effective the system would be on comparable unseen data. We recommend this approach to anyone designing data sets and conducting experiments with them.
- In the experiments above, we subdivide training data for use in training the statistical reading comprehension models and for use in training the DDQA ranker. We believe that the integrated systems would be more effective if we used folded training (see Section 4) instead. Our factoid results show *worse* results with the integrated system than with DDQA alone, which we believe is a consequence of subdividing. Folded training experiments are more time-consuming to conduct, especially if you are doing cross-fold validation. However, we believe that this added effort is essential for determining whether the single-strategy statistical systems can improve a highly engineered multi-strategy system like DDQA. We do not believe that is is essential for this paper, because here we are exploring a different question: Can single-strategy statistical systems *replace* highly engineered multi-strategy systems? We do not need the best possible strategy for training the integration of the methods to answer this question.
- We do not use the SQuAD test set for any purpose, because the SQuAD developers do not publish the answers to the SQuAD test set. Researchers can only assess results for the SQuAD test set by packaging their system up into a Docker container and uploading it to servers owned by the SQuAD data owners. That seems like an excellent policy for ensuring fairness in evaluations. However, unfortunately, it is technically infeasible for us to bundle up all of

DDQA into a single Docker container and deploy it outside of the IBM Cloud, particularly given our dependency on IBM cloud services that our team does not control. So we are reporting results only on the development set. Reporting results on the development set seems to be common practice for researchers who work on this data (e.g., Rajpurkar et al. 2016; Seo et al. 2017). Consequently we believe that reporting results on this data is useful in enabling comparisons to other work. However, we also acknowledge that it would be more useful if we could also deploy our system on the SQuAD server so we could also report results on the SQuAD test set (as most researchers who use this data also do).

- It is common practice for statistical systems trained across many epochs to select the model that is most effective on the development set and apply that model to the test set. As noted above, we use the development set as our test set, so using this approach would produce deeply misleading results. Instead, we just take the output of the last training epoch. A consequence of this design is that the models we produce are slightly less effective than they would be otherwise.
- As noted earlier, the Factoid-1527 data set is much smaller than SQuAD, and it is confidential so other researchers cannot produce results on this data and compare to ours. In future work, we hope to use larger data sets for factoid question answering, and we plan to focus on data sets that are broadly available to the research community, such as WebQuestions (Berant et al., 2013).
- We report results only on DDQA and single-strategy systems. The Factoid-1527 data set was also used to test other IBM-internal question-answering systems, including some which relied much more heavily on statistical methods and achieved results that were comparable or better to DDQA on some metrics. However, all the systems that got good results were systems with a substantial number of distinct components engineered into a pipeline to address distinct subtasks separately. We believe that both sets of results support the view that engineering complex systems is more effective than solving the entire task via a single classifier. However, our results should not be seen as proof that knowledge and rules are more effective than statistical analysis. We believe that more work (with more and better data) is needed to explore the benefits of such components for building practical question-answering systems.
- We report results for the Bi-Directional Attention Flow model; for this model we ran experiment with both SQuAD and Factoid-1527. We also ran experiments with the Multi-Perspective Context Matching model on SQuAD and with the Attention Sum Reader on Factoid-1527, but we have omitted these results from this paper because they cannot be used to compare across the two data sets (since we ran each of them on only one of the sets). If we had more time, we would run a consistent set of experiments with all three models on both data sets and then report results on all three.
- In the next section, we discuss flaws in SQuAD that we believe partially explain why single-strategy systems are so effective on it. There are other data sets used for reading comprehension research for which single-strategy statistical systems have also been very effective, and some of them do not have the same flaws (but they do have different flaws). More research in-

volving more data sets is needed to better understand where single-strategy statistical systems can be highly effective on their own.

We do not believe that these issues collectively invalidate the conclusions in the next section, but we will have more confidence in those conclusions when we conduct more complete and powerful experiments in the future.

7. Conclusions

Is engineered AI obsolete for the question-answering task? Can the power that engineered knowledge and engineered logic provide be replaced by a single statistical model? Our results on the SQuAD data set seem to suggest the answer is yes: a deep neural network can be trained to be so effective that a combination of big knowledge bases and many independent reasoning strategies add little or nothing to that one statistical model alone. However, SQuAD is, in some sense, an artificial ("toy") problem: there is no direct practical application for a system in which users give the system a paragraph and ask a question about that paragraph and hope that it can find the answer they intend for it to find. There is nothing inherently wrong with using toy problems to explore AI strategies, but it is important to understand the limitations of those toy problems when drawing conclusions about the relative merits of different solutions.

For example, one passage in the SQuAD development set includes the sentence "Various species of poison dart frogs secrete lipophilic alkaloid toxins through their flesh" and a question/answer pair for that passage is "What are dart frogs are known to secrete?" and "lipophilic alkaloid toxins." The question and answer both seem valid, but the phrasing of the question seems remarkably well aligned with the phrasing of the answer. Many other SQuAD questions contain question answer pairs where the alignment is less specific, for example, a passage with the phrase "the hero Siegfried killing a dragon on the Drachenfels (Siebengebirge) ('dragons rock')" has the question/answer pair "What is the translation of Siebengebirge?" and "dragons rock.' In this example, the verb "translation" in the question is missing from the passage. It is not unreasonable to expect that given a large amount of training data, a statistical model could learn that a phrase in parentheses after a word is likely to be the answer to a question asking for a "translation" of that word. Would such a model really have learned to comprehend text? Or would it merely have learned a systematic regularity in the way people write questions to be answered by a given piece of text? To the extent that these models do the latter, we would expect them to have less value for answering questions that were not written by someone who was looking at the text.

Our results for factoid question answering suggest that perhaps the remarkable successes that single-strategy statistical models have on SQuAD are (at least to some degree) artifacts of the way in which SQuAD is generated. For SQuAD, our single-strategy statistical system was so effective that adding all of our engineered AI capabilities to it provided little or no benefit. But on the factoid data set, that same single-strategy statistical system was much less effective than our engineered AI capabilities or the combination of both. Because the questions in the factoid set are not written to be answered by given passages, we believe that they represent a better proxy for real-world information gathering needs *and* that they represent a challenge that is less inherently amenable to a pure statistical analysis.

We suspect that the flaws that SQuAD has as a proxy for general-purpose question answering may exemplify a broader trend in modern machine learning research. Researchers develop clever and innovative methods to systematically generate enormous quantities of data that appear to be a good proxy for some important real-world task (e.g., Hermann et al. 2015; Maas et al. 2011; Benzi et al. 2016). They build statistical models that are spectacularly effective at learning from that data, but it is not entirely clear how much those systems are learning to succeed at the important real-world task, and how much they are merely learning statistical trends in the systematic data generation methods. If our suspicions are correct, AI may be headed toward a major realignment when many apparent academic breakthroughs fail to deliver on their initial promise. We would not argue that *all* single-strategy statistical work is invalid as a result of this issue: clearly there are some tasks such as identifying objects in images where deep neural networks are remarkably effective in practical applications. However, we feel that these methods are more valuable as one tool in a diverse toolbox of AI methods than as a replacement for the entire toolbox.

We intend to conduct more experiments in which we compare single-strategy statistical systems with engineered AI and integrated combinations of the two. We expect that single-strategy statistical methods will continue to excel at some tasks but under-perform at other tasks. We do recognize that the evidence in this paper has room for improvement (as noted in the previous section). We will need to test more single-strategy statistical systems over larger and more widely available data before our results are highly persuasive to anyone who believes (as many do) that deep neural networks are the only important AI technology and that everything else in the field should be discarded.

Acknowledgments

We would like to thank the other creators of the original IBM Watson 1.0 system who provided the starting point for this work. We would also like to thank Bowen Zhou who provided the leadership and guidance that made Discovery DeepQA possible.

References

Bast, H., & Haussmann, E. (2015). More accurate question answering on Freebase. *Proceedings of the Twenty-Fourth ACM International on Conference on Information and Knowledge Management* (pp. 1431–1440). New York: ACM.

Benzi, K., Defferrard, M., Vandergheynst, P., & Bresson, X. (2016). FMA: A dataset for music analysis. *Computing Research Repository*, *abs/1612.01840*.

Berant, J., Chou, A., Frostig, R., & Liang, P. (2013). Semantic parsing on Freebase from question-answer pairs. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (pp. 1533–1544). Seattle, WA: ACL.

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., & Hellmann, S. (2009). DBpedia - A crystallization point for the web of data. *Web Semantics*, *7*, 154–165.

Bollacker, K. D., Evans, C., Paritosh, P., Sturge, T., & Taylor, J. (2008). Freebase: A collaboratively created graph database for structuring human knowledge. *Proceedings of the ACM SIGMOD*

International Conference on Management of Data (pp. 1247–1250). Vancouver, Canada: ACM.

Burges, C. J. C. (2010). *From RankNet to LambdaRank to LambdaMART: An overview*. Technical Report MSR-TR-2010-82, Microsoft Research.

Chu-Carroll, J. (2017). Beyond the state of the art in reading comprehension. *Proceedings of the 2017 O'Reilly Artificial Intelligence Conference*. Video on Safari Books Online.

Chu-Carroll, J., Czuba, K., Duboue, P. A., & Prager, J. M. (2005). IBM's PIQUANT II in TREC2005. *Proceeding of the Fourteenth Text REtrieval Conference*. Gaithersburg, MD: NIST.

Ferrucci, D. (2012). Introduction to "This is Watson". *IBM Journal of Research and Development*, *56*, 1–15.

Gómez, J. M., Buscaldi, D., Rosso, P., & Sanchis, E. (2007). JIRS language-independent passage retrieval system a comparative study. *Proceedings of the Fifth International Conference on Natural Language Processing* (pp. 4–6). Hyderabad, India: ACL.

Gondek, D. C., Lally, A., Kalyanpur, A., Murdock, J. W., Duboue, P. A., Zhang, L., Pan, Y., Qiu, Z. M., & Welty, C. (2012). A framework for merging and ranking of answers in DeepQA. *IBM Journal of Research and Development*, *56*, 1–12.

Green, Jr., B. F., Wolf, A. K., Chomsky, C., & Laughery, K. (1961). Baseball: An automatic question-answerer. *Papers Presented at the 1961 Western Joint IRE-AIEE-ACM Computer Conference* (pp. 219–224). New York: ACM.

Hermann, K. M., Kociský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., & Blunsom, P. (2015). Teaching machines to read and comprehend. *Advances in Neural Information Processing Systems* (pp. 1693–1701). Montreal, Canada: NIPS.

Hill, F., Bordes, A., Chopra, S., & Weston, J. (2016). The Goldilocks principle: Reading children's books with explicit memory representations. *Proceedings of the Fourth International Conference on Learning Representation*. San Juan, Puerto Rico: ICLR.

Kadlec, R., Schmid, M., Bajgar, O., & Kleindienst, J. (2016). Text understanding with the attention sum reader network. *Proceedings of the Fifty-Fourth Annual Meeting of the Association for Computational Linguistics* (pp. 908–918). Berlin: ACL.

Kalyanpur, A., et al. (2012). Structured data and inference in DeepQA. *IBM Journal of Research and Development*, *56*, 1–14.

Kurata, G., Xiang, B., Zhou, B., & Yu, M. (2016). Leveraging sentence-level information with encoder LSTM for semantic slot filling. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (pp. 2077–2083). Austin, TX: ACL.

Lally, A., Prager, J. M., McCord, M. C., Boguraev, B. K., Patwardhan, S., Fan, J., Fodor, P., & Chu-Carroll, J. (2012). Question analysis: How Watson reads a clue. *IBM Journal of Research and Development*, *56*, 1–14.

Lehnert, W. (1977). Human and computational question answering. *Cognitive Science*, *1*, 47–73.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. *Proceedings of the Forty-Ninth Annual Meeting of the Association*

for Computational Linguistics (pp. 142–150). Portland, OR: ACL.

Moldovan, D., Clark, C., Harabagiu, S., & Maiorano, S. (2003). Cogex: A logic prover for question answering. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology* (pp. 87–93). Edmonton, Canada: ACL.

Murdock, J. W., Fan, J., Lally, A., Shima, H., & Boguraev, B. K. (2012a). Textual evidence gathering. *IBM Journal of Research and Development*, *56*, 1–14.

Murdock, J. W., Kalyanpur, A., Welty, C., Fan, J., Ferrucci, D. A., Gondek, D. C., Zhang, L., & Kanayama, H. (2012b). Typing candidate answers using type coercion. *IBM Journal of Research and Development*, *56*, 1–13.

Rajpurkar, P., Jia, R., & Liang, P. (2018). Know what you don't know: Unanswerable questions for squad. *arXiv preprint, arXiv:1806.03822v1*.

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (pp. 2383–2392). Austin, TX: ACL.

Richardson, M., Burges, C. J. C., & Renshaw, E. (2013). MCTest: A challenge dataset for the open-domain machine comprehension of text. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (pp. 193–203). Seattle: ACL.

Sanderson, M., & Croft, W. B. (2012). The history of information retrieval research. *Proceedings of the IEEE*, *100*, 1444–1451.

Seo, M., Kembhavi, A., Farhadi, A., & Hajishirzi, H. (2017). Bidirectional attention flow for machine comprehension. *Proceedings of the Fifth International Conference on Learning Representation*. Toulon, France: ICLR.

Trischler, A., Wang, T., Yuan, X., Harris, J., Sordoni, A., Bachman, P., & Suleman, K. (2016). NewsQA: A machine comprehension dataset. *arXiv preprint, arXiv:1611.09830*.

Voorhees, E. M. (1999). TREC-8 question answering track report. *Proceedings of the Eighth Text Retrieval Conference* (pp. 77–82). NIST: Gaithersburg, MD.

Wang, Z., Mi, H., Hamza, W., & Florian, R. (2017). Multi-perspective context matching for machine comprehension. *arXiv preprint, arXiv:1612.04211*.

Weston, J., Bordes, A., Chopra, S., & Mikolov, T. (2015). Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv preprint, arXiv:1502.05698*.

Zhai, C., & Lafferty, J. (2001). A study of smoothing methods for language models applied to ad hoc information retrieval. *Proceedings of the Twenty-Fourth Annual International Conference on Research and Development in Information Retrieval* (pp. 334–342). New Orleans: ACM Press.