Framing QA as Building and Ranking Intersentence Answer Justifications

Peter Jansen[*]
University of Arizona

Rebecca Sharp[**]
University of Arizona

Mihai Surdeanu[†]
University of Arizona

Peter Clark[‡]
Allen Institute for Artificial Intelligence

We propose a question answering (QA) approach for standardized science exams that both identifies correct answers and produces compelling human-readable justifications for why those answers are correct. Our method first identifies the actual information needed in a question using psycholinguistic concreteness norms, then uses this information need to construct answer justifications by aggregating multiple sentences from different knowledge bases using syntactic and lexical information. We then jointly rank answers and their justifications using a reranking perceptron that treats justification quality as a latent variable. We evaluate our method on 1,000 multiple-choice questions from elementary school science exams, and empirically demonstrate that it performs better than several strong baselines, including neural network approaches. Our best configuration answers 44% of the questions correctly, where the top justifications for 57% of these correct answers contain a compelling human-readable justification that explains the inference required to arrive at the correct answer. We include a detailed characterization of the justification quality for both our method and a strong baseline, and show that information aggregation is key to addressing the information need in complex questions.

1. Introduction

To encourage an emphasis on the task of explainable inference for question answering (QA), Clark (2015) introduced the Aristo challenge, a QA task focused on developing methods of automated inference capable of passing standardized elementary school

[*] School of Information, University of Arizona, Tucson, AZ, 85721. E-mail: pajansen@email.arizona.edu.
[**] Department of Linguistics, University of Arizona, Tucson, AZ, 85721.
[†] Department of Computer Science, University of Arizona, Tucson, AZ, 85721.
[‡] Allen Institute for Artificial Intelligence, 2157 N Northlake Way, Suite 110, Seattle, WA 98103.

Submission received: 24 November 2015; revised version received: 22 August 2016; accepted for publication: 28 November 2016.

doi:10.1162/COLI_a_00287

science exams while also providing human-readable explanations (or justifications) for those answers. Science exams are an important proving ground for QA because inference is often required to arrive at a correct answer, and, commonly, incorrect answers that are high semantic associates of either the question or correct answer are included to "lure" students (or automated methods) away from the correct response. Adding to the challenge, not only is inference required for science exams, but several kinds of inference are present. In an analysis of three years of standardized science exams, Clark et al. (2013) identified three main categories of questions based on the methods likely required to answer them correctly. Examples of these questions can be seen in Table 1, highlighting that 65% of questions require some form of inference to be answered correctly.

We propose a QA approach that jointly addresses answer extraction and justification. We believe that justifying why the QA algorithm believes an answer is correct is, in many cases, a critical part of the QA process. For example, in the medical domain, a user would not trust a system that recommends invasive procedures without giving a justification as to why (e.g., "Smith (2005) found procedure *X* healed 90% of patients with heart disease who also had secondary pulmonary complications"). A QA tool is clearly more useful when its human user can identify both when it functions correctly and when it delivers an incorrect or misleading result—especially in situations where incorrect results carry a high cost.

To address these issues, here we reframe QA from the task of scoring (or reranking) answers to a process of *generating and evaluating justifications* for why a particular answer candidate is correct. We focus on answering science exam questions, where many questions require complex inferences, and where building and evaluating answer justifications is challenging. In particular, we construct justifications by *aggregating*

Table 1
Categories of questions and their relative frequencies as identified by Clark et al. (2013). Retrieval-based questions (including *is–a*, dictionary definition, and property identification questions) tend to be answerable using information retrieval methods over structured knowledge bases, including taxonomies and dictionaries. More complex general inference questions make use of either simple inference rules that apply to a particular situation, a knowledge of causality, or a knowledge of simple processes (such as *solids melt when heated*). Difficult model-based reasoning questions require a domain-specific model of how a process works, like how gravity causes planets to orbit stars, in order to be correctly answered. Note that we do not include diagram questions, as they require specialized spatial reasoning that is beyond the scope of this work.

Category	Example	
Retrieval (35%)	Q: The movement of soil by wind or water is called:	
	(A) condensation	(B) evaporation
	(C) erosion	(D) friction
General Inference (39%)	Q: Which example describes an organism taking in nutrients?	
	(A) A dog burying a bone	(B) A girl eating an apple
	(C) An insect crawling on a leaf	(D) A boy planting tomatoes in the garden
Model-based Inference (26%)	Q: When a baby shakes a rattle, it makes a noise. Which form of energy was changed to sound energy?	
	(A) electrical	(B) light
	(C) mechanical	(D) heat

multiple sentences from a number of textual knowledge bases (e.g., study guides [SGs], science dictionaries), which, in combination, aim to explain the answer. We then rank these candidate justifications based on a number of measures designed to assess how well-integrated, relevant, and on-topic a given justification is, and select the answer that corresponds to the highest-ranked justification.

The specific contributions of this work are:

1. We propose a method to construct answer justifications through information aggregation (or fusion). In particular, we aggregate multiple sentences into hierarchical graph structures (called text aggregation graphs) that capture both intrasentence syntactic structures and intersentence lexical overlaps. Further, we model whether the intersentence lexical overlap is between contextually relevant keywords critical to the justification, or other words that may or may not be relevant. Our empirical analysis demonstrates that modeling the contextual relevance of intersentence connections is crucial for good performance. These requirements highlight the fundamental differences between selecting a single answer sentence or short passage in an answer sentence selection task (Severyn and Moschitti 2012, 2013; Severyn, Nicosia, and Moschitti 2013, inter alia), and the task of generating complete answer justifications through information aggregation.
2. We introduce a latent-variable ranking perceptron algorithm that learns to jointly rank answers and justifications, where the quality of justifications is modeled as the latent variable.
3. We evaluate our system on a large corpus of 1,000 elementary science exam questions from third to fifth grade, and demonstrate that our system significantly outperforms several strong learning-to-rank baselines at the task of choosing the correct answer. Further, we manually annotate answer justifications provided by the best baseline model and our intersentence aggregation method, and show that the intersentence aggregation method produces good justifications for 57% of questions answered correctly, significantly outperforming the best baseline method.
4. Through an in-depth error analysis we show that most of the issues encountered by the intersentence aggregation method center on solvable surface issues rather than complex inference issues. To our knowledge, this is the largest evaluation and most in-depth error analysis for explainable inference in the context of elementary science exams.

The article is structured as follows. We review related work in Section 2. We introduce the overall architecture of our QA system in Section 3. We describe our approach of identifying which words in the question are relevant for inference in Section 4. Building upon this knowledge, in Section 5 we introduce text aggregation graphs as the underlying representation for multisentence justifications, and characterize the types of connections captured by text aggregation graphs in Section 6. In Section 7 we introduce our latent-variable ranking perceptron, which jointly learns to identify good justifications and correct answers. Sections 8, 9, and 10 empirically demonstrate performance, discuss the results, and analyze the error classes observed, respectively. We conclude in Section 11.

2. Related Work

In one sense, QA systems can be described in terms of their position along a formality continuum ranging from shallow models that rely on information retrieval, lexical semantics, or alignment to highly structured models based on first-order logic.

On the shallower end of the spectrum, QA models can be constructed either from structured text, such as question–answer pairs, or unstructured text. Alignment models (Berger et al. 2000; Echihabi and Marcu 2003; Soricut and Brill 2006; Riezler et al. 2007; Surdeanu, Ciaramita, and Zaragoza 2011; Yao et al. 2013) require aligned question–answer pairs for training, a burden that often limits their practical usage (though Sharp et al. [2015] recently proposed a method for using the discourse structure of free text as a surrogate for this alignment structure). Lexical semantic models such as neural-network language models (Yih et al. 2013; Jansen, Surdeanu, and Clark 2014; Sultan, Bethard, and Sumner 2014), on the other hand, have the advantage of being readily constructed from free text. Fried et al. (2015) called these approaches first-order models because associations are explicitly learned, and introduced a higher-order lexical semantics QA model where indirect associations are detected through traversals of the association graph. Other recent efforts have applied deep learning architectures to QA to learn non-linear answer scoring functions that model lexical semantics (Iyyer et al. 2014; Hermann et al. 2015). However, although lexical semantic approaches to QA have shown robust performance across a variety of tasks, a disadvantage of these methods is that, even when a correct answer is selected, there is no clear human-readable justification for that selection.

Closer to the other end of the formality continuum, several approaches were proposed to not only select a correct answer, but also provide a formally valid justification for that answer. For example, some QA systems have sought to answer questions by creating formal proofs driven by logic reasoning (Moldovan et al. 2003a, 2007; Balduccini, Baral, and Lierler 2008; MacCartney 2009; Lewis and Steedman 2013; Liang, Jordan, and Klein 2013) answer-set programming (Tari and Baral 2006; Baral, Liang, and Nguyen 2011; Baral and Liang 2012; Baral, Vo, and Liang 2012), or connecting semantic graphs (Banarescu et al. 2013; Sharma et al. 2015). However, the formal representations used in these systems (e.g., logic forms) are both expensive to generate and tend to be brittle because they rely extensively on imperfect tools such as complete syntactic analysis and word sense disambiguation. We offer the lightly structured sentence representation generated by our approach (see Section 5) as a shallower and consequently more robust approximation of those logical forms, and show that they are well-suited for the complexity of our questions. Our approach allows us to robustly aggregate information from a variety of knowledge sources to create human-readable answer justifications. It is these justifications that are then ranked in order to choose the correct answer, using a reranking perceptron with a latent layer that models the correctness of those justifications.

Covering the middle ground between shallow and formal representations, learning to rank methods based on tree-kernels (Moschitti 2004) perform well for various QA tasks, including passage reranking, answer sentence selection, or answer extraction (Moschitti et al. 2007; Moschitti and Quarteroni 2011; Severyn and Moschitti 2012, 2013; Severyn, Nicosia, and Moschitti 2013; Tymoshenko and Moschitti 2015, inter alia). The key to tree kernels' success is their ability to automate feature engineering rather than having to rely on hand-crafted features, which allows them to explore a larger representation space. Further, tree kernels operate over structures that encode syntax and/or shallow semantics such as semantic role labeling (Severyn and Moschitti

2012), knowledge from structured databases (Tymoshenko and Moschitti 2015), and higher-level semantic information such as question category and focus words (Severyn, Nicosia, and Moschitti 2013). Here, we similarly use structural features based on syntax, enriched with additional information about how the answer candidate, the question, and the aggregated justification relate to each other. A key difference between our work and methods based on tree kernels is that rather than selecting a contiguous segment of text (sentence or paragraph), our justifications are aggregated from multiple sentences, often from different documents. Because of this set-up, we explore content representations that continue to use syntax, but are combined with robust strategies for cross-sentence connections. Further, because our justification search space is increased considerably thanks to the ability to form cross-sentence justifications, we restrict our learning models to linear classifiers that learn efficiently at this scale. However, as discussed, tree kernels offer distinct advantages over linear models. We leave the adaptation of tree kernels to the problem discussed here as future work.

Information aggregation (or fusion) is broadly defined as the assembly of knowledge from different sources, and has been used in several NLP applications, including summarization and QA. In the context of summarization, information aggregation has been used to assemble summaries from non-contiguous text fragments (Barzilay, McKeown, and Elhadad 1999; Barzilay and McKeown 2005, inter alia), whereas in QA, aggregation has been used to assemble answers to both factoid questions (Pradhan et al. 2002) and definitional questions (Blair-Goldensohn, McKeown, and Schlaikjer 2003). Critical to the current work, in an in-depth open-domain QA error analysis, Moldovan et al. (2003b) identified a subset of questions for which information from a single source is not sufficient, and designated a separate class within their taxonomy of QA systems for those systems that were capable of performing answer fusion. Combining multiple sources, however, creates the need for context disambiguation—an issue we tackle through the use of question and answer focus words.

Identifying question focus words, a subtask of question decomposition and identifying information needs, was found relevant for QA (especially factoid) early on (Harabagiu et al. 2000; Moldovan et al. 2003b, inter alia) mainly as a means to identify answer types (e.g., “What is the *capital* of France?” indicates that the expected answer type is *City*). Recently, Park and Croft (2015) have used focus words to reduce semantic drift in query expansion by conditioning on the focus words when expanding non-focus query words. Similarly, here, we use focus words (from both question and answer) to reduce the interference of noise in both building and ranking answer justifications. By identifying which words are most likely to be important for finding the answer, we are able to generate justifications that preferentially connect sentences together on these focus words. This results in justifications that are better able to remain on-context, and as we demonstrate in Section 8, this boosts overall performance.

Once the candidate answer justifications are assembled, our method selects the answer that corresponds to the best (i.e., highest-scoring) justification. We learn which justifications are indicative of a correct answer by extending ranking perceptrons (Shen and Joshi 2005) that have been previously used in QA (Surdeanu, Ciaramita, and Zaragoza 2011) to include a latent layer that models the correctness of the justifications. Latent-variable perceptrons have been proposed for several other NLP tasks (Liang et al. 2006; Zettlemoyer and Collins 2007; Sun et al. 2009; Hoffmann et al. 2011; Fernandes, Dos Santos, and Milidiú 2012; Björkelund and Kuhn 2014), but to our knowledge, we are the first to adapt them to reranking scenarios.

Finally, we round out our discussion of question answering systems with a comparison to the famous Watson QA system, which achieved performance on par with the human champions in the Jeopardy! game (Ferrucci 2012). Several of the ideas proposed in our work are reminiscent of Watson. For example, our component that generates text aggregation graphs (Section 5) shares functionality with the Prismatic engine used in Watson. Similar to Watson, we extract evidence from multiple knowledge bases. However, there are three fundamental differences between Watson and this work. First, whereas Watson includes components for evidence gathering and scoring (we call these **justifications**), it uses a fundamentally different strategy for evidence generation. Similar to most previous work, the textual evidence extracted by Watson always takes the form of a contiguous segment of text (Murdock et al. 2012),[1] whereas our justifications aggregate texts from different documents or knowledge bases. We demonstrate in this work that information aggregation from multiple knowledge bases is fundamental for answering the science exam questions that are our focus (Section 8). Second, our answer-ranking approach jointly ranks candidate answers and their justifications using a latent-variable learning algorithm, whereas Watson follows a pipeline approach where first evidence is generated, then answers are ranked (Gondek et al. 2012). We show in Section 8 that jointly learning answers and their justifications is beneficial. Last but not least, Watson was implemented as a combination of distinct models triggered by the different types of Jeopardy! questions, whereas our approach deploys a single model for all questions. Our analysis in Section 10 suggests that there are limits to our simple approach: We measure a ceiling performance for our single-model approach of approximately 70%. To surpass this ceiling, one would have to implement dedicated domain-specific methods for the difficult problems left unsolved by our approach.

3. Approach

The architecture of our proposed QA approach is illustrated in Figure 1, and proceeds in a stage-like fashion.

Prior to the QA task, in an offline process, we decompose sentences from six corpora into a lightly structured graphical representation (“graphlets”) that splits sentences on clausal and prepositional boundaries (Section 5). As shown later, this is fundamental to the creation and evaluation of answer justifications. All other stages of the framework occur online.

The QA pipeline receives as input questions with multiple choice answers, similar to the questions shown in Table 1, and proceeds as follows. First, the questions are fed into a focus word extractor (Section 4) that produces a weighted list of the keywords from both the question and answer candidates, sorted in descending order of their likely relevance to the information needed in the question. These keywords are used by the sentence aggregation component (Section 5) to create an exhaustive list of potential answer justifications for each answer candidate. These answer justifications are in the form of **text aggregation graphs** (TAGs), each composed of two to three graphlets produced by the preprocessing step.

After the sentence aggregation step, each of the multiple choice answers has a long list of candidate justifications. Because of the large number of candidate justifications created for each question–answer pair (typically tens or hundreds of thousands), we filter the list to include only the top *k* justifications based on an inexpensive score,

[1] Watson also generates “structured evidence,” which is obtained by converting texts to structured representations similar to logic forms, which are then matched against structured databases for answer extraction. However, this “logical representation of a clue and then finding the identical representation” in a database resulted in “confident answers less than 2% of the time” (Ferrucci 2012).

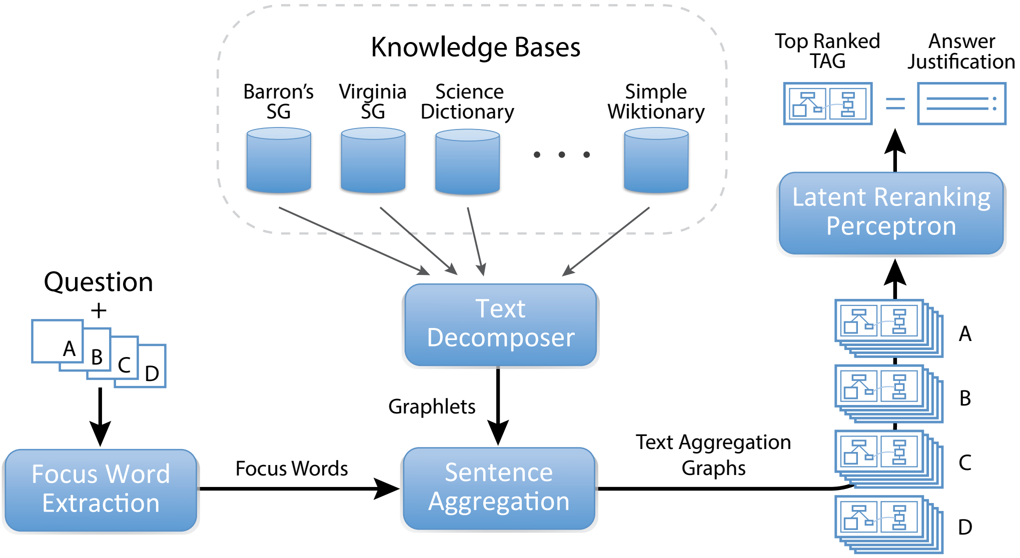

Figure 1
Our QA approach, which centers on the construction of answer justifications as text aggregation graphs, and ranking them using a model that treats the justification quality as a latent variable.

implemented as the sum of the weights of the focus words present in each justification.[2] Using the focus words, we extract a number of features from each candidate justification, which measure how well the justification answers the question (Section 6), and present this information to a latent-variable perceptron ranker (Section 7). This learning framework learns which answer candidate is correct based on the candidate justifications, while also jointly learning to rank justification quality as a latent variable, selectively elevating good answer justifications to the top of the list.

The candidate answer with the highest-scoring justification is taken to be the correct answer. Although the justification is expressed in the form of a text aggregation graph to make it easier to assemble and evaluate, we use the original sentences from the knowledge base used to construct that text aggregation graph as a human-readable justification for why that answer candidate is correct.

4. Focus Word Extraction

Questions often contain text that is not strictly required to arrive at an answer and that can introduce noise into the answering process. This is particularly relevant for science exams, where questions tend to include a narrative or example for the benefit of the reader, and this extra text can make determining a question's **information need** more challenging. For example, the question in Table 2 could be simplified to *What tools can be used to measure speed?*, but instead grounds the question in an example about turtles walking along a path. As a first step in answering, we need to identify whether the focus

[2] We keep ties in this filtered list, which, because of the simplicity of the score, may increase the size of the filtered lists considerably. For example, if $k = 25$, it is common that the filtered list includes between 100 and 1,000 candidate justifications.

Table 2
Focus word decomposition of an example question, suggesting the question is primarily about measuring the speed of walking, and not about turtles or paths. (Correct answer: "a stopwatch and meter stick.") For a given word: *Conc* refers to the psycholinguistic concreteness score, *Tag* refers to the focus word category (*FOCUS* signifies a focus word, *EX* an example word, *ATYPE* an answer-type word, and *ST* a stop word), *Score* refers to the focus word score, and *Weight* refers to the normalized focus word scores.

Words	What	tools	could	determine	the	**speed**	of	turtles	**walking**	along	a	path ?
Conc	2.0A	4.6C	1.3A	2.1A	1.4A	**3.6**	1.7A	5.0C	**4.1**	2.1A	1.5A	4.4C
Tag	ST	ATYPE	ST	ST	ST	**FOCUS**	ST	EX	**FOCUS**	ST	ST	EX
Score	–	1	–	–	–	**14**	–	2	**14**	–	–	3
Weight	–	0.03	–	–	–	**0.41**	–	0.06	**0.41**	–	–	0.09

of the question is about *speed*, *turtles*, *walking*, or *paths*, so that we can appropriately constrain our intersentence aggregation, and decrease the chance of generating noisy or unrelated justifications.

Note that the **focus word** terminology was first introduced in the context of factoid QA, where it represents the question word or phrase that is indicative of the expected answer type, which is then used to constrain the search for candidate answers (Harabagiu et al. 2000; Moldovan et al. 2003b). For example, *capital* is the focus word in the question *What is the capital of France?* This information is used to constrain the search for answers to entities that are of names of cities. In contrast, such words (e.g., *tools* for the previous exam question) are often of low importance for multiple choice science exams, as this information is already implicitly provided in the multiple choice answers. Instead, our task is to identify the central concept the question is testing (e.g., *measuring speed*), and to eliminate words that are part of an example or narrative (e.g., *turtle, path*) that are unlikely to contribute much utility (or, may introduce noise) to the QA process. However, because at a high level focus words identify the information need of a question, which is what we aim to do as well, we continue to use the same terminology in this work.

Our approach borrows from cognitive psychology, which suggests that elementary school students tend to reason largely with concrete concepts (i.e., those that are easy to mentally picture), because their capacity for abstract thinking develops much more slowly into adulthood (e.g., Piaget 1954). Recently Brysbaert et al. (2014) collected a large set of psycholinguistic concreteness norms, rating 40,000 generally known English lemmas on a numerical scale from 1 (highly abstract) to 5 (highly concrete). We have observed that highly abstract words (e.g., *expertise: 1.6, compatible: 2.3, occurrence: 2.6*) tend to be part of a question's narrative or too abstract to form the basis for an elementary science question, whereas highly concrete words (e.g., *car: 4.9, rock: 4.9, turtle: 5.0*) are often part of examples. Words that are approximately 50% to 80% concrete (*e.g., energy: 3.1, measure: 3.6, electricity: 3.9, habitat: 3.9*) tend to be at an appropriate level of abstraction for the cognitive abilities of elementary science students, and are often the central concept that a question is testing.[3]

Making use of this observation, we identify these focus words with the following algorithm. The algorithm is implemented as a sequence of ordered sieves applied in

[3] Although these concreteness thresholds may be particular to elementary science exams, we hypothesize that the information need of a question tends to be more abstract than the examples grounding that question. We believe this intuition may be general and applicable to other domains.

Table 3
Syntactic patterns used to detect answer type words. Square brackets represent optional elements.

Pattern	Example
(SBARQ (WHNP (WHNP (WDT) **(NN))** **[(PP)]**...	What *kind of energy* ...
(SBARQ (WHNP (WP)) (SQ (VBZ is) **(NP)**...	What is *one method* that ...
(S (NP (NP (DT A) **(NN)**) (SBAR (WHNP (WDT that)) ...	A *tool* that ..
(S (NP **(NP)** (PP)) (VP (VBZ is) ...	The *main function* of ... is to ...

decreasing order of precision (Lee et al. 2013). Each of the five sieves attempts to assign question words into one of the following categories:[4]

1. **Lists and sequences:** Lists in questions generally contain highly important terms. We identify comma delimited lists of the form `X, Y, ..., <and/or> Z` (e.g., *sleet, rain, and hail*). Given the prevalence of questions that involve causal or process knowledge, we also identify from/to sequences (e.g., *from a solid to a liquid*) using paired `prep_from` and `prep_to` Stanford dependencies (De Marneffe and Manning 2008).
2. **Focus words:** Content lemmas (nouns, verbs, adjectives, and adverbs) with concreteness scores between 3.0 and 4.2 in the concreteness norm database of Brysbaert et al. (2014) are flagged as focus words.
3. **Abstract, concrete, and example words:** Content lemmas with concreteness scores between 1.0 and 3.0 are flagged as highly abstract, and those with concreteness scores between 4.2 and 5.0 are flagged as highly concrete. Named entities recognized as either durations or locations (e.g., *In New York State*) are flagged as belonging to examples.
4. **Answer type words:** Answer type words are flagged using both four common syntactic patterns shown in Table 3, as well as a short list of transparent nouns (e.g., *kind, type, form*).
5. **Stop words:** A list of general and QA-specific stop words and phrases, as well as any remaining words not captured by earlier seives, are flagged as stop words.

Scoring and weights: We then assign a score to each question word based on its perceived relevance to the question as follows. Stop words are not assigned a score, and are not included in further processing. All answer type words are given a score of 1. Words flagged as abstract, concrete, or example words are sorted based on their distance from the concreteness boundaries of 3.0 (for abstract words) or 4.2 (for concrete words), where words closer to the concreteness boundary tend to be more relevant to the question, and should receive higher scores. These words are then given incrementally increasing scores starting at 2 for the most distant word, and increasing by one until all words in

[4] This same process is used to extract focus words for each of the multiple choice answer candidates, though it is generally much simpler given that the answers tend to be short.

this category have been assigned a score. To create an artificial separation between focus words and the less relevant words, and to encourage our intersentence aggregation method to preferentially make use of focus words, we assign all focus words a uniform score 10 points higher than the highest-scoring abstract, concrete, or example word. Finally, list words—the most important category—receive a uniform score one higher than focus words. The scores of all words are converted into weights by normalizing the scores to sum to one. An example of the scoring process is shown in Table 2.

It is important to note that the proposed focus word extraction algorithm is simple and leaves considerable room for improvement. We hypothesize that better algorithms could be implemented by switching to a learning-based approach. However, the simple unsupervised algorithm proposed requires no data annotations, and it captures the crucial intuition that some words in the question contribute more towards the overall information need than others. We demonstrate that this has a considerable impact on the performance of our overall QA system in the ablation studies discussed in Section 8.4.4.

5. Text Aggregation Graphs

Semantic drift, or the tendency for a graph traversal to drift to unrelated topics, constitutes a major hurdle for lexical semantic QA models. This was noted recently by Fried et al. (2015), who proposed an approximation of inference for QA as the traversal of a graph that connected concepts (words or lexicalized syntactic dependencies) along lexical semantic associations. Although this method is robust and performs better than approaches relying on alignment or embedding models alone, it does not have a good way of keeping an inference on-context when traversing the concept association graph. Moreover, even when a correct answer is selected, these methods operate on sets of probabilities over word distributions, and are unable to provide a compelling human-readable explanation to the user as to why a given answer is correct.

To address these issues, we propose to construct multisentence answer justifications using a sentence aggregation method that combines the robustness of word-level connections with sentence-level context. Our approach works in two steps, both of which are performed offline (i.e., before the actual QA process). First, we decompose sentences that are likely to justify science exam answers (e.g., from knowledge bases such as study guides) into smaller units based on clausal and prepositional phrase boundaries. Intuitively, this allows us to maintain sufficient context to control semantic drift, while mitigating the sparsity of complete sentences. Following previous QA terminology, we call these smaller units, which represent clauses or prepositional phrases, **information nuggets** (Voorhees 2003). We connect two information nuggets within the same sentence if there are any syntactic dependencies that connect any words in these nuggets. We call the resulting graph of these information nuggets, which represents an entire sentence, a **graphlet**. Figure 2 shows a number of example graphlets. Formally, we transform sentences into graphlets using the following algorithm:

1. **Parse knowledge base sentences:** All knowledge base sentences are syntactically parsed using the Stanford CoreNLP toolkit (Manning et al. 2014).

2. **Decompose sentences into information nuggets:** For each sentence, beginning at the root of the dependency graph, we traverse the dependency graph and segment the sentence into nuggets along clausal complements (`ccomp, xcomp`), adverbial and relative clause modifiers

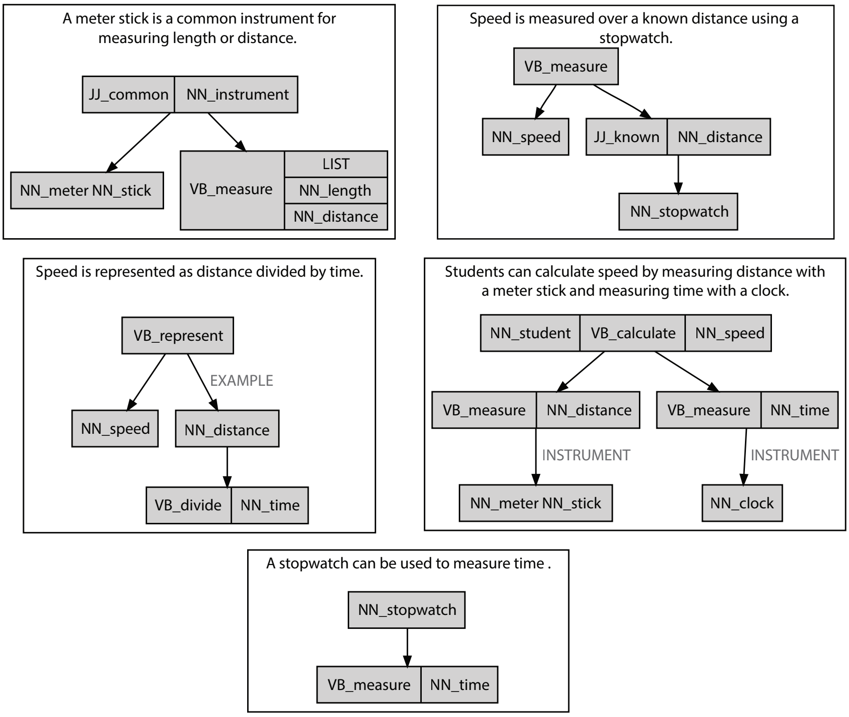
A meter stick is a common instrument for measuring length or distance.
JJ_common
NN_instrument
NN_meter NN_stick
VB_measure
LIST
NN_length
NN_distance
Speed is measured over a known distance using a stopwatch.
VB_measure
NN_speed
JJ_known
NN_distance
NN_stopwatch
Speed is represented as distance divided by time.
VB_represent
EXAMPLE
NN_speed
NN_distance
VB_divide
NN_time
Students can calculate speed by measuring distance with a meter stick and measuring time with a clock.
NN_student
VB_calculate
NN_speed
VB_measure
NN_distance
VB_measure
NN_time
INSTRUMENT
INSTRUMENT
NN_meter NN_stick
NN_clock
A stopwatch can be used to measure time .
NN_stopwatch
VB_measure
NN_time

Figure 2
Five example graphlets for five sentences that could be aggregated together in different combinations to justifiably answer the question *What tools could determine the speed of turtles walking along a path? (Answer: stopwatch and meter stick)*. Each graphlet contains one or more information nuggets (gray boxes) composed of one or more terms. For example, the graphlet for the sentence *A stopwatch can be used to measure time* contains two information nuggets. Edges between nuggets within a graphlet are shown with arrows, where a subset of these edges are labelled (e.g., *EXAMPLE, INSTRUMENT*), and the rest are unlabelled.

(`advcl`, `rcmod`), reduced non-finite verbal modifiers (`vmod`), and a subset of empirically determined prepositional phrases.[5]

3. **Segment nuggets into terms:** Each nugget is segmented into one or more terms. A term is nominally a single word, but we join both noun compounds and lists into single terms. Noun compounds (e.g., *meter stick* in Figure 2) are detected using the `nn` dependency tag, and lists of words (e.g., *sleet, rain, hail, or snow*) are detected through the same patterns used by the focus word extractor (see Section 4).

4. **Construct graphlets:** Two nuggets within a graphlet are connected if any of the words in one nugget have links in the syntactic dependency tree with words in the other nugget.

[5] The full list of prepositional dependencies along which we split is: `prep_with`, `prep_through`, `prep_from`, `prep_to`, `prep_as`, `prep_into`, `prep_by`, `prep_in`, `prep_such_as`, `prep_because_of`, `prep_over`, `prep_on`, `prep_between`, `prepc_as`, `prepc_by`, `prepc_of`, `prepc_with`, `prepc_after`, `prepc_for`, `prepc_before`, `prep_before`, `prep_after`, `prep_during`, `prepc_during`, `prepc_because_of`, `prep_without`, and `prepc_without`.

5. **Label a subset of the graphlet edges:** In general, to increase robustness, edges between nuggets are unlabeled. The exceptions to this are a small set of high-confidence labels: the label definition derived from the semi-structured dictionary resources that distinguish between a defined word and its definition, and the labels instrument, process, example, temporal, and contrast derived from prepositional dependencies.[6]

6. **Remove stop-words:** Finally, we remove any stop words, as well as words that are not nouns, verbs, or adjectives. Any nuggets that are empty after this filtering are removed.

After the sentences have been transformed into graphlets, we construct candidates for multi-sentence justifications by connecting sentences with any lexical overlap between information nuggets. We currently connect up to three graphlets into constructs we call **text aggregation graphs (TAGs)**, where each TAG corresponds to one potential answer justification. Figure 3 shows an example of two TAGs. Note that a TAG contains multiple levels of structure: (a) clause-level structure within nuggets, (b) sentence-level structure within graphlets, and (c) intersentence structure within TAGs.[7] We will exploit this information in the next section, where we design features to capture the fitness of a TAG as a justification for a given answer.

6. Text Aggregation Graph Features

Once candidate answer justifications have been reframed as text aggregation graphs, the TAGs for each answer candidate need to be ranked such that a good justification for the correct answer will be ranked above all other answer justifications for incorrect answers. We describe the features that we extract from TAGs for this ranking in this section, and the latent ranking algorithm in Section 7.

6.1 Features

We developed a set of TAG features that capture both the type of connections between the graphlets in a TAG and how well the TAG as a whole relates to the corresponding question–answer pair. The features can be broadly grouped into count features and mass features. **Count features** count the integer number of instances of a given event in a TAG—for example, the number of nuggets that are entirely focus words. **Mass features** sum the weights of focus words (as computed in Section 4)—for example, summing the weight of all question or answer focus words found either within a single graphlet, or across the entire TAG.

[6] More specifically, in our method the instrument label corresponds to the dependencies prep_with, prep_through, and prep_by. The label process corresponds to the dependencies prep_from, prep_to, prep_into, prep_because_of, and prepc_because_of. The example label corresponds to prep_as, prepc_as, prep_such_as, and prepc_such_as. The temporal label corresponds to prep_before, prepc_before, prep_after, prepc_after, prep_during, and prepc_during. Finally, the contrast label corresponds to prep_without and prepc_without.

[7] Currently, the graphlets and TAGs make use of lexical and syntactic structure only. Although this provides a robust representation of much of the structure in text, in a future system we would also like to explore adding semantic role and discourse information in our graphlets. Both of these have been shown to be useful, albeit for simpler QA tasks. (Surdeanu, Ciaramita, and Zaragoza 2011; Jansen, Surdeanu, and Clark 2014).

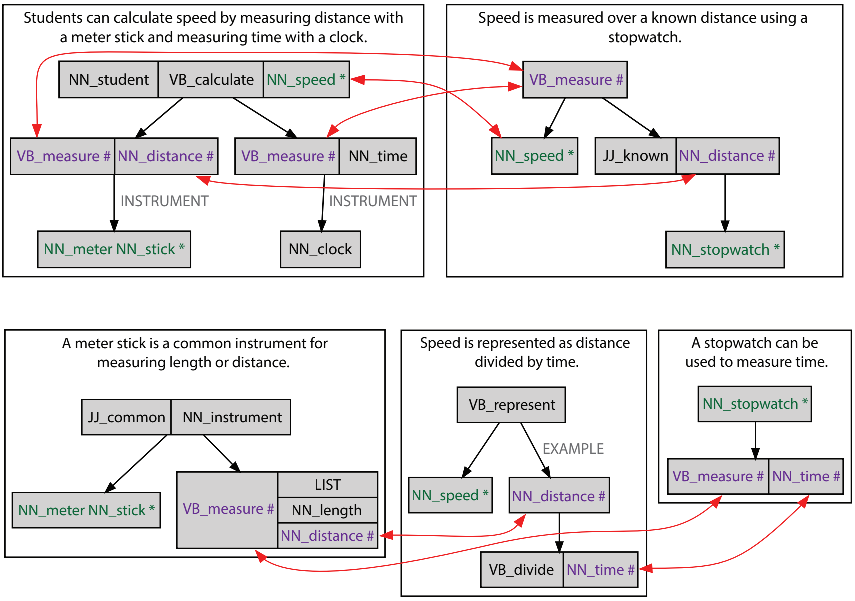

Figure 3
Two example TAGs that justifiably answer the question *What tools could determine the speed of turtles walking along a path* for the answer *a stopwatch and meter stick*. Asterisks (*) denote that a given term is either a question or answer focus word, and pounds (#) denote terms that are not found in the question or answer, but which are shared between graphlets. Links between the graphlets in a given TAG are highlighted. (Top) A two-sentence TAG, where the edges between graphlets connect on a focus word *(speed)* and other words shared between the graphlets *(distance, measure)*. (Bottom) A three-sentence TAG, where the edges between graphlets connect entirely on shared words between the graphlets *(distance, measure, time)* that are not focus words.

A full list of the features and their descriptions can be found in Table 4. The features are grouped into three categories: TAG-level, nugget-level, and bridge features. The TAG-level features use focus words to evaluate the likelihood that an entire justification is relevant to the question and answer candidate. Nugget-level features provide a finer-grained measure of how well the individual sentences in a justification relate to each other by quantifying both *how fully* they are connected (e.g., are all the words in a nugget matched with words in other nuggets, or only some), and what type of words they connect on (e.g., *focus words*, or *other words* not found in the question but shared between sentences). Separate features encode whether a nugget is all focus words, all shared words, all unmatched words, or any partial combination of these three categories. Finally, we define a graphlet that contains both question and answer focus words as a **bridge graphlet**, in that its content tends to span or *bridges* the question and answer, and include a set of bridge features for evaluating this subset of highly relevant graphlets.

6.2 Modeling Different TAG Types Using Domain Adaptation

Good justifications for an answer may take a variety of forms, from those with little lexical overlap due to the "lexical chasm" (Berger et al. 2000) between question and answer, to those with a great deal of overlap. For example, of the two TAGs shown in

Table 4
Features used to score candidate answer justifications represented as TAGs.

Feature	Description
TAG-level features	
numFocusQ	Number of unique Q focus words present in the TAG
numFocusA	Number of unique A focus words present in the TAG
massFocusQ	Sum of the unique Q focus word weights present in the TAG
massFocusA	Sum of the unique A focus word weights present in the TAG
numRepeatedFocus	Number of focus words contained in more than one graphlet, with repetition
numOtherAnswerF	Negative predictor: the number of focus words from other multiple choice answers included in the current TAG
minConcShared	The minimum psycholinguistic concreteness score (i.e., abstractness) of any shared word in the TAG
Nugget-level features	
numNugF	Number of nuggets that are entirely focus words
numNugFS	Number of nuggets that contain only focus words and shared words
numNugFSO	Number of nuggets that contain focus, shared, and other unmatched words
numNugFO	Number of nuggets that contain only focus words and other words
numNugS	Number of nuggets that contain only shared words
numNugSO	Number of nuggets that contain only shared words and other words
numNugO	Number of nuggets that contain only other words
numDefinedFocus	Number of nuggets containing only focus words with outgoing `definition` edge
numDefinedShared	Number of nuggets containing only shared words with outgoing `definition` edge
numQLinksFocus	Number of nuggets containing only focus words that have an incoming labeled link (e.g., `definition, instrument, process, temporal`)
numQLinksShared	Number of nuggets containing only shared words that have an incoming labeled link
numNuggetMultiF	Number of nuggets that contain more than one focus word
Bridge features	
massMaxBridgeScore	Bridge graphlets are graphlets that contain at least one focus word from both the question and answer, signifying that they are highly relevant to the question and the corresponding answer. A graphlet's bridge score is the sum of this focus word mass. We calculate the minimum, maximum, and delta (max − min) bridge scores across all graphlets within a TAG
massMinBridgeScore	
massDeltaBridgeScore	

Figure 3, in the first TAG both sentences connect on question focus words, whereas in the second TAG the graphlets only connect on other non-focus words.

Understanding the connections between sentences in a TAG serves as a robust proxy to modeling discourse structures (e.g., whether the second sentence elaborates on the concepts introduced in the first), which has been previously shown to increase both open-domain and science-domain QA performance (Jansen, Surdeanu, and Clark 2014). This is important in the context of feature representations because we conjecture that some features may be important across all the ways a justification can connect (so these features should be jointly modeled across connection types), whereas others are specific to certain connection types (so they should be modeled separately by type). As shown in Section 8, this hypothesis is strongly empirically supported.

To model this phenomenon, we adapt a technique from the field of domain adaptation (Daumé III 2007). First, we label each of the sentences within a TAG based on whether they contain question focus words and/or answer focus words. We then characterize the entire TAG by determining whether the words shared between sentences are question focus words, answer focus words, or other non-focus words. This

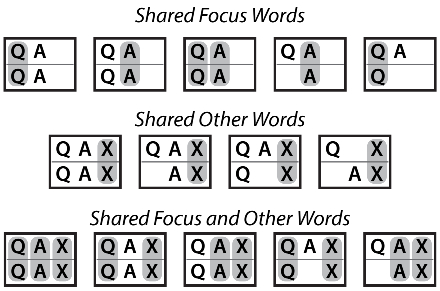

Figure 4
Connections between sentences are characterized based on lexical overlap between graphlets. Here, each box represents a two-sentence TAG, with graphlets stacked vertically. The presence of question or answer focus words is marked with *Q* or *A*, and the presence of other non-focus words shared between the two graphlets is marked with *X*. Lexical overlap within a category is highlighted.

leads to 14 possible connection types, depicted in Figure 4.[8] Second, following Daumé's method, we generate 15 versions for each of the features introduced in Section 6.1: a generic one that is type independent, and 14 that are affixed by each of the connection types. For example, for a given feature such as *numFocusQ*, we create 15 copies: *numFocusQ*, $numFocusQ_1$ ... $numFocusQ_{14}$, where the subscript (when present) indicates a specific connection type. For a TAG of connection type *i*, only values for the type-specific feature $numFocusQ_i$ as well as the general feature *numFocusQ* are populated—all other $numFocusQ_j$ features are set to a value of zero. This allows the learning algorithm in the next section to learn whether each feature generalizes across connection types, or not. As shown in Finkel and Manning (2010), this approach is equivalent to a joint model with a hierarchical prior.

The features introduced in Table 4 apply to justifications containing any number of sentences, but characterizing justifications that are longer than two sentences is not straightforward, as the number of connection types (Figure 4) would become prohibitively large. We handle three-sentence justifications by treating each as a set of three two-sentence TAGs, and characterize each two-sentence connection individually. In the case where two groups of sentences have the same connection type, we take the highest scoring version of each overlapping feature. Although this method would extend to justifications of arbitrary length, justifications longer than three sentences are not currently generated by our system.

7. Learning Model

We perform answer selection by focusing on ranking answer *justifications* rather than actual answers. Our algorithm (detailed in Algorithm 1) determines the best answer to a given question by first finding the highest-scoring TAG (i.e., justification) out of all the

[8] In practice, for a configuration that allows TAGs of one or two sentences, we have 15 connection types, including the 14 two-sentence connection types, plus a single-sentence type. We ignore the single-sentence type in this discussion for simplicity, though it is included in our model.

TAGs for each of its candidate answers, and then selecting the corresponding answer candidate. This introduces an additional complication during training: Although we know which answer candidate is correct, the actual quality of any given TAG is hidden. That is, many TAGs that connect a correct answer to the question do so in the wrong context, producing a poor justification. Conversely, even an incorrect answer can be connected back to the question; otherwise the multiple choice test itself would be too easy.

We address this issue with a reranking perceptron (Shen and Joshi 2005; Surdeanu, Ciaramita, and Zaragoza 2011) extended with a latent layer that models the correctness of the justifications. During training, we take advantage of the assumption that, with enough knowledge coverage, the best TAG for a correct answer will be better than the best TAG for an incorrect one. As a result, our algorithm optimizes two goals jointly: choosing a good answer for a given question, and justifying it with a human-readable TAG that connects sentences from our knowledge bases.

7.1 Learning Algorithm

The input to the reranking algorithm consists of a corpus of n training questions, $\mathcal{Q}$, where each $q_i \in \mathcal{Q}$ has a set of m candidate answers, $\boldsymbol{A}$ (in the particular case of the multiple choice test, these are the four multiple choice answers). Each $a_j \in \boldsymbol{A}$ for a given q_i has a set of TAGs, $\boldsymbol{X_{i,j}}$, which connect q_i to a_j. Crucially, for a correct answer, we assume that there is *at least one* valid TAG $x_c \in \boldsymbol{X_{i,j}}$ that justifies the correctness of the chosen answer. This assumption is similar to the one made by Hoffmann et al. (2011), who assumed that, for a binary relation extraction task, for each training relation between two entities, e_1 and e_2, there exists at least one correct sentence that supports the extraction of the given relation among the sentences that contain both e_1 and e_2.

Algorithm 1 Learning algorithm for the latent reranking perceptron. We consider, without loss of generality, that the correct answer appears at position 1 in training.

```
Input:
T – the number of training epochs
Q – the set of n training questions, each of which has m answer candidates
X_{i,j} – the set of all TAGs connecting a question q_i with a candidate answer a_j
Output: $\boldsymbol{\Theta}$ – parameter vector
$\boldsymbol{\Theta} = \boldsymbol{0}$
for $t \leftarrow 1$ –to– $T$ do
    for $i \leftarrow 1$ –to– $n$ do
        $k = \arg\max_{j=1,\ldots,m} F(q_i, a_j)$
        if $k \neq 1$ then
            for $x \leftarrow P(q_i, a_1)$ do
                $\boldsymbol{\Theta} = \boldsymbol{\Theta} + \Phi(x)$
            end for
            for $x \leftarrow P(q_i, a_k)$ do
                $\boldsymbol{\Theta} = \boldsymbol{\Theta} - \Phi(x)$
            end for
        end if
    end for
end for
return $\boldsymbol{\Theta}$
```

The learning algorithm is formalized in Algorithm 1. The two fundamental building blocks of the algorithm are the functions P and F. Intuitively, the function P identifies which justifications (or TAGs) for a given answer best explain the answer according to the current model. For example, for the question *Which organism is a producer?* and the answer *grass*, a good justification selected by function P is a TAG that connects the two sentences: *Producer is an organism that produces its own food and is food for other organisms: usually a green plant.* and *Grass is a green, leafy plant that often covers the ground.* Because P implements a latent operation (i.e., which justification is correct?), its goodness evolves together with the learned model. Note that, at this stage, we do not control how many justifications are produced by function P for a given answer other than constraining it to select *at least one*. The function F computes the overall score of a given answer by simply averaging the scores of the justifications chosen by P.

More formally, for a given question q_i and candidate answer a_j, P chooses a subset of TAGs from $\boldsymbol{X}_{i,j}$ that are deemed correct, according to the current model. F marginalizes over these TAGs to compute the overall score of the answer a_j:

$$F(q_i, a_j) = \frac{\sum_{x \in P(q_i, a_j)} \Theta \cdot \Phi(x)}{|P(q_i, a_j)|} \tag{1}$$

where the score of a given TAG, x, is computed as the dot product between the parameter vector $\boldsymbol{\Theta}$ and the vector produced by a feature function, Φ, which operates over individual TAGs.

Given the functions P and F that control the latent operation of choosing valid answer justifications, the algorithm proceeds similarly to the reranking perceptron (Shen and Joshi 2005). If a prediction (i.e., a choice for the correct answer for a given question [line 6]) is incorrect (line 7), the algorithm updates its parameter vector, $\boldsymbol{\Theta}$, by adding all valid justifications of the correct answer as produced by P (line 9), and subtracting all TAGs considered valid for the current top answer (line 12).

The fundamental operation here is thus the implementation of P. For their classification task, Hoffmann et al. (2011) implement this function as an edge-cover problem, which guarantees that the overall score for the correct label (similar to our F function) is maximized and at least one sentence supports the correct classification. For our reranking task, we found that a conservative interpretation of this *at least one* idea, where *exactly* one justification is chosen, performed the best.[9] That is, P returns only the top TAG with the highest score according to the dot product with the current $\boldsymbol{\Theta}$. We discuss the other extreme, that is, using all TAGs, in Section 8.4.4 (when taken to this level, the latent layer is essentially disregarded entirely, assuming that each TAG for a correct answer is correct, and that each TAG for an incorrect answer is incorrect). As discussed there, this performed far worse.

Inference: During inference, the algorithm ranks all candidate answers for a given question in descending order of their score (as given by F, but using the averaged parameter vector, $\boldsymbol{\Theta}_{avg}$) and returns the top answer.

Practical concerns: Two practical issues were omitted in Algorithm 1 for clarity, but improved performance in practice. First, we used the averaged percetron at inference

[9] In fact, we found that performance consistently dropped as the number of justifications chosen by P increased.

time (Collins 2002). That is, instead of using the latest $\boldsymbol{\Theta}$ after training, we averaged all parameter vectors produced during training, and used the averaged vector, $\boldsymbol{\Theta}_{avg}$, for inference.

Second, we used a large-margin perceptron, similar to Surdeanu, Ciaramita, and Zaragoza (2011). In particular, we update the model not only when the predicted answer is incorrect (line 5), but also when the current model is not confident *enough*—that is, when the predicted answer is correct, but the difference in F scores between this answer and the second predicted answer is smaller than a small positive hyper parameter τ.

8. Experiments

8.1 Data

Questions: We assembled a corpus of 1,000 third to fifth grade standardized elementary school science exam questions, consisting of 346 publicly available questions gathered from standardized exams in 12 states, as well as 654 closed questions from an exam-generating service. All questions are multiple choice with four possible answer candidates. Questions vary in length from 1 to 6 sentences, and the four multiple choice answer candidates are generally either single words or short phrases. Because of the small size of this corpus, our models were evaluated using 5-fold crossvalidation, with three folds for training, one for development, and one for test.

Knowledge bases: Sentences from six text resources served as input for TAG generation. Five of these resources are in the science domain and include two state-specific science exam study guides, a teacher's manual, a children's science dictionary, and a set of exam review flashcards. Each of these in-domain resources was selected to be at a third-to-fifth grade knowledge level, and contained between 283 and 1,314 sentences (for a total of 3,832 in-domain sentences). The Simple Wiktionary[10] was included as a large open-domain dictionary resource written at knowledge level similar to that of the exam questions. Starting with over 24,000 definitions, we filtered these to include only definitions for nouns, verbs, and adjectives, for a total of 17,473 sentences after filtering.

8.2 Tuning

We tuned the following hyperparameters once on the development data to optimize both performance and stability.

Number of candidate justifications: Each of the multiple choice answers can have a large number of candidate justifications. To reduce runtime and assist learning, we filter this initial list of TAGs to include only a subset of TAGs with a high focus word mass. We kept all justifications tied for focus mass with the justification in 25th place, resulting in a variable number of TAGs for each QA pair. For our data set, the mean number of TAGs for each QA pair was 141.

Perceptron hyperparameters: We investigated the perceptron hyperparameters using a coarse grid search and found a stable optimum with 10 epochs, a τ of 1.0, burn in of 5 epochs (i.e., the weight updates were not added to the average weights for the first five

[10] `http://simple.wiktionary.org`.

epochs), and a learning rate (which dampened the updates to the weight vector) of 0.1. Because model results can vary depending on the random seed used for initialization, rather than using a single perceptron we use an ensemble of 50 perceptron models initialized with random weights. These models are combined in a simple voting scheme, where each model casts a single vote for each question (distributing the vote only in the case of ties).

Feature normalization: To minimize the effect of outliers in the feature space, we log-transformed the feature values, and then rescaled each feature independently to lie within the range of −1 to 1, using the formula $normalized = lower + (original - min)\frac{(upper-lower)}{(max-min)}$, where *upper* and *lower* are the desired boundaries for the normalization (1 and −1, respectively), *max* and *min* are the maximum and minimum values for the corresponding feature across the training data set, and *normalized* is the result of normalizing *original*. Note that during testing it is possible to see feature values outside of their known $[min, max]$ interval, which means that after normalization their values will fall outside the $[-1, 1]$ interval. We do not do any additional post-processing for these values.

8.3 Baselines

We include the following baselines:

Random: Selects an answer randomly.

Candidate retrieval (CR): Ranks answers using an approach similar to the baseline model in Jansen et al. (2014), which uses features that measure cosine similarity over *tf-idf* vectors (e.g., Manning et al. 2008, ch. 6) to rank answer candidates in a "learning to rank" framework. Traditionally, with retrieval systems, short answers to questions are dynamically constructed from larger parent documents, and the top-scoring answer (defined as the answer with the highest *tf-idf* score between that answer candidate and a query vector made from the question) is taken to be the winner. Here we adapt this setup to multiple choice exams by: (a) using query vectors that contain words from both the question and multiple choice answer candidate, and (b) generating features from the top *tf-idf* score for each answer candidate in a given question, which are then combined in the learning-to-rank framework. We then take the short passage retrieved by the system as a justification for why that answer candidate is correct.

Documents are constructed across the six knowledge bases by dividing each corpus by subsection (for texts), by definition (for dictionaries), or by flashcard. We implemented the document indexing and retrieval system using Lucene.[11] Each sliding window of N sentences in a given document served as a potential answer justification. Using the development questions, we empirically determined that a two-sentence window performed best, though performance did not significantly differ for window sizes between one and five sentences. This model generates two features: a cosine similarity score between the query vector and the best-scoring candidate answer justification in a given corpus, and a linear model that combines this cosine similarity with the cosine similarity between the query vector and the entire document, blending both local (answer justification) and global (document) context.

[11] `http://lucene.apache.org`.

Because our six corpora are of different genres (study guide, teachers guide, dictionary, flashcards), domains (science-domain vs. open-domain), and lengths (300 to 17,000 sentences), we implement six separate *tf-idf* models, each containing documents only from a single corpus. We then combine the two retrieval features from each model (12 features total) into a ranking perceptron (Shen and Joshi 2005; Surdeanu, Ciaramita, and Zaragoza 2011) to learn which knowledge bases are most useful for this task. This ensemble retrieval model produces a single score for each multiple choice answer candidate, where the top-scoring answer candidate is selected as the winner. The top-scoring answer justifications from each of the six retrieval models then serve as justifications.

Jansen et al. (2014): The best-performing combined lexical semantic (LS) and CR model of Jansen et. al (2014), which was shown to perform well for open domain questions. Similar to the CR model, we adapted this model to our task by including six separate recurrent neural network language models of Mikolov et al. (2013, 2010), each trained on one of the six knowledge bases. Two features that measure the overall and pairwise cosine similarity between a question vector and multiple choice answer candidate vector are included. The overall similarity is taken to be the cosine similarity of the composite vectors of both the question and answer candidate, obtained by summing the vectors for the individual words within either the question or answer candidate vectors, then renormalizing these composite vectors to unit length. The pairwise similarity is computed as the average pairwise cosine similarity between each word in the question and answer candidate. Two features from each of the six LS models are then combined with the two features from each of the CR models (24 features total), as above, using a ranking perceptron, with the top-scoring answer candidate taken as correct. Because the LS features do not easily lend themselves to constructing answer justifications, no additional human-readable justifications were provided by this model.

8.4 Results

Here we first investigate the performance of two variants of the TAG model with respect to justification length. We then compare the best-performing model with the baselines, and show that the TAG and CR models can be combined to increase performance. We use the standard implementation for precision at 1 (P@1) (Manning, Raghavan, and Schütze 2008) and a tie-aware implementation of mean reciprocal rank (MRR) (McSherry and Najork 2008).

8.4.1 Justification Length. Table 5 shows the performance of the TAG model as a function of increasing justification length. Here, the *k*G models contain exclusively justifications of length *k* (we explore TAGs of varying length later in this section). Short two-sentence (or two-graphlet) TAGs significantly outperform single-sentence TAGs, with single-sentence (1G) TAGs starting at 35.3% P@1, increasing to 38.2% for two-sentence (2G) TAGs, then decreasing slightly to 37.8% for three-sentence (3G) TAGs. In previous work, we observed that for a variety of word-level graphs and traversal algorithms, QA performance tends to rapidly peak when aggregating two or three words, then slowly decreases as more words are aggregated because of "inference drift" in the graph traversal process (i.e., following the connections from *breakfast* → *hashbrowns* → *potato* → *field* would potentially connect questions about breakfast to information about potato or even soccer fields) (Fried et al. 2015).

Though our sentence aggregation model contains far more structure than the higher-order LS graphs of Fried et al. (2015), and is represented at the level of the

Table 5
Performance as a function of justification length in sentences (or, the number of graphlets in a TAG) for two models: one aware of connection-type, and one that is not. **Bold font** indicates the best score in a given column for each model group.

Model	P@1	P@1 Impr.	MRR	MRR Impr.
Normal				
1G	35.33	–	59.26	–
2G	**38.16**	**8.0%**	**61.33**	**3.5%**
3G	37.78	6.3%	61.14	3.2%
Connection-type aware (Daumé)				
$1G_{CT}$	34.80	–	58.86	–
$2G_{CT}$	**39.91**	**14.7%**	**62.53**	**6.2%**
$3G_{CT}$	38.53	10.7%	61.65	4.7%

sentence or graphlet rather than individual lemmas, we hypothesize based on this previous work that we may be observing the beginning of the same characteristic peak in performance reported there. Because runtime increases exponentially with the number of sentences included in a TAG, it quickly becomes intractable to test this with TAGs containing more than three graphlets.

Extending the model to include a knowledge of the connection type (or the type of lexical overlap, see Section 6.2) between sentences in a given TAG using Daumé's method (Daumé III 2007) increases performance, suggesting that different kinds of connections are best identified through different feature patterns. Here, the connection-type aware $2G_{CT}$ model outperforms the regular 2G model by nearly 2% P@1 (absolute), increasing performance to 39.9%—an increase of +14.7% (relative) over using only single-sentence TAGs.[12]

8.4.2 Combined Models. Where Table 5 lists models that contain justifications of static length, Fried et al. (2015) showed that combining paths of different lengths into a single classifier can increase performance. The performance of TAG models that combine justifications of different lengths, as well as the baseline models, is shown in Table 6.

Baselines: Where the lexical semantics model of Jansen et al. (2014) outperformed their CR baseline by +36% (relative) on a large corpus of open-domain questions from Yahoo Answers, here on elementary science exams, the lexical semantic features decrease performance. Our conjecture is that the performance difference is due to the difference in the size of the corpora used to train the lexical semantic model—where Jansen et al. trained their lexical semantic model using Gigaword, here we are limited by the relatively small size of our text resources. Jansen et al. (2014) reported that a domain-specific version of their lexical semantic model performed poorly when trained on a

[12] Each of the 50 models in the ensemble reranker is initialized with random weights, causing the small performance difference between 1G and $1G_{CT}$.

Table 6
Performance of the baseline and best-performing TAG models, both separately and in combination. TAG justifications of different short lengths were found to best combine in single classifiers (denoted with a +), where models that combine the CR baseline or long (3G) TAG justifications best combined using voting ensembles (denoted with a ∪). **Bold font** indicates the best score in a given column for each model group. Asterisks indicate that a score is significantly better than the highest-performing baseline model (* signifies $p < 0.05$, ** signifies $p < 0.01$). The dagger indicates that a score is significantly higher than the score in the line number indicated in superscript ($p < 0.01$). All significance tests were implemented using one-tailed non-parametric bootstrap resampling using 10,000 iterations.

#	Model	P@1	P@1 Impr.	MRR	MRR Impr.
	Baselines				
1	Random	25.00	–	52.08	–
2	CR	**40.20**	–	**62.49**	–
3	Jansen et al. (2014)	37.30	–	60.95	–
	Combined models with justifications of variable lengths (Single classifier)				
4	1G + 2G	38.69	–	61.43	–
5	$1G_{CT} + 2G_{CT}$	**42.88**$\dagger^{4}$	**+6.7%**	**63.94%**	**+2.3%**
	Combined models that include the CR baseline (Voting)				
6	$CR \cup 1G_{CT} \cup 2G_{CT} \cup 3G_{CT}$	43.15*	+7.3%	64.51*	+3.2%
7	$CR \cup (1G_{CT} + 2G_{CT}) \cup 3G_{CT}$	**44.46****	**+10.6%**	**65.53****	**+4.9%**

biology textbook and subset of Wikipedia, and others have since shown that lexical semantic models perform poorly with small amounts of training data (Sharp et al. 2015).

Combined models: Single-classifier models containing both 1G and 2G TAGs were generated for both the normal and connection-type-aware models. The 1G + 2G model performs only slightly better than 2G alone, but when the models are connection-type aware, there is greater benefit to combining the different path lengths—the connection-type-aware $1G_{CT} + 2G_{CT}$ model (line 5) increases performance to 42.9% P@1 (compare with the static-length $2G_{CT}$ performance of 39.9%).

As the CR baseline and the TAG models are performing inherently different tasks (information *retrieval* and information *aggregation*, respectively), we are able to combine them in a voting model in order to create a full system that contains the benefits of each. The voting models that incorporate both the CR baseline and TAG models across all justification lengths are included on lines 6 and 7. Both models significantly increase performance over the CR baseline, with the voting model that couples $1G_{CT} + 2G_{CT}$ as a single classifier (and single vote) performing better than when $1G_{CT}$ and $2G_{CT}$ vote separately. This best-performing model reaches 44.5% P@1, increasing performance over the CR baseline by +10.6% (relative). This set of experiments demonstrates that our approach of jointly ranking answers and justifications is complementary to a strong information retrieval baseline, and significantly improves performance at the task of selecting the correct answer.

Table 7
Example justifications from the CR baseline and their associated ratings.

Question
What is the interaction between the producer and the consumer in a food chain? **[A] The consumer eats the producer for energy.** [B] The consumer does not interact directly with the producer. [C] The producer eats other producers for energy. [D] The producer eats the consumer for energy.

Rating	Example Justification
Good	A primary (1st) consumer eats producers (plants). A secondary (2nd) consumer eats primary consumers. *[Barrons SG]*
Half	The food chain starts with a producer (a plant) and ends with a decomposer. *[Flashcards]*
Topical	A herbivore is an organism that depends on plants for most of its food and energy. *[Science Dictionary]*
Offtopic	When a plate moves suddenly a great amount of energy is released. These waves cause damage... *[Virginia SG]*

8.4.3 Justifications. The previous experiments demonstrate that our method performs well at identifying correct answers. But how well does it perform at the task of *justifying* those answers? We evaluated justification performance for both the best baseline (CR) and the best performing TAG model that is independent of CR (i.e., $1G_{CT}$ + $2G_{CT}$). Where TAG justifications took the form of the sentences being aggregated, justifications for the CR model were taken to be the highest-scoring short passages from each of the six knowledge bases. As the CR model was tuned to a retrieval size of two sentences to maximize P@1 performance, each CR justification was two sentences in length.[13] To facilitate easy comparison, because the CR model provides six justifications (one from each knowledge base), we evaluate the top six scoring TAG justifications.

The correctness of justifications was independently annotated by two of the authors. Any detected conflicts were resolved post hoc by the two annotators working together. Answer justifications were rated on a four-point scale, based on their ability to provide a convincing justification to the user as to why a given answer choice was correct (see Table 7 for CR examples, and Figure 5 for a TAG example). Justifications rated as **good** describe the inference required to arrive at the correct answer. Those rated as **half** contained at least a portion of an explanation of the inference, but missed some critical aspect required to answer the question, like discussing producers but not consumers in Table 7. The two final ratings are for justifications that did not address the inference – **topical** justifications do not address the question but discuss a similar topic, where **offtopic** justifications are unrelated to the question.

We evaluated the top-rated answer justifications using an **at least one** method. With this method, performance reflects the highest rating attained by *at least one* of the six justifications for a given question. For example, for a question to be classed as *good*, at least one of the six answer justifications must be rated as *good*. *At least one* answer

[13] Documents for the dictionary and flashcard corpora typically contained only a single sentence, and so answer justifications from these corpora are often shorter than two sentences.

Question	Which organism is a producer? (GR:5)
Focus Word(s)	(NN_producer, 0.92) (NN_organism, 0.08)
Answers	(A) frog (B) mushroom (C) grass (D) lizard
Correct Answer	grass
Focus Word(s)	(NN_grass, 1.00)

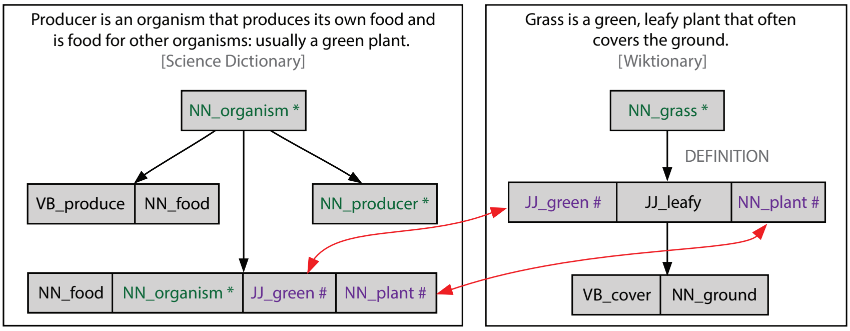

Figure 5
An example TAG and justification rated as *good*. The two sentences connect on non-focus "other" shared words (e.g., *green, plant*) that are not found in the question or answer, but that are highly related to the focus words.

justification performance is listed in Table 8. For the TAG model, 56.7% of the questions had at least one justification rated as *good*, outperforming CR justifications by 11.4% (absolute).

This experiment supports our original intuition that justifications must be aggregated from multiple resources. Although the small window of sentences from the CR model is sufficient to justifiably answer many questions, a large number of questions require knowledge to be aggregated from *non-adjacent* sentences within a given corpus, or from sentences in different corpora altogether, to compose convincing answer justifications. While the CR model leaves many of these questions with only partial justifications (34.9% of justifications are rated as *half*), the TAG model is able to aggregate sentences from multiple sources, and finds complete *good* justifications for many of the questions only partially answered by the CR model.

8.4.4 Ablation Studies. To verify the contribution of the components of our system, we include the following ablation studies:

Latent perceptron: The complete removal of the latent layer (i.e., using the average of all TAG scores to score the candidate answer, and performing perceptron updates

Table 8
At least one justification performance for both CR and TAG models, reflecting the highest rating attained by at least one of the top six justifications for a given question.

Rating	CR	TAG
Good	45.3%	56.7%
Half	34.9%	20.7%
Topical	11.9%	14.4%
Offtopic	7.9%	8.2%

with all TAGs) decreases the performance of the best performing TAG model ($1G_{CT}$ + $2G_{CT}$) from 42.88 to 35.43 P@1. Alternatively, we experimented with using the sum of the TAG scores and the maximum TAG score as the candidate score, while still doing updates with all TAGs. These configurations decreased performance to 34.46 and 38.09 P@1, respectively. This demonstrates the importance of modeling justification quality as a latent variable.

Focus words: Focus words are used both to find relevant sentences to aggregate into answer justifications, as well as to characterize those justifications when expressed as TAGs. Replacing focus words with uniform weights for all content words (NN, VB, JJ) in a question reduces performance of the 1G+2G model from 38.69 to 33.89 P@1. For the connection-type-aware model ($1G_{CT}$ + $2G_{CT}$), performance decreases from 42.88 to 40.03 P@1. This supports our observation that science exam questions contain several layers of information (e.g., the underlying question, and the example the question is grounded in), each contributing different utility to the QA process.

Graphlet structure: Graphlets segment sentences based on clausal and prepositional boundaries to facilitate evaluating how well two sentences connect using structures larger than a word but more fine-grained than the entire sentence. In other words, graphlets are the key structure in our representation of answer justifications because they drive both intra- and intersentence connections in a TAG. Eliminating this structure (i.e., considering each sentence as a bag of words, which is equivalent to a graphlet with a single nugget) substantially reduces the performance of the 1G + 2G model from 38.69 to 28.90 P@1 and the performance of the $1G_{CT}$ + $2G_{CT}$ model from 42.88 to 28.08 P@1.

8.5 From Latent Perceptron to Latent Neural Networks

Neural networks (NNs) have recently received much attention in many NLP tasks and have been shown to be quite effective in certain question answering tasks (Bordes, Chopra, and Weston 2014; Iyyer et al. 2014; Bordes et al. 2015; Iyyer et al. 2015; Dong et al. 2015; Wang and Nyberg 2015; Yih et al. 2015; He and Lin 2016; Suggu et al. 2016). To determine if a deep learning approach would benefit our system, we explored providing our text aggregation graph features and the candidate retrieval scores to a neural ranking architecture. Under this framework we were also able to straightforwardly experiment with including vectorized representations (i.e., embeddings) of the question, candidate answer, and corresponding justification, in hopes of accommodating some of the lexical variation in natural language.

Neural network systems have varied dramatically in terms of their primary architecture (feed-forward, convolutional, recurrent, etc.) as well as their shape (number of hidden layers, number of nodes in a given layer, etc.). Recently, however, Iyyer et al. (2015) found that in their QA task, a simple network that *averaged* the word embeddings of the questions and the answer candidates outperformed a much more complex tree recurrent NN. Chen and Manning (2016) have recently validated this observation in a different QA task. We based our NN on this simpler system, adapting it to our task by including not only an averaged vector for the question and answer, but also an averaged vector for the answer justification. Additionally, we include our connection-type aware structural TAG features (cf. Table 4) and the CR features in the network input. The network architecture is shown in Figure 6.

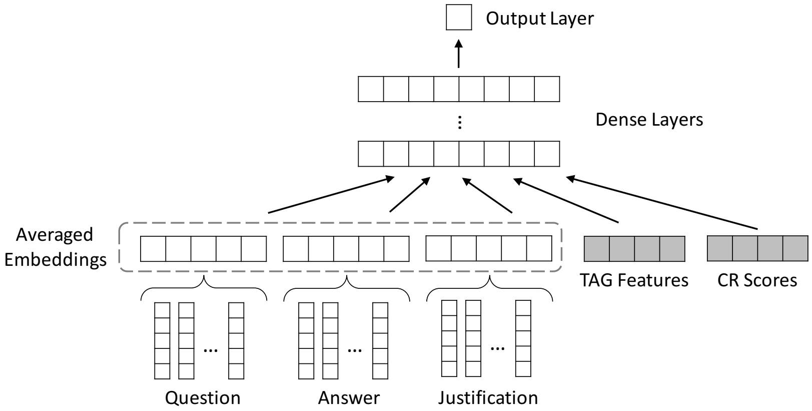

Figure 6
The architecture for the neural network variation of our TAG system. We use a fully connected feed-forward network that takes as input the content words (i.e., nouns, verbs, and adjectives) from the question, candidate answer, and corresponding justification. For each of these, we create a composite vector by averaging the embeddings of the individual words. We then concatenate these three composite vectors, as well as the TAG features and the CR scores, to form the input layer to the network. The output of the network is a single, real-valued score for the candidate answer justification.

For learning, we use the hinge ranking loss function (Collobert et al. 2011) and update using stochastic gradient descent. During training, for each question we pair the correct answer candidate with each of the incorrect candidates, creating three pairs. For each of these pairs, we score both the correct answer candidate and the incorrect answer candidate and then compute the loss:

$$L = \max(0, m - F(q, a_{correct}) + F(q, a_{incorrect})) \tag{2}$$

where $F(q, a_{correct})$ is the score of the correct answer candidate, $F(q, a_{incorrect})$ is the score of the incorrect answer candidate, and m is the margin.

There are two important aspects of the proposed architecture:

a) The latent layer—that is, identifying good justifications to use for a given answer, which we implement by modifying stochastic gradient descent to keep track of the latent aspect. Specifically, to maintain the latent variable aspect of our ranking perceptron in our ranking neural network, we adapt Equation (1) to use a feed-forward pass through the network to determine the score of a given TAG (rather than the inner product of the features and the parameter vector), and continue to use the implementation of P that uses only the TAG with highest score. In this way, for each pair of candidate answers we have two justifications (one for the correct answer and one for the incorrect), and so perform a model update (i.e., backpropagation) for each.

b) The combination of embeddings with explicit features that come from the CR system and the TAG features. This strategy of mixing latent and explicit features has been demonstrated to be successful in other NLP tasks (Chen and Manning 2014; Suggu et al. 2016).

Despite their advantages, neural networks have many hyperparameters that need to be tuned. Continuing the inspiration of Iyyer et al. (2015), we lightly turned the network on development, both in terms of network shape as well as additional parameters. Doing so, we arrived at a network that has a single dense layer with length 100 followed by the output layer of length one.[14] For our word embeddings, we used a recurrent neural network language model (Mikolov et al. 2010, 2013) trained over a concatenation of all of our in-domain elementary science resources (i.e., the study guides, flashcards, and dictionaries) to generate embeddings with 200 dimensions.[15]

All network nodes use the sigmoid activation function, which performed better and was more stable to variations in the hyperparameters than the rectified linear unit activation. Additionally, we used an L2 regularization of 1.0 and a dynamic learning rate that began at 1.0 and decayed by half each time the performance on validation decreased. Though we experimented with dropout, there did not seem to be a consistent improvement, so our final models do not include it. We used 50 epochs for training with early stopping if the validation performance decreased and failed to regain its previous best after 10 epochs. Similar to the perceptron, we found that the network performance varied with the random seed. To mitigate the effects of this variation, we report scores from an ensemble of five networks, each with a different random seed, where the trained networks each voted for a single answer choice, splitting their vote only in the event of a tie.

Neural Network Results. The final performance for the neural network variants of our proposed system as well as the CR baseline are shown in Table 9. Once again, we observe that the performance of the combined model (CR + TAG + embeddings, line 4) is better than the performance of the CR model by itself (line 1) or CR + embeddings (line 3). However, here this difference is not significant. This suggests that representing the justification as a simple bag of words with latent feature representations is not as effective as representing them with features derived from the structure of the text aggregation graph, even with the non-linear capabilities of the neural network.

Additionally, we find that the neural network variants perform worse overall than the latent ranking perceptron models and the voting ensemble (cf. Table 6). This could be due to the fact that the TAG and CR features we are supplying to the neural network are already abstracted many levels from raw text inputs, and so the NN approach benefits less from the non-linearity of the NN. Another likely reason for the NN performing worse than the perceptron is that neural architectures need larger quantities of training data in order to properly generalize. Notably, in the recent Allen Institute for Artificial Intelligence Kaggle challenge,[16] a large contest with 170 participating teams that also required answering multiple choice elementary and middle school science questions, none of the top-performing participants obtained better results with neural networks. The participants' analyses suggested that this is largely because of a lack of training

[14] We experimented with some deeper networks and several hidden layer sizes (both larger and smaller).

[15] We also experimented with using embeddings trained over English Gigaword (Graff et al. 2003) because that resource is much larger and would potentially yield more robust word representations. However, we found that the performance was consistently worse, which we suspect is due to the difference in the domains.

[16] `https://www.kaggle.com/c/the-allen-ai-science-challenge`.

Table 9
Performance of the Latent Ranking Neural Network models. Models with *CR* include the candidate retrieval scores as input, models with *TAG* use the features from the best performing TAG model ($1G_{CT}$+$2G_{CT}$), and models with *embeddings* include an average embedding for each of the questions, the answers, and the text from which the justification graphlet was derived. Significance tests were performed using bootstrap resampling with 10,000 iterations, but none of the differences between the neural network models and the CR baseline were significant.

#	Neural Network Models	P1	MRR
1	CR	40.74%	62.56%
2	CR + TAG	40.52%	62.48%
3	CR + embeddings	38.74%	61.61%
4	CR + TAG + embeddings	41.82%	63.11%

data. Finally, the space of possible neural network architectures is large, and the network we report here is fairly simple. Although it could be the case that given a more complex architecture (i.e., with a recurrent or convolutional network, optimizers, and learned rather than pre-trained embeddings) we could obtain even higher numbers than those reported here, with complex architectures, issues resulting from the limited training data would likely be worsened. We leave that exploration to future work.

Incorporating Embeddings into the Perceptron. Though the embedding-based neural architecture (as implemented) was not successful, following a reviewer suggestion we also experimented with adding the precomputed text embeddings as additional features for the latent ranking perceptron. In this way, the feature vector for each TAG, $\Phi(x)$, contains the original features as well as an averaged embedding for each of the question, answer, and justification texts. This provides an additional 600 features, as we are using 200-dimensional embedding vectors. We included these extra features in our best performing single TAG model, the connection-type aware 1G+2G model shown in Table 6, line 5. The resulting model performed worse (39.36% P@1 compared with 42.88% P@1 without the embeddings). This shows that, although distributed representations of words have many benefits (including robustness to lexical variation), their incorporation into a learning model is not necessarily trivial, and we leave that study to future work.

9. Discussion

To further characterize the performance of our QA approach, we address the following questions:

How does performance compare with methods using manually constructed knowledge bases? The TAG system automatically aggregates sentences from six free-text corpora first by building graphlets from those sentences using syntactic dependencies, then connecting those graphlets together into multisentence text aggregation graphs that are then used to both answer questions and provide a compelling human-readable justification for the selected answers. Recently, Khashabi et al. (2016) demonstrated that graphs for elementary science QA can also be constructed using a semistructured knowledge base of tables. In this formalism, dozens of themed tables are manually or semi-automatically constructed, each around a particular theme. A table's theme is encoded in its columns—that is, a table for the color of objects contains two rows, one for the object of interest (e.g., "leaf"), and another for its color (e.g., "green"), while separate instances (e.g., leaf—green, trunk—brown) are encoded as different rows. Each table has between two and five columns.

The TableILP algorithm answers questions by chaining facts between different table rows, starting from a row that contains question terms, then traversing to a new table row that contains some lexical overlap with the previous row(s), until answer terms are found. The TAG and TableILP systems are conceptually similar, with the central differences being: (1) The TableILP table row is roughly equivalent to a TAG graphlet with flat structure, and limited to two to five information nuggets containing only single terms; (2) TAG graphlets are read automatically from free text corpora, where TableILP tables are largely manually constructed, with methods to automate this construction being actively developed; and (3) the traversal algorithms are different, with TableILP graph building being modeled as an integer linear programming (ILP) problem that finds paths that maximize QA performance.

The TableILP system reported by Khashabi et al. (2016) contains 69 tables containing a total of 7,600 rows, with 64 of these tables (approximately 5,000 rows) designed around material in the study guides and a development corpus, and the remaining 2,600 rows distributed among four automatically constructed tables. On a corpus of 200 questions drawn from the 1,000 questions used here, TableILP achieved a score of 45.6% P@1,[17] compared with the 44.6% P@1 from the best-performing TAG model in Table 6. The performance difference between these systems is likely not statistically significant.[18]

We view these systems as complementary, converging, and with each capable of exploring different aspects of graph-based inference for science QA. Although the TAG focuses on automatically building graphs from free text, this is currently a challenging and noisy process, and as we have shown in Table 5 and Fried et al. (2015), highly susceptible to inference drift as the amount of information required to be aggregated becomes large. On the other hand, building graphs from manually constructed knowledge bases allows us to investigate the graph-building process in isolation, reducing inference drift due to noise, and further moving this area forward.

How does performance vary by grade level? The question corpus contains third, fourth, and fifth grade questions. A human with a level of knowledge equivalent to a fourth grade science student might be expected to show better performance for the simpler third grade questions, and decreasing performance as question difficulty increases from fourth to fifth grades. Table 10 shows P@1 performance by grade level for both the CR and best performing TAG model ($1G_{CT}$ + $2G_{CT}$). The TAG model shows decreasing performance as question difficulty increases, dropping from 49% for third grade questions to 42% for fourth and fifth grade questions. The CR baseline, however, displays a qualitatively different pattern, with a peak performance of 51% for fourth grade questions, and *near chance* performance for third grade questions. We believe that observing such a pattern in performance may suggest that the TAG model is a closer approximation of human inference than the baseline based solely on information retrieval. Here, the relatively small number of third and fourth grade questions prevents us from drawing any conclusions, but suggests that crafting question sets to allow

[17] We wish to thank Khashabi et al. (2016) for providing us with this performance figure.

[18] We did not have access to system output. However, in our experiments on this data set, we observed that only differences in P@1 scores of 3% or higher (absolute) tend to be statistically significant at $p < 0.05$.

Table 10
Precision@1 by grade level.

Grade Level	Questions	CR	TAG
Grade 3	60	28.3%	49.2%
Grade 4	69	50.7%	41.3%
Grade 5	871	40.2%	42.6%

evaluating the distribution of performance by grade level may provide a further measure of comparison between human and machine performance.

Which knowledge resources are generating the most useful answer justifications? Shown in Table 11, the Barron's Study Guide contributes more of the *good* justification sentences than any other source, followed by the science dictionary, then the other resources. Interestingly, the Simple Wicktionary contributes the fewest sentences to the *good* justifications for the CR system (7.5%), but for the TAG system it is the third largest contributor (14.8%). That is, whereas the CR system is typically unable to find a *good* justification from the Wiktionary, likely owing to its general nature, the TAG system is able to successfully aggregate these sentences with sentences from other domain-specific sources to build complete justifications.

The vast majority of the *good* justifications generated by the TAG system are aggregates from non-adjacent text: 67% of the justifications aggregate sentences from *different* corpora, 30% aggregate non-adjacent sentences within a *single* corpus, and only 3% of *good* justifications contain sentences that were adjacent in their original corpus. This is clear evidence that information aggregation (or fusion) is fundamental for answer justification.

How orthogonal is the performance of the TAG model when compared with CR? Both the TAG and CR models use the same knowledge resources, which on the surface suggests the models may be similar, answering many of the same questions correctly. The voting models in Table 6 appear to support this, where combining the TAG and CR models increases performance by just under 2% P@1 over the best-performing TAG model. To investigate this, we conducted an orthogonality analysis to determine the number of questions both models answer correctly, and the number of questions each model uniquely answers correctly.

Table 11
Most useful knowledge resources for justifications classified as "good."

Resource	Sentences	CR	TAG
Barrons SG	1,200	39.3%	43.0%
Flashcards	283	16.2%	8.2%
Teacher's Guide	302	7.1%	7.0%
Virginia SG	1,314	9.1%	9.2%
Science Dictionary	733	20.8%	17.8%
Simple Wiktionary	17,473	7.5%	14.8%

Comparing the TAG ($1G_{CT}$ + $2G_{CT}$) and CR models, nearly half of questions are answered correctly by one model and incorrectly by the other. When combined into a two-way voting model, this causes a large number of ties—which, resolved at chance, would perform at 42%, with ceiling performance (i.e., all ties resolved correctly) at 60%. This indicates that although the TAG and CR models share about half of their performance, each model is sensitive to different kinds of questions, suggesting that further combination strategies between TAG and CR are worth exploring.

10. Error Analysis

At a high level, our sentence aggregation model can be thought of as a system for building and evaluating answer justifications. When questions are answered incorrectly, it is unclear whether the system was unable to *build* a good justification for the correct answer, or unable to *detect* a good justification that was successfully built. To examine this, in addition to evaluating justification quality for correct answers, we also evaluated the top six justifications for 100 questions that the TAG system answered incorrectly. We investigated the type of TAG patterns in the correct justifications (from Section 6.2), as well as the kinds of inference required for questions that were answered incorrectly. In addition, we conducted a large open-ended error analysis and identified several broad classes of issues that contribute to poor system performance.[19]

What patterns of lexical overlap do we observe in justification sentences? We begin with an encouraging result. For questions that were answered incorrectly, nearly half (45.1%) contain at least one *good* justification within the top 6 for the correct answer. This suggests that the current model is much better at generating and broadly ranking justifications than it is at the final step—pushing the correct justification from the top 6 to the top position. This result suggests that the TAG model may benefit from an improved ranking model. For example, whereas we currently use lexicalization to model TAG structures and quantify the overlap between candidate TAGs and the question, none of the features used by the ranking perceptron are explicitly lexicalized. We explored some neural network variants of our models in Section 8.5, which were better able to incorporate lexicalization, but did not see a performance gain. These models can be extended in future work, however, to see if it is ultimately possible to make use of this lexicalization.

Further, for questions answered correctly, the *good* justifications in the top 6 tend to connect on several of the three lexical overlap categories (i.e., question focus words, answer focus words, other shared non-focus words). Table 12 shows that for questions answered correctly, in 69% of cases good justifications are connected on two or three lexical overlap categories. Conversely, for questions that are answered incorrectly, the *good* justifications are far more likely to contain sentences that only connect on a single lexical overlap category (57.6% vs. 31.0% for questions answered correctly). This suggests that although less lexically connected answer justifications are not necessarily less numerous or of lower quality, they are much more challenging to detect given our current connectivity-centered features. That is, the current features are very good at

[19] Our current focus word extractor is not yet well-suited to complex multisentence questions. Although the question corpus as a whole consists of 62% single-sentence questions, 25% two-sentence questions, and 14% questions that are three sentences or longer, we elected to focus on shorter single-sentence questions for this error analysis whenever possible. As a result, 96% of the questions analyzed were single-sentence and 4% were two-sentence.

Table 12
Proportion of *good* justifications with a given number of connecting word categories (Q, A, X) for both correct and incorrect answers (Top 6).

Connecting Categories	Correct	Incorrect
1	31.0%	57.6%
2	56.3%	39.4%
3	12.7%	3.0%

detecting well-connected answer justifications, which correlate with good answers, but the features are not able to directly detect good answer justifications, which is a more challenging theoretical problem.

Is it harder to detect justifications for certain kinds of inference? We predict that questions requiring more challenging kinds of inference as identified by Clark et al. (2013), like model-based or general inference questions, are likely more difficult for the TAG system to answer than simpler retrieval-based questions. Following the criteria described by Clark et al., we classified the 100 questions in the error analysis based on the inference type required to answer them correctly. This analysis, shown in Table 13, suggests that it is much more challenging to detect *good* justifications for questions that contain challenging inference. Where 58% of retrieval-based questions contained a *good* justification within the top six, this decreased to 29% for questions requiring general inference, and to 13% for questions requiring complex model-based reasoning. Taken in conjunction with our analysis of lexical overlap categories, this suggests that good justifications for complex inference may tend to be less lexically connected, and thus more challenging to detect with our current framework. This suggests important avenues for future work.

10.1 Broader Error Classes

To expand on the above two issues, we conducted a broader, open-ended error analysis, and identified six main error classes made by the system: focus word failures, noise in the knowledge base, complex inference, semantic cluster matching, coverage of the

Table 13
A summary of the inference type necessary for incorrectly answered questions. The summary is broken down into three categories: incorrectly answered questions with a good justification in the top six, incorrectly answered questions without a good justification in the top six, as well as the overall proportions across these two conditions.

Inference Type	Good Justification	No Good Justification	Proportion of Questions
Retrieval	58%	23%	39%
General Inference	29%	43%	37%
Model-Based	13%	34%	25%

Table 14
A summary of the classes of the errors made by the system. On any given question, more than one error may have been made. The summary is broken down into three categories: incorrectly answered questions with a good justification in the top six, incorrectly answered questions without a good justification in the top six, as well as the overall proportions across these two conditions.

Error Class	Good Justification	No Good Justification	Overall
FOCUS WORDS			
Focus Word — Question	49%	46%	48%
Focus Word — Answer	20%	32%	27%
Focus Word — Answer Lists	7%	5%	6%
Focus Word — Compounds/Collocations	9%	16%	13%
NOISE IN THE KNOWLEDGE BASE			
Excessively Long Sentence (Chosen Ans)	33%	9%	20%
Excessively Long Sentence (Correct Ans)	13%	4%	8%
COMPLEX INFERENCE			
More Sentences Required	4%	16%	11%
Causal or Process Reasoning	27%	30%	29%
Quantifiers	4%	23%	15%
Negation	2%	9%	6%
MISCELLANEOUS			
Semantic Cluster Matching	16%	5%	10%
Coverage of Knowledge Bases	4%	36%	22%
Other	29%	9%	18%

knowledge bases, and other errors. The distribution of these errors is shown in Table 14 and detailed here.

10.1.1 Focus Word Errors. Extracting focus words from questions and answers can be difficult, as often these questions are grounded in complex examples. Moreover, although examples can often be filtered out, sometimes they are necessary to the question. Within the larger category of focus word errors, we subdivided these errors into more specific categories:

Question and Answer Focus Words: We considered a question or answer to have focus word issues if either the words selected by the focus word extractor were not critical to the inference, or if the weights did not reflect the relative importance of each word to the inference. An example of this can be seen in Table 15.

Answer Choice Lists: Candidate answers may not be single choices, but occasionally take the form of lists, as in the question *which of the following organisms are decomposers?* and its correct answer *worms, mushrooms, and insects*. In these cases each list item for a given answer candidate must be independently verified for correctness, as incorrect lure answers often contain one or more correct list items. The current system does not handle this type of complex problem solving method.

Table 15
Example of failure to extract appropriate focus words from the question.

Focus Words — Question	
Question	What type of simple machine is Bruce using as he opens a twist top bottle of soda? (GR:5)
Focus Word(s)	(NN_bruce, 0.22) (VB_open, 0.22) (NN_twist, 0.22) (JJ_top, 0.22) (NN_bottle, 0.05) (NN_soda, 0.03) (NN_machine, 0.02)
Issue	The concept the question is testing is embedded in a complex example that the focus word extractor is only partially able to suppress. Here, the most critical word for the inference is *twist*, but some example words (*bruce, open*, and *top*) are not suppressed, and receive the same weight as *twist*.

Compounds/Collocations: The current focus word extractor operates at the level of the word, and does not join noun–noun compounds or collocations, such as *simple machine* or *water cycle*. This causes compounds to be split, with different focus weights assigned to each word. This also adds noise to the inference process, as (for example) a justification sentence that contains *simple* but not *machine* is less likely to be on context.

10.1.2 Noise in the Knowledge Base. We included nearly all sentences contained in each of the five science-domain corpora. Although most of this is high-quality text, occasionally extremely long sentences are present (e.g., prefacing a chapter with a list of keywords). These sentences can cause broad topic-level lexical connections between sentences that are only marginally related to each other, and are a source of noise in the justification building process.

10.1.3 Complex Inference. Some questions require complex inference to be answered correctly. This may take the form of requiring longer sequences of graphlets to construct a complete answer justification, requiring an understanding of quantifiers or negation, or a complex inference process.

More Sentences Required: In our knowledge base, a sentence tends to capture a single step in some process. Although two sentences are often sufficient to construct a good justification for retrieval-based questions, for general inference and model-based reasoning questions, an inference may require more than two steps. Further, for questions that are grounded in a concrete example, additional graphlets may be needed to elevate the example words to the more general level of the concepts in our knowledge base – for example, elevating *polar bear* to *predator* or *animal*, in Table 16.

Causal or Process Reasoning: Questions that require causal or process reasoning often require a small structured and ordered representation of that process. For example, in Table 17, a knowledge of the sequential nature of the water cycle—that *evaporation* leads to *condensation*, and that *condensation* leads to *precipitation*—is necessary to answer the question.

Quantifiers: Some questions require an understanding of quantifiers or scope to arrive at a correct answer, whether these quantifiers are included in the question, answer, or knowledge base. As illustrated in Table 18, our current system does not implement a knowledge of quantifiers, nor their relationship to each other (*some* is less than *most*, *smaller* is less than *big*, etc.).

Table 16
Example of a question that needs more than two sentences to answer.

Complex Inference — More sentences required to construct a complete answer	
Question	Which of the following would result in a decrease in the polar bear population in the arctic? (GR:5)
Focus Word(s)	(JJ_polar, 0.29) (NN_population, 0.29) (NN_arctic, 0.29) (VB_result, 0.07) (NN_decrease, 0.04) (NN_bear, 0.02)
Issue	Requires additional graphlets, including that *bears* eat *fish*, *eating* an animal makes you a *predator* and it your *prey*, and a decrease in *prey* population also causes a decrease in *predator* population. Here, with a limited justification length of two sentences, the system is not able to construct a justification that includes all the crucial focus words, and all answer candidates are left with the same general justification fragment that includes the highest-weighted focus words.
Correct Answer	a decrease in the fish population
Focus Word(s)	(NN_population, 0.80) (NN_decrease, 0.13) (NN_fish, 0.07)
Justification	A polar bear is a big, white bear that lives in the arctic. (Wiktionary) The population of a place is the people or animals that live there. (Wiktionary)
Chosen Answer	a decrease in the human population
Focus Word(s)	(NN_population, 0.92) (NN_decrease, 0.08)
Justification	A polar bear is a big, white bear that lives in the arctic. (Wiktionary) The population of a place is the people or animals that live there. (Wiktionary)

Table 17
Example of a question that requires reasoning over a causal structure or process.

Complex Inference — Causal or Process Reasoning	
Question	In the water cycle, which process occurs **after** condensation? (GR:5)
Focus Word(s)	(NN_condensation, 0.75) (VB_occur, 0.13) (NN_water, 0.06) (NN_cycle, 0.06)
Issue	Requires knowing that the water cycle is a stage-like process that proceeds as follows: *evaporation*, *condensation*, *precipitation*. The focus word extractor does not currently detect causal or process markers, and as such the focus words for the question do not include *after*.
Correct Answer	Precipitation
Justification	Condensation, in the water cycle, **this step leads to** precipitation. (Science Dictionary) When the humidity reaches 100 percent it is very likely that some form of precipitation will occur, depending on the temperature. (Barrons SG)
Chosen Answer	Evaporation
Justification	When water vapor changes to liquid water it is called condensation. (Barrons SG) Evaporation is defined as the change of water from its liquid phase to its gaseous phase. (Barrons SG)

Negation: Our current framework does not implement a knowledge of negation. Within the scope of elementary science exams, where questions tend to ask for positive rather than negative evidence, this is often not an issue, with the overall prevalence of questions requiring negation at 6%. However, our knowledge bases do include many negated sentences or phrases that provide a contrast between two categories of items through negation (e.g., *melting is a physical change, not a chemical change*). As can be seen in Table 19, these sentences can be misused by our system.

Table 18
Example of a question that requires an understanding of the quantifiers in both the question and the answers.

Complex Inference — Quantifiers	
Question	Which of the following is a factor that will cause different species to compete **less?** (GR:5)
Focus Word(s)	(NN_species, 0.50) (JJ_less, 0.17) (VB_compete, 0.13) (VB_cause, 0.10) (JJ_different, 0.07) (NN_factor, 0.03)
Issue	Requires connecting the idea that a *large* supply causes *less* competition. The focus word extractor does not currently detect quantifiers, and as such the focus words for the correct answer do not include *large*.
Correct Answer	A **large** supply of resources
Focus Word(s)	(NN_supply, 0.92) (NN_resource, 0.08)
Chosen Answer	**Lack** of space
Focus Word(s)	(NN_space, 0.92) (NN_lack, 0.08)

Table 19
Example of a question that requires an understanding of negation.

Complex Inference — Negation	
Question	Which of the following is an example of a chemical change? (GR:5)
Issue	Requires detecting negation in the graphlets. The chosen answer justification contains negative evidence against itself.
Correct Answer	Milk souring
Justification	Examples of chemical properties include the souring of milk and the ability to burn in the presence of oxygen. (Barrons SG) Evidence of a chemical change could be change in temperature, light, heat, or sound given off, or the formation of gasses. (Science Dictionary)
Chosen Answer	Ice cream melting
Justification	Melting of ice is a physical change, **not a chemical change**. (Barrons SG) Melting is a process of an object changing from the solid state to the liquid state without a change in temperature. (Wiktionary)

10.1.4 Miscellaneous

Semantic Cluster Matching: Although the current system reduces noise by making lexical connections only between words in sentences that have the same lemma and part of speech, this strict criterion prevents some *good* justifications from being built, and others from being recognized as *good* justifications. Ideally, semantically related terms such as *condensation* and *condensing* (as in Table 20), or *heredity* and *inherit*, could be clustered together to facilitate building more robust answer justifications independent of lexical choice.

Coverage of Knowledge Bases: Inevitably our knowledge base suffers from a degree of sparsity, where some topics or specific concepts included in questions are not discussed in any of our corpora. This rarely happens with grade-appropriate science topics (but does occur for topics such as *competition*). Coverage issues are much more frequent with the concrete examples that ground the questions, though we mitigate this by including the simple Wiktionary as a grade-appropriate general knowledge resource.

Table 20
Example of a failure to recognize relatedness or equivalence of words.

Semantic Cluster Matching	
Question	Which is an example of water condensing? (GR:4)
Focus Word(s)	(NN_condensing, 0.92) (NN_water, 0.08)
Issue	Requires connecting *condensing* in the question with *condensation* in the correct answer justification, based on their high degree of semantic relatedness.
Correct Answer	Dew forming on plants during a cold night
Focus Word(s)	(JJ_cold, 0.68) (NN_dew, 0.16) (NN_night, 0.11) (NN_plant, 0.05)
Justification	Clouds, dew, water droplets on the outside of a glass on a hot day, are all caused by **condensation**. (Virginia Flash Cards) Think back to a hot summer day, when you poured yourself a glass of cold water and took it outside. (Barrons SG)
Chosen Answer	A puddle disappearing on a hot summer afternoon
Focus Word(s)	(NN_summer, 0.41) (NN_afternoon, 0.41) (JJ_hot, 0.09) (VB_disappear, 0.06) (NN_puddle, 0.03)
Justification	After a few hours or days those puddles disappear. (Barrons SG) Think back to a hot summer day, when you poured yourself a glass of cold water and took it outside. (Barrons SG)

Other: Although we were able to determine the source for the majority of errors, for 18% of incorrect questions we were unable to readily identify the issue. Because of the difficulty of this QA task, we hypothesize that many of these cases may result from the limitations of learning complex latent variables with our learning framework and limited training data.

10.2 Summary of Errors

To summarize, this error analysis suggests that a majority of errors are caused by surface issues, and not fundamental limitations of this approach to question answering for science exams. Were we to solve these issues, including focus-word extraction errors, noise in the knowledge base, knowledge coverage, and semantic cluster matching, our analysis suggests that 50.5% of questions answered incorrectly could then be answered correctly. Using this figure, the ceiling performance for the current model is estimated to be 71.4%.

The remainder of errors center on difficulties with complex questions, including questions requiring a knowledge of causality or processes to answer, longer inference chains, or a knowledge of quantifiers or negation. We hypothesize that many of these questions can be addressed by extending the current model towards including more structure, and making better use of the existing structure within graphlets. For example, much of the structure within a sentence that explicitly conveys causal or process information (e.g., *from a solid to a liquid*) is not explicitly labelled within a graphlet, or used in a targeted way towards addressing questions that require these kinds of complex inference. By extending our general methods of justification construction and evaluation to address the specific information needs of these complex questions, we believe that the ceiling performance can be increased considerably, limited only by questions that require complex domain-specific mechanisms, such as spatial reasoning, to answer. The tradeoff between generality versus domain specificity appears intimately coupled with a model's performance, and other QA systems such as IBM Watson approach QA by assembling a large ensemble of domain-specific models tailored to a given problem representation. To surpass the ceiling we observe in our error analysis, one would likely also have to adopt this approach, and implement dedicated domain-specific methods for the difficult problems left unsolved by our approach.

11. Conclusions

We have proposed an approach for QA where producing human-readable justifications for answers, and evaluating answer justification quality, is the critical component. Our interdisciplinary approach to building and evaluating answer justifications includes cognitively inspired aspects, such as making use of psycholinguistic concreteness norms for focus word extraction, and making use of age-appropriate knowledge bases, which together help move our approach towards approximating the qualities of human inference on the task of question answering for science exams. Intuitively, our structured representations for answer justifications can be interpreted as a robust approximation of more formal representations, such as logic forms (Moldovan and Rus 2001). However, our approach does not evaluate the quality of connections in these structures by their ability to complete a logic proof, but through a reranking model that measures their correlations with good answers.

In our quest for explainability, we have designed a system that generates answer justifications by chaining sentences together. Our experiments showed that this approach improves explainability, and, at the same time, answers questions out of reach of information retrieval systems, or systems that process contiguous text. We evaluated our approach on 1,000 multiple-choice questions from elementary school science exams, and experimentally demonstrated that our method outperforms several strong baselines at both selecting correct answers, and producing compelling human-readable justifications for those answers. We further validated our three critical contributions: (a) modeling the high-level task of determining justification quality by using a latent variable model is important for identifying both correct answers and good justifications, (b) identifying focus words using psycholinguistic concreteness norms similarly benefits QA for elementary science exams, and (c) modeling the syntactic and lexical structure of answer justifications allows good justifications to be assembled and detected.

We performed a detailed error analysis that suggests several important directions for future work. First, though the majority of errors can be addressed within the proposed formalism and by improving focus word extraction, 47.5% of incorrectly answered questions would also benefit from more complex inference mechanisms, ranging from causal and process reasoning, to modeling quantifiers and negation. This suggests that our robust approach for answer justification may complement deep reasoning methods for QA in the scientific domain (Baral, Liang, and Nguyen 2011). Second, our text aggregation graphs currently capture intersentence connections solely through lexical overlap. We hypothesize that extending these structures to capture lexical-semantic overlap driven by word embeddings (Mikolov et al. 2013), which have been demonstrated to be beneficial for QA (Yih et al. 2013; Jansen, Surdeanu, and Clark 2014; Fried et al. 2015), would also be beneficial here, and increase robustness on small knowledge bases, where exact lexical matching is often not possible. Finally, whereas our answer justifications are currently short, future justifications might be quite long, and aggregate sentences from knowledge bases of different domains and genres. In these situations, combining our procedure for constructing justifications with methods that improve text coherence (Barzilay and Lapata 2008) would likely improve the overall user experience for reading and making use of answer justifications from automated QA systems.

To increase reproducibility, all the code behind this effort is released as open-source software,[20] which allows other researchers to use our entire science QA system as is, or to explore adapting the various components to other tasks.

Acknowledgments

We thank the Allen Institute for Artificial Intelligence for funding this work, and the helpful comments from three anonymous reviewers. Dr. Mihai Surdeanu discloses a financial interest in Lum.ai. This interest has been disclosed to the University of Arizona Institutional Review Committee and is being managed in accordance with its conflict of interest policies.

References

Balduccini, Marcello, Chitta Baral, and Yuliya Lierler. 2008. Knowledge representation and question answering. *Foundations of Artificial Intelligence*, 3:779–819.

Banarescu, Laura, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Baral, Chitta and Shanshan Liang. 2012. From knowledge represented in frame-based languages to declarative representation and reasoning via asp. In KR.

Baral, Chitta, Shanshan Liang, and Vo Nguyen. 2011. Towards deep reasoning with respect to natural language text in scientific domains. In *DeepKR Workshop*. Citeseer.

Baral, Chitta, Nguyen Ha Vo, and Shanshan Liang. 2012. Answering why and how questions with respect to a frame-based knowledge base: a preliminary report. In *ICLP (Technical Communications)*, pages 26–36.

Barzilay, Regina and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Barzilay, Regina and Kathleen R McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Barzilay, Regina, Kathleen R. McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 550–557.

Berger, Adam, Rich Caruana, David Cohn, Dayne Freytag, and Vibhu Mittal. 2000. Bridging the lexical chasm: Statistical approaches to answer finding. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research & Development on Information Retrieval*, Athens, Greece.

Björkelund, Anders and Jonas Kuhn. 2014. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *Proceedings of the Association for Computational Linguistics.*

Blair-Goldensohn, Sasha, Kathleen McKeown and Andrew Hazen Schlaikjer. 2003. A hybrid approach for answering definitional questions. Technical Report CUCS-006-03, Columbia University.

Bordes, Antoine, Sumit Chopra and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP) 2014*.

Bordes, Antoine, Nicolas Usunier, Sumit Chopra and Jason Weston. 2015. Large-scale simple question answering with memory networks. arXiv preprint arXiv:1506.02075.

Brysbaert, Marc, AmyBeth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known English word lemmas. *Behavior Research Methods*, 46(3):904–911.

Chen, Danqi, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the CNN / Daily Mail reading comprehension task. In *Proceedings of the Association for Computational Linguistics (ACL).*

Chen, Danqi and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Empirical MNLP*, pages 740–750.

[20] `https://github.com/clulab/releases/tree/master/cl2017-qa`.

Clark, Peter. 2015, Elementary school science and math tests as a driver for AI: take the aristo challenge! In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 4019–4021. Austin, TX.

Clark, Peter, Philip Harrison, and Niranjan Balasubramanian. 2013. A study of the knowledge base requirements for passing an elementary science test. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*, AKBC'13, pages 37–42.

Collins, Michael. 2002, Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, EMNLP '02, pages 1–8, Stroudsburg, PA.

Collobert, Ronan, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 2493–2537.

Daumé III, Hal. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 256–263, Prague.

De Marneffe, Marie-Catherine and Christopher D. Manning. 2008. Stanford typed dependencies manual. Technical report, Stanford University.

Dong, Li, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the Association for Computational Linguistics*, pages 260–269.

Echihabi, Abdessamad and Daniel Marcu. 2003. A noisy-channel approach to question answering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics-Volume 1*, pages 16–23.

Fernandes, Eraldo Rezende, Cícero Nogueira Dos Santos, and Ruy Luiz Milidiú. 2012. Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)—Shared Task*, pages 41–48.

Ferrucci, David. A. 2012. Introduction to "this is Watson." *IBM Journal of Research and Development*, 56(3.4).

Finkel, Jenny Rose and Christopher D. Manning. 2010. Hierarchical joint learning: Improving joint parsing and named entity recognition with non-jointly labeled data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 720–728.

Fried, Daniel, Peter Jansen, Gustave Hahn-Powell, Mihai Surdeanu, and Peter Clark. 2015. Higher-order lexical semantic models for non-factoid answer reranking. *Transactions of the Association for Computational Linguistics*, 3:197–210.

Gondek, D. C., Adam Lally, Aditya Kalyanpur, J. William Murdock, Pablo Ariel Duboué, Lei Zhang, Yue Pan, ZM Qui, and Chris Welty, 2012. A framework for merging and ranking of answers in deepqa. *IBM Journal of Research and Development*, 56(3.4).

Graff, David, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword, ldc2003t05. Linguistic Data Consortium, Philadelphia.

Harabagiu, Sanda, Dan Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunescu, Roxana Girju, Vasile Rus, and Paul Moraraescu. 2000. Falcon: Boosting knowledge for answer engines. In *Proceedings of the Text REtrieval Conference (TREC)*, Gaithersburg, MD.

He, Hua and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of NAACL-HLT*, pages 937–948.

Hermann, Karl Moritz, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

Hoffmann, Raphael, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550.

Iyyer, Mohit, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Empirical Methods in Natural Language Processing*.

Iyyer, Mohit, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Association for Computational Linguistics*.

Jansen, Peter, Mihai Surdeanu, and Peter Clark. 2014. Discourse complements lexical semantics for non-factoid answer reranking. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Khashabi, Daniel, Tushar Khot, Ashish Sabharwal, Peter Clark, Oren Etzioni and Dan Roth. 2016. Question answering via integer programming over semi-structured knowledge. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, 1145–1152.

Lee, Heeyoung, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4).

Lewis, Mike and Mark Steedman. 2013. Combining distributional and logical semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192.

Liang, Percy, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006, An end-to-end discriminative approach to machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 761–768.

Liang, Percy, Michael I. Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

MacCartney, Bill. 2009. *Natural Language Inference*. PhD thesis.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

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

McSherry, Frank and Marc Najork. 2008. Computing information retrieval performance measures efficiently in the presence of tied scores. In *30th European Conference on IR Research (ECIR)*.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Mikolov, Tomas, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*.

Moldovan, Dan, Christine Clark, Sanda Harabagiu, and Daniel Hodges. 2007. Cogex: A semantically and contextually enriched logic prover for question answering. *Journal of Applied Logic*, 5(1):49–69.

Moldovan, Dan, Christine Clark, Sanda Harabagiu and Steve Maiorano. 2003a. Cogex: A logic prover for question answering, In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 87–93.

Moldovan, Dan, Marius Paşca, Sanda Harabagiu, and Mihai Surdeanu. 2003b. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems*, 21(2):133–154.

Moldovan, Dan I. and Vasile Rus. 2001. Logic form transformation of Wordnet and its applicability to question answering. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 402–409.

Moschitti, Alessandro. 2004. A study on convolution kernels for shallow semantic parsing. In *Proceedings of the 42th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Moschitti, Alessandro and Silvia Quarteroni. 2011. Linguistic kernels for answer re-ranking in question answering systems. *Information and Processing Management*.

Moschitti, Alessandro, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question/answer classification. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 776–783, Prague.

Murdock, J. William, James Fan, Adam Lally, Hideki Shima, and B. K. Boguraev. 2012. Textual evidence gathering and analysis. *IBM Journal of Research and Development*, 56(3.4).

Park, Jae Hyun and W. Bruce Croft. 2015. Using key concepts in a translation model for retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 927–930, New York, NY.

Piaget, Jean. 1954. *The Construction of Reality in the Child*. Basic Books.

Pradhan, Sameer S., Valerie Krugler, Steven Bethard, Wayne Ward, Daniel Jurafsky, James H. Martin, Sasha Blair-Goldensohn, Andrew Hazen Schlaikjer, Elena Filatova, Pablo Ariel Duboué et al. 2002. Building a foundation system for producing short answers to factual questions. In *TREC*.

Riezler, Stefan, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 464–471.

Severyn, Aliaksei and Alessandro Moschitti. 2012. Structural relationships for large-scale learning of answer re-ranking. In *SIGIR*.

Severyn, Aliaksei and Alessandro Moschitti. 2013. Automatic feature engineering for answer selection and extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Severyn, Aliaksei, Massimo Nicosia, and Alessandro Moschitti. 2013. Learning adaptable patterns for passage reranking. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning (CoNLL)*.

Sharma, Arpit, Nguyen H. Vo, Somak Aditya, and Chitta Baral. 2015. Towards addressing the winograd schema challenge-building and using a semantic parser and a knowledge hunting module. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*.

Sharp, Rebecca, Peter Jansen, Mihai Surdeanu, and Peter Clark. 2015. Spinning straw into gold: Using free text to train monolingual alignment models for non-factoid question answering. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 231–237, Denver, CO.

Shen, Libin and Aravind K. Joshi. 2005. Ranking and reranking with perceptron. *Machine Learning. Special Issue on Learning in Speech and Language Technologies*, 60(1):73–96.

Soricut, Radu and Eric Brill. 2006. Automatic question answering using the Web: Beyond the factoid. *Journal of Information Retrieval - Special Issue on Web Information Retrieval*, 9(2):191–206.

Suggu, Sai Praneeth, Kushwanth N. Goutham, Manoj K. Chinnakotla, and Manish Shrivastava. 2016. Deep feature fusion network for answer quality prediction in community question answering. arXiv preprint arXiv:1606.07103.

Sultan, Md. Arafat, Steven Bethard, and Tamara Sumner. 2014. Back to basics for monolingual alignment: Exploiting word similarity and contextual evidence. *Transactions of the Association for Computational Linguistics*, 2:219–230.

Sun, Xu, Takuya Matsuzaki, Daisuke Okanohara, and Jun'ichi Tsujii. 2009. Latent variable perceptron algorithm for structured classification. In *IJCAI*, volume 9, pages 1236–1242.

Surdeanu, Mihai, Massimiliano Ciaramita, and Hugo Zaragoza. 2011. Learning to rank answers to non-factoid questions from Web collections. *Computational Linguistics*, 37(2):351–383.

Tari, Luis and Chitta Baral. 2006. Using AnsProlog with link grammar and Wordnet for QA with deep reasoning. In *Proceedings of the 9th International Conference on Information Technology*, pages 125–128.

Tymoshenko, Kateryna and Alessandro Moschitti. 2015. Assessing the impact of syntactic and semantic structures for answer passages reranking. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM)*.

Voorhees, Ellen M. 2003. Evaluating answers to definition questions. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Companion Volume of the Proceedings of HLT-NAACL 2003–short Papers - Volume 2*, NAACL-Short '03, pages 109–111.

Wang, Di and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. *ACL, July*.

Yao, Xuchen, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. Semi-Markov phrase-based monolingual alignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yih, Wen-tau, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Yih, Wen-tau, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Zettlemoyer, Luke S. and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 678–687.