




COMPUTATIONAL NATURAL LANGUAGE INFERENCE: ROBUST AND INTERPRETABLE QUESTION ANSWERING

by

Rebecca Reynolds Sharp

   Creative Commons Attribution-No Derivative Works 3.0 License

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF LINGUISTICS

In Partial Fulfillment of the Requirements

For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2017

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Dissertation Committee, we certify that we have read the dissertation prepared by Rebecca Reynolds Sharp, titled Computational Natural Language Inference: Robust and Interpretable Question Answering and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

Michael Hammond

Date: 14 August 2017

Mihai Surdeanu

Date: 14 August 2017

Peter Jansen

Date: 14 August 2017

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

Dissertation Director: Michael Hammond

Date: 14 August 2017

Dissertation Director: Mihai Surdeanu

Date: 14 August 2017

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at the University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of the source is made. This work is licensed under the Creative Commons Attribution-No Derivative Works 3.0 United States License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nd/3.0/us/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

SIGNED: Rebecca Reynolds Sharp

ACKNOWLEDGEMENTS

I would like to thank the Allen Institute for Artificial Intelligence for funding a large portion of the work contained here. Additionally, this work was partially funded by the Defense Advanced Research Projects Agency (DARPA) Big Mechanism program under ARO contract W911NF-14-1-0395.

There are many many hands that have helped me along the way, and many pairs of shoulders I have stood on to get to where I am.

Thank you to the professors who encouraged me to think I could do this, and who equipped me with the knowledge and skills to actually succeed! Thank you to my co-advisors for their assistance all along the way. Mihai, you invested much into my education and encouraged me to pursue my own ideas, while also allowing me to pursue some of yours. Working in your lab changed the course of my education, and subsequently my future – thank you! Peter, thanks for all the time you spent in the lab teaching me to tie my proverbial shoes, the papers are what they are because of you!

I also want to thank my parents for instilling in me the understanding that an education is priceless and encouraging me from the beginning. Emily, you set a great example, and I am delighted I got to follow in your footsteps. Nora, Isaac, and River – you have all been a part of this journey (River may as well be a co-author!) and I wouldn't have wanted it any other way! Dave – this absolutely would not have been remotely possible without you, and we both know that. I love you, and am beyond grateful for *everything* you have done for me and given up so I could do this – thank you so much!

DEDICATION

Dedicated to Nora, Issac, and River. Achieve great things, but do not define yourselves by your achievements.

TABLE OF CONTENTS

LIST OF FIGURES	9
LIST OF TABLES	12
ABSTRACT	16
CHAPTER 1 INTRODUCTION	17
1.1 Robustness	19
1.1.1 Aligning questions and answers	20
1.1.2 Customizing approaches to a specific question type	23
1.2 Interpretability	24
1.2.1 Multiple choice science questions as a proving ground	25
1.2.2 Reranking justifications with weak supervision	27
1.3 Contributions	28
1.3.1 Contribution 1: Using discourse structures to generate arti- ficially aligned pairs for training question answering models	28
1.3.2 Contribution 2: Generating customized, task-specific word embeddings based on the question type	29
1.3.3 Contribution 3: Creating and ranking justifications for inter- pretable question answering	30
1.3.4 Contribution 4: Using neural networks to rank justifications for interpretable question answering	31
1.4 Overview	31
CHAPTER 2 RELATED WORK	33
CHAPTER 3 Using Free Text to Train Monolingual Alignment Models for Question Answering	38
3.1 Chapter overview	38
3.2 Related Work	39
3.3 Approach	40
3.4 Models and Features	42
3.5 Experiments	44
3.6 Results and Discussion	45
3.7 Conclusion	49

TABLE OF CONTENTS – *Continued*

CHAPTER 4	Creating Causal Embeddings for Question Answering	
	with Minimal Supervision	51
4.1	Chapter overview	52
4.2	Related Work	52
4.3	Chapter overview	54
4.4	Extracting Cause-Effect Tuples	55
4.5	Models	58
4.6	Direct Evaluation: Ranking Word Pairs	61
	4.6.1 Data	62
	4.6.2 Baselines	62
	4.6.3 Results	63
4.7	Indirect Evaluation: QA Task	67
	4.7.1 Data	67
	4.7.2 Models and Features	68
	4.7.3 Results	70
	4.7.4 Error Analysis	72
4.8	Conclusion	72
CHAPTER 5	Interpretable Question Answering: Building and Ranking Inter-	
	sentence Answer Justifications	74
5.1	Chapter Outline	75
5.2	Related Work	76
5.3	Approach	77
5.4	Focus Word Extraction	79
	5.4.1 Scores and weights	81
5.5	Text Aggregation Graphs	82
5.6	Text Aggregation Graph Features	86
	5.6.1 Features	86
	5.6.2 Modeling Different TAG Types Using Domain Adaptation	88
5.7	Learning Model	90
	5.7.1 Learning Algorithm	91
5.8	Experiments	93
	5.8.1 Data	93
	5.8.2 Tuning	94
	5.8.3 Baselines	95
	5.8.4 Results	97
5.9	Discussion	105
5.10	Error Analysis	108
	5.10.1 Broader Error Classes	111

TABLE OF CONTENTS – *Continued*

5.10.2	Summary of Errors	117
5.11	Conclusion	118
CHAPTER 6	Robust and interpretable: A neural approach	120
6.1	Chapter Outline	121
6.2	Related Work	121
6.3	Approach	123
6.4	Model and Features	124
6.4.1	Candidate Justification Retrieval	126
6.4.2	Feature Extraction	126
6.4.3	Neural Network	129
6.5	Experiments	130
6.5.1	Data and Setup	130
6.5.2	Baselines	131
6.5.3	Corpora	131
6.5.4	Model Tuning	132
6.6	Results	133
6.6.1	QA Performance	133
6.6.2	Justification Performance	135
6.6.3	Error Analysis	138
6.7	Conclusion	141
CHAPTER 7	CONCLUSION	143
CHAPTER 8	Causal Extraction Rules Appendix	148
REFERENCES	153

LIST OF FIGURES

1.1	In this work we focus on two desirable aspects of a question answering approach: robustness and interpretability. Here we plot several approaches, including the four that are the focus of this work (the colored dots, please see corresponding chapters for model details), in terms of these aspects. Note that the ideal model would be found in the upper right-hand corner – robust to language variation and easily interpretable by a human user.	18
3.1	An example of the alignments produced by the two discourse models. The sequential model aligns pairs of consecutive sentences, capturing intersentence associations such as <i>cider-apples</i> , and <i>orchard-autumn</i> . The Rhetorical Structure Theory based model generates alignment pairs from participants in all (binary) discourse relations, capturing both intrasentence and intersentence alignments, including <i>apples-orchard</i> , <i>cider-apples</i> , and <i>cider-autumn</i>	40
3.2	Overall performance for the two discourse-based alignment models (the Rhetorical Structure Theory based model and the sequential model), compared against the IR baseline, random baseline, and an embedding-based reranker. The x axis indicates the number of training documents used to construct all models. Each point represents the average of 10 samples of training documents.	46
4.1	Architecture of the causal convolutional network.	59
4.2	Precision-recall curve showing the ability of each model to rank causal pairs above non-causal pairs. The Look-up model has no data points beyond the 35% recall point.	63
4.3	Illustration of the indirect learning of associations that facilitates approximate inference. Note that in this illustrated example, the learned associations are noisy, but this mechanism is critical – it is what allows embedding models to generalize beyond the explicit training examples. The blue highlight indicates words embedded in the cause (target) space, and the yellow indicates words embedded in the effect (context) space.	64

LIST OF FIGURES – *Continued*

4.4	Precision-recall curve including the bidirectional variants of the causal models. For clarity, we do not plot the basic noise aware causal embedding model (cEmbedNoise), which performs worse than its bidirectional variant (cEmbedBiNoise, described in Section 4.7.3).	66
5.1	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.	77
5.2	Five example graphlets for five sentences that could be aggregated together in different combinations to justifiably answer the question <i>What tools could determine the speed of turtles walking along a path?</i> (Answer: <i>stopwatch and meter stick</i>). Each graphlet contains one or more information nuggets (grey boxes) composed of one or more terms. For example, the graphlet for the sentence <i>A stopwatch can be used to measure time</i> contains two information nuggets. Edges between nuggets within a graphlet are shown with arrows, where a subset of these edges are labelled (e.g., <i>EXAMPLE</i> , <i>INSTRUMENT</i>), and the rest are unlabelled.	83
5.3	Two example Text Aggregation Graphs (TAGs) that justifiably answer the question <i>What tools could determine the speed of turtles walking along a path</i> for the answer <i>a stopwatch and meter stick</i> . Asterisks (*) denote that a given term is either a question or answer focus word, while 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 (<i>speed</i>) and other words shared between the graphlets (<i>distance</i> , <i>measure</i>). (bottom) A three-sentence TAG, where the edges between graphlets connect entirely on shared words between the graphlets (<i>distance</i> , <i>measure</i> , <i>time</i>) that are not focus words.	85
5.4	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 <i>Q</i> or <i>A</i> , while the presence of other non-focus words shared between the two graphlets is marked with <i>X</i> . Lexical overlap within a category is highlighted.	88

LIST OF FIGURES – *Continued*

- 6.1 Architecture of our question answering approach. Given a question, candidate answer, and a free-text knowledge base as inputs, we generate a pool of candidate justifications, from which we extract feature vectors. We use a neural network to score each and then retain *only* the highest-scoring justification from the pool of candidates (i.e., max-pooling), thus selecting the current best justification. This justification score serves as the score for the candidate answer itself. The red border indicates the components that are trained online. 122
- 6.2 Detailed architecture of the model’s scoring component. The question, candidate answer, and justification are encoded (by summing their word embeddings) to create vector representations of each. These representations are combined in several ways to create a set of representation-based similarity features that are concatenated to additional explicit features capturing lexical overlap, discourse and IR information and fed into a feed-forward neural network. The output layer of the network is a single node that represents the score of the justification candidate. 124
- 6.3 Example showing the discourse feature extracted from a question, answer, and justification. Words occurring in the question are shown in blue and words occurring in the answer are shown in green. The boxes around the justification text indicate the elementary discourse units identified by the discourse parser, and the arrow indicates the found discourse relation (with the assigned relation label given in red). 128
- 6.4 Number of questions for which our complete model chooses a new justification at each epoch during training. While this is for a single random seed, we see essentially identical graphs for each random initialization. 138

LIST OF TABLES

1.1	Examples of questions from Yahoo! Answers, a community question answering website, that demonstrate the variable quality of the community-chosen <i>best</i> answer.	21
1.2	Example of an 8th grade science question with a justification for the correct answer that completes the needed inference between question and answer. Note the lack of direct lexical overlap present between the justification and the correct answer, demonstrating the difficulty of the task of finding justifications using traditional distant supervision methods.	25
1.3	Categories of questions and their relative frequencies as identified by Clark et al. (2013a). Retrieval-based questions (including <i>is-a</i> , 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 <i>solids melt when heated</i>). 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 here that we do not include diagram questions, as they require specialized spatial reasoning that is beyond the scope of this work.	26
3.1	The 18 discourse relation labels that can be assigned by the Rhetorical Structure Theory (RST) parser of Surdeanu et al. (2015).	41
3.2	Feature descriptions for alignment models and the embedding baseline.	42
3.3	Comparison of performance for the discourse model when all discourse relations are used versus when only the top 6 most frequent (<i>elaboration, attribution, background, contrast, joint, and same-unit</i>) are used.	48
4.1	Number of causal tuples extracted from each corpus.	57

LIST OF TABLES – *Continued*

4.2	Performance in the QA evaluation, measured by precision-at-one (P@1). The “Bi” suffix indicates a bidirectional model; the “Noise” suffix indicates a model that is noise aware. * indicates that the difference between the corresponding model and the IR + vEmbed baseline is statistically significant ($p < 0.05$), as determined through a one-tailed bootstrap resampling test with 10,000 iterations.	69
4.3	SVM weights learned for each of the features in the combination model IR + vEmbed + cEmbedBi. Recall that feature values themselves are all independently normalized to lie between 0.0 and 1.0. . .	71
4.4	Results of an error analysis performed on a random sample of 20 incorrectly answered questions showing the source of the error and the percentage of questions that were affected. Note that questions can belong to multiple categories.	71
5.1	Example of an elementary science question with a justification constructed by our approach (in this case, each sentence comes from a different dictionary resource). Note that while each sentence is relevant to the inference required to answer the question, neither is sufficient without the other. When combined, however, the sentences complete the necessary inference.	75
5.2	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: <i>Conc</i> refers to the psycholinguistic concreteness score, <i>Tag</i> refers to the focus word category (<i>FOCUS</i> signifies a focus word, <i>EX</i> an example word, <i>ATYPE</i> an answer-type word, and <i>ST</i> a stop word), <i>Score</i> refers to the focus word score, and <i>Weight</i> refers to the normalized focus word scores.	79
5.3	Features used to score candidate answer justifications represented as TAGs.	87
5.4	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.	98

LIST OF TABLES – *Continued*

5.5	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 IR baseline or long (3G) TAG justifications best combined using voting ensembles (denoted with a \cup). 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.	100
5.6	Example justifications from the IR baseline and their associated ratings.	101
5.7	An example TAG and justification rated as <i>good</i> . The two sentences connect on non-focus "other" shared words (e.g., <i>green</i> , <i>plant</i>) which are not found in the question or answer, but which are highly related to the focus words.	102
5.8	<i>At least one</i> justification performance for both IR and TAG models, reflecting the highest rating attained by at least one of the top six justifications for a given question.	103
5.9	Most useful knowledge resources for justifications classified as "good".	106
5.10	Proportion of good justifications with a given number of connecting word categories (Q, A, X) for both correct and incorrect answers. (Top 6)	108
5.11	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. . .	109
5.12	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.	110
5.13	Example of failure to extract appropriate focus words from the question.	111
5.14	Example of a question that needs more than two sentences to answer.	113

LIST OF TABLES – *Continued*

5.15	Example of a question that requires reasoning over a causal structure or process.	114
5.16	Example of a question that requires an understanding of the quantifiers in both the question and the answers.	114
5.17	Example of a question that requires an understanding of negation.	115
5.18	Example of a failure to recognize relatedness or equivalence of words.	116
6.1	Summary of the features calculated for each candidate justification.	125
6.2	Performance on the AI2 Kaggle questions, measured by precision-at-one (P@1). *s indicate that the difference between the corresponding model and the IR baseline is statistically significant (* indicates $p < 0.05$ and ** indicates $p < 0.001$) and †s indicate significance compared to IR ⁺⁺ , All significance values were determined through a one-tailed bootstrap resampling test with 100,000 iterations.	132
6.3	Ablation of feature groups results, measured by precision-at-one (P@1) on validation data. Emb indicates our embedding-based features, LO indicates our lexical overlap features, lexDisc signifies our semi-lexicalized discourse features, and IR ⁺⁺ indicates our information retrieval based features. Significance is indicated as in Table 6.2.	133
6.4	Example justifications from the our model and their associated ratings.	136
6.5	Percentage of questions that have at least one <i>good</i> justification within the top 1 (Good@1) and the top 5 (Good@5) justifications, as well as the normalized discounted cumulative gain at 5 (NDCG@5) of the ranked justifications. Significance indicated as in Table 6.2.	137
6.6	Summary of the findings of the 30 question error analysis. Examples of several categories are provided in separate tables. Note that a given question may fall into more than one category.	139
6.7	Example of the system preferring a justification for which all the terms were found in either the question or answer candidate, while the justification for the correct answer contained additional information necessary to fully explain the answer. (Justifications shown in italics)	139
6.8	Example of a question for which complex inference is required. In order to answer the question, you would need to assemble the event chain: cut grass left on the ground → grass decomposes → decomposed material provides nutrients.	140
6.9	Example of a question for which knowledge base noise (here, in the form of over-generalization) was an issue.	140

ABSTRACT

We address the challenging task of *computational natural language inference*, by which we mean bridging two or more natural language texts while also providing an explanation of how they are connected. In the context of question answering (i.e., finding short answers to natural language questions), this inference connects the question with its answer and we learn to approximate this inference with machine learning. In particular, here we present four approaches to question answering, each of which shows a significant improvement in performance over baseline methods. In our first approach, we make use of the underlying discourse structure inherent in free text (i.e. whether the text contains an *explanation*, *elaboration*, *contrast*, etc.) in order to increase the amount of training data for (and subsequently the performance of) a monolingual alignment model. In our second work, we propose a framework for training customized lexical semantics models such that each one represents a single semantic relation. We use *causality* as a use case, and demonstrate that our customized model is able to both identify causal relations as well as significantly improve our ability to answer causal questions. We then propose two approaches that seek to answer questions by learning to rank human-readable justifications for the answers, such that the model selects the answer with the best justification. The first uses a graph-structured representation of the background knowledge and performs information aggregation to construct multi-sentence justifications. The second reduces pre-processing costs by limiting itself to a single sentence and using a neural network to learn a latent representation of the background knowledge. For each of these, we show that in addition to significant improvement in correctly answering questions, we also outperform a strong baseline in terms of the quality of the answer justification given.

CHAPTER 1

INTRODUCTION

The ever-increasing proliferation of online text means that information is available on essentially any topic, but it also becomes increasingly difficult to find as more and more irrelevant information needs to be sifted through. As of the writing of this, the internet is estimated to have 1.2 billion websites (Real Time Statistics Project, 2017). To give a richer picture, English Wikipedia has about 5.5 million pages with more than 700 new articles being added each day (Wikipedia, 2017). The popular open-access scientific e-print service, arXiv¹ now has almost 1.3 million articles and receives over 10,000 new article submissions each month (arXiv, 2017), and the open-access biomedical database PubMed² contains well over 27 million articles.

Clearly, ever-improving tools are needed to handle this massive (and growing) abundance of information, and as the vast majority of this data is in the form of natural language (rather than a structured database), these tools must operate over natural language queries and natural language results. While certain queries and results can be anticipated using statistics, in order to handle a new query, the system must be able to *infer* what the desired result is from the query. While this natural language inference is at the heart of several natural language processing (NLP) tasks, it is particularly crucial for the task of question answering.

Question answering (QA), i.e., finding short answers to natural language questions, is one of the most important but challenging tasks on the road towards natural language understanding (Etzioni, 2011). As mentioned above, a QA system first faces the challenge that both the question and any potential answer sources are in the form of natural language, which inherently contains a large degree of lexical, syntactic, and even dialectal variation. Additionally, unlike search or information

¹<https://arxiv.org/>

² [<https://www.ncbi.nlm.nih.gov/pubmed>]

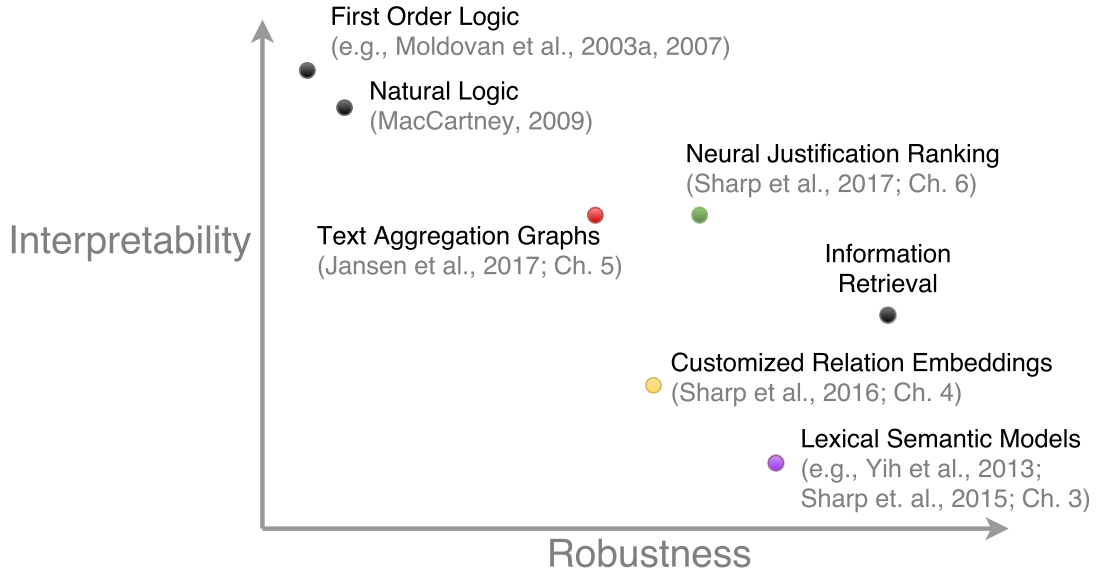


Figure 1.1: In this work we focus on two desirable aspects of a question answering approach: robustness and interpretability. Here we plot several approaches, including the four that are the focus of this work (the colored dots, please see corresponding chapters for model details), in terms of these aspects. Note that the ideal model would be found in the upper right-hand corner – robust to language variation and easily interpretable by a human user.

retrieval, answers often don’t contain lexical overlap with the question (e.g. *What should we eat for breakfast? – Zoe’s Diner has good pancakes*). This requires QA models to draw upon more complex methods to bridge this “lexical chasm” (Berger et al., 2000), i.e., to infer what the answer should be. These methods range from robust shallow models based on lexical semantics (e.g., the model of Yih et al. (2013) which is based on semantic relations such as *hypernymy* and *synonymy* as well as distributional similarity), to deeper, explainably-correct, but much more brittle inference methods based on first order logic (e.g., the Natural Logic of MacCartney (2009) that uses hand-crafted logic rules that operate over natural language). We exemplify this continuum in Figure 1.1, which relates examples of previous approaches (described in Chapter 2) to our four approaches (see Chapters 3 through 6) in terms of their robustness (Section 1.1) and their interpretability (Section 1.2).

1.1 Robustness

While first order logic methods are attractive for their formality, it is this same formality that renders them too brittle to be of much use outside of small, toy domains. In order to be used, these methods require parsing natural language into formal meaning representations (a task which is very difficult given the previously mentioned language variations) and then performing inference over these representations using logic rules and axioms, again made more difficult by the open-domain (i.e., questions can be about anything and of any level of complexity) and the fact that there are never enough rules and axioms to encode everything you need to know about the world and language. Here we focus on *approximating* this formal logic inference using natural language and machine learning instead of formal representations and hand-crafted rules. While we lose some of the ability to *prove* an answer, we gain much-needed robustness to both language and domain.

Many of these shallower methods that are based on machine learning (rather than pre-written rules) require a large amount of supervised training data, that is, data for which the label is known. In the context of QA, these are questions for which we know the *true* answer(s)³. In many domains, however, this requirement is expensive as it is difficult to find large amounts of high-quality data (i.e., sufficiently diverse and sufficiently difficult questions, with complete and concise answers). Therefore, in this work, we emphasize methods which use semi- and distant supervision⁴, rather than needing a large amount of hand-generated training data. This emphasis allows these methods to be applicable across a wider variety of domains.

³The true, or gold, answer to a question can be debatable, and there are many situations in which there can be more than one correct answer. Typically, an answer is considered correct if a human expert labels it as such, but again, this is open to interpretation.

⁴With semi-supervision, you begin with a small amount of hand-labeled data and learn to label new examples automatically, expanding your pool of labeled examples. With distant supervision, the labeling is done automatically from the start, e.g., by aligning a database of known facts with free text instances containing portions of those facts. For example, if you have a fact triple such as (*Jane_Smith*, *president_of*, *College_X*), you can find all sentences that contain those three elements and count them all as correct instances of the relation *president_of*. This type of approach is able to scale to large data-sources, but is very prone to noise (e.g., not all sentences truly demonstrate the *president_of* relation).

Another potential problem associated with these shallower, machine learning-based methods is that with the relaxation of the formality, the explainability may also be lost, and with it the ability to understand *why* the model produces the answers it does. Without this understanding, users cannot easily discern sources of error or determine how to fix them (which is critical for industry applications, where users are typically not NLP experts). Additionally, if a model cannot explain why it makes the predictions it does, it does not allow a user to trust the prediction, which becomes more important in high-stakes scenarios, such as in medical or military applications. To address interpretability, we include models (Chapters 5 and 6, see also Section 1.2) that use a human-readable intermediate output generated by the model from background knowledge to provide an explanation for the inference performed by the model. This answer justification can be thought of as proxy for a formal proof, where the background knowledge serves as an approximation for the axioms and the method by which we form the justification serves in place of the inference rules. In this work, however, which only approximates formal inference rather than abiding by the rigid structure of first order logic, we gain the robustness of a statistical approach.

1.1.1 Aligning questions and answers

Berger et al. (2000) proposed that the "lexical chasm", or lack of lexical overlap that often exists between questions and answers, might be partially bridged by making use of techniques popular in machine translation, the task of automatically translating text from one language to another language. At a high level, in typical machine translation, a model is given a sentence in one language aligned with its translation in another language. The model keeps track of statistics such as how frequently words in each language co-occur in the sentence-translation pairs and uses these statistics to generate word-alignment scores. Berger et al. suggested repurposing these statistical machine translation models for QA. Instead of translating text from one language to another, these *monolingual* alignment models (i.e., models that align

Quality	Question and Community-Chosen Answer
High	Q: What causes a cat to lose its fur? A: It could be a dietary issue, such as an allergy, or a more serious physical condition...
Fair	Q: What causes you to fart alot? A: Broccoli
Poor	Q: What causes sheets to get little balls on them. I think it is called pilling? A: i've always wondered the same thing

Table 1.1: Examples of questions from Yahoo! Answers, a community question answering website, that demonstrate the variable quality of the community-chosen *best* answer.

words within the same language) learn to translate from question to answer⁵. Given the previously mentioned example question, *What should we eat for breakfast? - Zoe's Diner has good pancakes*, these models learn common associations from question terms such as *eat* or *breakfast* to answer terms like *kitchen*, *pancakes*, or *cereal*, and the strength of the association serves as a proxy for the validity of the inference.

While monolingual alignment models have enjoyed a good deal of recent success in QA (Berger et al., 2000; Echihiabi and Marcu, 2003a; Soricut and Brill, 2006; Riezler et al., 2007; Surdeanu et al., 2011; Yao et al., 2013), they have expensive training data requirements, requiring a large set of aligned in-domain question-answer pairs for training. In most domains these pairs are expensive to generate, and one of the current methodological challenges in QA is locating or building QA pairs that are high enough quality (in terms of answer completeness, question variety, and topic coverage) to be used for training and testing. Even large open-domain international evaluations and workshops such as the Text REtrieval Conference (TREC)⁶ and the Cross Language Evaluation Forum (CLEF),⁷ are often limited to sets of a few

⁵In practice, alignment for QA is often done from answer to question, as answers tend to be longer and provide more opportunity for association (Surdeanu et al., 2011).

⁶<http://trec.nist.gov>

⁷<http://www.clef-initiative.eu>

hundred factoid questions, many of which are highly related. As a result, for open domain QA one often makes use of Community Question Answering data from websites such as Yahoo! Answers or Stack Overflow, which offer tens of thousands of questions, but of highly variable quality, as demonstrated with the example questions in Table 1.1. While techniques have been developed to automatically generate new questions using syntactic rearranging, question templates, or deep neural networks (e.g., Heilman and Smith, 2010; Serban et al., 2016; Du et al., 2017), at the current state of the art they often create questions and answers that are overly simplistic, repetitive, and unlike natural language (often not even syntactically valid). Additionally, the neural network based methods for generation also often have heavy training requirements themselves. For low-resource languages or specialized domains like science or biology, often the only option is to enlist a domain expert to generate gold QA pairs – a process that is both expensive and time consuming. All of this means that only in rare cases are we accorded the luxury of having enough QA pairs to properly train an alignment model, and so these models are often underutilized or left struggling for resources.

Making use of recent advancements in discourse parsing (Feng and Hirst, 2012)⁸, in Chapter 3 we address this issue, and investigate whether alignment models for QA can be trained from *artificial* question-answer pairs generated from discourse structures imposed on free text. Specifically, we align the words from the nucleus and the satellite of each discourse relation in lieu of a question and answer. That is, given a sentence such as *Water is an efficient solvent because of this polarity*, which has the discourse relation $[Water\ is\ an\ efficient\ solvent] \rightarrow_{cause} [because\ of\ this\ polarity]$, we learn alignments between words from the discourse relation’s nucleus (e.g., *water* and *solvent*) and its satellite (e.g., *polarity*). This method can be thought of as a form of distant supervision, as we don’t have true labels for which free-text pairs are truly adequate proxies for question-answer pairs.

⁸Discourse parsing involves the segmentation of text into adjacent spans which are then recursively joined and assigned both a direction (which span is the head, and which is the dependent) and a label (e.g., *elaboration*, *contrast*, *attribution*, etc).

1.1.2 Customizing approaches to a specific question type

This alignment approach for QA can be considered as falling into a larger group of approaches which prefer answers that are closely related to the question, where the relatedness is determined by the associations of the alignment model (e.g., as briefly referred to in Section 1.1.1) or by associations provided by other lexical semantic models such as word embeddings (Yih et al., 2013; Jansen et al., 2014; Fried et al., 2015). While appealing for its robustness to natural language variation, this one-size-fits-all category of approaches does not take into account the wide range of distinct question types that can appear in any given question set (see, e.g., Table 1.3), and that are best addressed individually (Chu-Carroll et al., 2004; Ferrucci et al., 2010; Clark et al., 2013b) for a specific question set.

Given the variety of question types, we suggest that a better approach is to look for answers that are related to the question *through the appropriate relation*, e.g., a causal question, should have a cause-effect relation with its answer. For example, to score a candidate answer for a question such as *What do hurricanes cause?*, while general purpose word associations would give a high score to a sentences containing near associates such as *tornadoes* and *tropical storms*, we suggest that a better approach would be using a set of relation-specific word associations that would provide high scores for words that are in a cause-effect relation with *hurricane*, such as *flooding* or *damage*. Adopting this view, and working with embeddings as a mechanism for assessing relationship, in Chapter 4 we address a key question: how do we train and use task-specific embeddings cost-effectively? Using causality as a use case, we answer this question with a framework for producing causal word embeddings, i.e., a set of word embeddings which encode causality rather than similarity, with minimal supervision, and a demonstration that such task-specific embeddings significantly benefit causal QA.

1.2 Interpretability

Developing interpretable machine learning models, that is, models where a human user can *understand* what the model is learning, is considered by many to be crucial for ensuring usability and accelerating progress (Craven and Shavlik, 1996; Kim et al., 2015; Letham et al., 2015; Ribeiro et al., 2016). For many applications of question answering (QA), simply providing an answer (or even an answer alongside a list of word-associations and their corresponding model weights, as with the approaches briefly described in Section 1.1) is not sufficient. 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”). Additionally, a QA tool is 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. For this reason we include methods that are interpretable, i.e., able to *explain* why an answer is correct.

One approach to interpreting complex models is to make use of human-interpretable information generated by the model to gain insight into what the model is learning. We follow the intuition of Lei et al. (2016), whose two-component network first generates text spans from an input document, and then uses these text spans to make predictions. Lei et al. utilize these intermediate text spans to infer the model’s preferences. By learning these intermediate representations end-to-end with a downstream task (i.e., question answering or another task that the user is interested in), they are optimized to correlate with what the model learns is discriminatory for the task, and they can be evaluated against what a human would consider to be important.

Question: Which of these is a response to an internal stimulus?

- (A) A sunflower turns to face the rising sun.
- (B) A cucumber tendril wraps around a wire.
- (C) A pine tree knocked sideways in a landslide grows upward in a bend.
- (D) **Guard cells of a tomato plant leaf close when there is little water in the roots .**

Justification: Plants rely on hormones to send signals within the plant in order to respond to internal stimuli such as a lack of water or nutrients.

Table 1.2: Example of an 8th grade science question with a justification for the correct answer that completes the needed inference between question and answer. Note the lack of direct lexical overlap present between the justification and the correct answer, demonstrating the difficulty of the task of finding justifications using traditional distant supervision methods.

1.2.1 Multiple choice science questions as a proving ground

In Chapters 5 and 6, we apply this general framework for model interpretability to QA, and in particular to answering multiple-choice science exam questions (Clark, 2015), by generating a human-readable justification for each selected answer. An example science exam question is provided in Table 1.2 to demonstrate the role of the justification in explaining the correct answer choice as well as to illustrate the fact that retrieving such a justification is non-trivial due to lack of lexical overlap.

In addition to the general difficulties of QA previously described, this particular QA domain is challenging as approximately 70% of science exam questions have been shown to require complex forms of inference to solve (Clark et al., 2013a; Jansen et al., 2016) (and indeed, we also have difficulty with this, see Sections 5.10 and 6.6.3). In Table 1.3 we provide example questions from each of the three main categories of questions from Clark et al. (2013a), where the categories are based on the methods likely required to answer them correctly. Not only do the majority of questions require some form of inference to solve (i.e., there is little or no lexical overlap between question and answer), but there are few structured knowledge bases to support this inference, and also commonly incorrect answers that

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

Table 1.3: Categories of questions and their relative frequencies as identified by Clark et al. (2013a). 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 here that we do not include diagram questions, as they require specialized spatial reasoning that is beyond the scope of this work.

are high semantic associates of either the question or correct answer are included to “lure” students (or automated methods) away from the correct response.

Having answer choices at all, however, no matter how closely related, does make this task simpler than if we were given the same questions with no answer options (open-ended QA). With open-ended QA, the system first needs to find a large pool of candidate answer texts, with no guarantee that the correct answer is there. Additionally, while multiple-choice questions are straightforward to evaluate, a correct answer to an open-ended question may be expressed in a variety of ways, making automatic evaluation problematic. We do not explore these challenges here, as we instead isolate the task of learning approximate inference from a question to a given answer, but our approaches could be extended to include the candidate answer

generation and evaluation components needed for open-ended QA.

1.2.2 Reranking justifications with weak supervision

Within the domain of multiple-choice science QA, we propose two approaches that 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. Each of these approaches learn to both select and explain answers, when the only supervision available is for which answer is correct (but not how to explain it). Intuitively, our approaches choose the justifications that provide the most help towards ranking the correct answers higher than incorrect ones. More formally, our approaches alternate between using the current model to choose the highest scoring justifications for answers, and optimizing the answer ranking model given these justifications. Thus, for both of these approaches, for each question and candidate answer we gather (or create) a pool of potential justifications. Then we allow the model to learn how to rerank these justifications such that the highest-scoring justification for the correct answer is better than the highest-scoring justification for any of the incorrect answers. Crucially, for both approaches, these reranked texts serve as our human-readable answer justifications, and by examining them, we gain insight into what the model learned was useful for the QA task.

The first approach (Chapter 5) uses aggregation of information from several sources along with structured representations of the text to create justifications of the answer choices. In particular, we aggregate multiple sentences into hierarchical graph structures (called text aggregation graphs, Section 5.5) 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 which may or may not be relevant. These justifications provide the inference needed to answer the question.

The second (Chapter 6) uses a shallower approach without aggregation or structured representations. In this shallow approach, we consider only justifications which are single sentences from a corpus and we replace the graph representation of sen-

tences with embeddings and a small set of explicit features (that model lexical overlap, length, etc). These changes allow us to operate over larger text resources.

Despite the differences between these two approaches, however, each allows us to address the challenge of question answering while prioritizing interpretability of the model.

1.3 Contributions

With this work we tackle the challenging task of question answering, seeking methods which balance the (sometimes conflicting) demands of robustness and interpretability. We address these challenges with four approaches. In particular, the specific contributions of this work are:

1.3.1 Contribution 1: Using discourse structures to generate artificially aligned pairs for training question answering models

We demonstrate that by exploiting the discourse structure of free text, monolingual alignment models can be trained to surpass the performance of models built from expensive in-domain question-answer pairs. To this end, we compare two methods of discourse parsing: a simple sequential model, and a deep model based on Rhetorical Structure Theory (RST Mann and Thompson, 1988) and show that the RST-based method captures within and across-sentence alignments and performs better than the sequential model, but the sequential model is an acceptable approximation when a discourse parser is not available. The proposed methods are evaluated on two corpora, including a low-resource domain where training data is expensive (biology). We experimentally demonstrate that monolingual alignment models trained using our method considerably outperform state-of-the-art neural network language models in low resource domains.

1.3.2 Contribution 2: Generating customized, task-specific word embeddings based on the question type

We propose a methodology for generating causal word-embeddings (encoding causality rather than similarity) cost-effectively by bootstrapping cause-effect pairs extracted from free text using a small set of seed patterns, e.g., *X causes Y*. We then train dedicated causal word embedding (as well as two other distributional similarity) models over this data. Levy and Goldberg (2014) have modified the algorithm of Mikolov et al. (2013b) to use an arbitrary, rather than linear, context. Here we make this context *task-specific*, i.e., the context of a cause is its effect. That is, for a word like *hurricane* in the sentence *The hurricane caused extensive flooding and damage*, we replace the standard sliding window context (*the, caused, and extensive*) with the words which are the effects: *flooding* and *damage*. Further, to mitigate sparsity and noise, our models are bidirectional, and noise-aware (by incorporating the likelihood of noise in the training process). We implement a QA system that uses these causal embeddings to answer questions and demonstrate that they significantly improve performance over a strong baseline. Further, we show that causal embeddings encode complementary information to the standard (or *vanilla*) word-embeddings, even when trained from the same knowledge resources. We also analyze direct vs. indirect evaluations for task-specific word embeddings – evaluating our causal models both *directly*, in terms of measuring their capacity to rank causally-related word pairs over word pairs of other relations, as well as *indirectly* in the downstream causal QA task. In both tasks, our analysis indicates that including causal models significantly improves performance. However, from the direct evaluation, it is difficult to estimate which models will perform best in real-world tasks. Our analysis re-enforces recent observations about the limitations of word similarity evaluations (Faruqui et al., 2016): we show that they have limited coverage and may align poorly with real-world tasks.

1.3.3 Contribution 3: Creating and ranking justifications for interpretable question answering

We propose a method to construct graph-structured answer justifications (text aggregation graphs) through aggregating relevant information from multiple sources. Our graph structures model both intrasentence structures (i.e., syntax) and intersentence lexical overlap. Our empirical analysis demonstrates that modeling the contextual relevance of intersentence connections is crucial for good performance. This approach uses a latent-variable ranking perceptron algorithm that learns to jointly rank answers and justifications. The perceptron, as a linear model, is less prone to overfitting with limited training data and simple to implement and extend. We use a ranking (rather than classification) framework because with multiple choice questions, the incorrect answers may not be completely wrong (as with “lure” answers), so rather than trying to learn a binary classification, we attempt to learn a ranking of answer choices where the correct answer is ranked higher than all the incorrect ones. In this extension of a traditional ranking perceptron, since we do not have labels for the *quality* of the justification, we model this as a latent variable to be learned. 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. 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.

1.3.4 Contribution 4: Using neural networks to rank justifications for interpretable question answering

We propose a neural network variation of the general framework briefly described in Section 1.3.3 for learning to answer questions and select a justification using distant supervision – i.e., the answer ranking supervises the justification re-ranking. With this shallower method, we replace the graph-structured text representations with learned representations and remove the aggregation component, allowing for much faster processing and thus usage with much larger resources. In this way, this approach is more robust while but still maintains the interpretability. We investigate two distinct categories of features in this “little data” domain: explicit features, and learned representations. We show that, with limited training, explicit features perform far better despite their simplicity. We evaluate the performance over 8th grade science exam questions and show that even with the removal of much of the justification structure, we still demonstrate a large (+9%) improvement in justification selection over a strong information retrieval baseline, as well as maintaining near state-of-the-art performance on the QA task.

1.4 Overview

The rest of this work is organized as follows. In Chapter 2 we outline the previous work relevant to our task. We then detail our word-association approaches to question answering that emphasize robustness: in Chapter 3 we present our shallowest form of inference – an alignment approach whereby we generate artificially aligned texts to serve as a proxy for aligned question-answer pairs. Then in Chapter 4 we present our method for tailoring word-associations (here, in the form of word-embeddings) to a specific question-type, allowing for relation-specific inference. The next two chapters contain our methods which focus more on model-interpretability, providing a human-readable justification for each answer selected by the model. In Chapter 5 we detail an approach that uses aggregation to generate justifications for answers, extracts features based on a structured representation of the aggregated

justification, and then uses these features in a latent-variable reranking perceptron. The perceptron learns to simultaneously rank these justifications and answer candidates, thus providing the answer as well as the human-readable justification that completes the inference needed to connect the question to the correct answer. In Chapter 6 we modify this approach to operate over much larger resources by removing the aggregation component as well as the structured representations, meanwhile extending the learning framework to a non-linear neural network. Finally, in Chapter 7 we discuss the work as a whole as well as future directions.

CHAPTER 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 (as depicted in Figure 1.1).

On the shallower end of the spectrum, QA models can be constructed either from structured text, such as question–answer pairs, or unstructured text. These models are largely based on finding patterns in word association scores, which serve as a proxy for formal inference. For example, using one of the questions from Table 1.3, *Q: Which example describes an organism taking in nutrients? A: A girl eating an apple*, rather than knowing specifically that *girl* is an organism and *eating* is a manner of taking in nutrients, the model would look for a higher association between the question words *organism*, *taking* and *nutrients* and the correct answer words *girl*, *eating* and *apple* than between those same question words and words from an incorrect answer, such as *insect*, *crawling* and *leaf*. Understandably, these methods struggle with more complex inference and with lure answers which are close associates with the correct answer.

Within this category of association-based model, monolingual alignment models (Berger et al., 2000; Echihabi and Marcu, 2003a; Soricut and Brill, 2006; Riezler et al., 2007; Surdeanu et al., 2011; Yao et al., 2013) have been widely employed. These models utilize statistical machine translation techniques to learn *translations* not from one language to another, but rather from question words to words likely to be found in their answers. These alignment-based models, however, like their machine translation counterparts, require a large number of aligned text pairs for training. In the case of the alignment models, this training data consists of aligned question–answer pairs, a burden which often limits their practical usage. In Chapter

3 we address this burden by proposing 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 (Jansen et al., 2014; Sultan et al., 2014; Yih et al., 2013), on the other hand, have the advantage of being readily constructed from free text. These models use distributional similarity via high-dimensional dense word embeddings to create sets of similarity features. These features are intended to capture how related a question is to a given answer candidate, a proxy for likelihood of being correct. Fried et al. (2015) called these approaches first-order models because associations are explicitly learned for words that co-occur in text. In this way, if two words never co-occur, an association would never be *directly* learned¹. To achieve a form of approximated inference based on chaining together associations, they introduced a higher-order lexical semantics QA model where indirect associations are detected through traversals of the association graph. For example, the directly learned associations between word pairs like *virus* \leftrightarrow *infection* and *infection* \leftrightarrow *fever* would serve to form or strengthen the transitive association between *virus* \leftrightarrow *fever*.

These alignment and lexical semantic approaches, however do not take into account the wide variety of question types which often exist in any given question set. Therefore, they attempt to answer *all* questions with word-pair associations from the same set of standard (similarity-based) word embeddings. To address this, we propose an approach in Chapter 4 for training a dedicated set of word embeddings that are customized to a particular semantic relation (here, causality) and demonstrate that the semantic information contained in these customized embeddings is complementary to that which is contained in standard word embeddings and useful for answering questions of the corresponding type (again, here we address *causal*

¹While two words which never occur together would never have an association directly learned by the model, indirect learning takes place (which is, in a large part, what makes these models so robust). That is, during the training process, words which occur with similar context words are indirectly drawn closer together in the high-dimensional embedded space as they are each independently drawn closer to their overlapping context words. For example, words like *dog* and *platypus* will each be independently drawn closer to a common context word like *eat*, inevitably resulting in them being indirectly drawn closer to each other.

questions, though the method is readily extendable to other question types).

While these shallower (i.e., alignment and lexical semantic) approaches to QA have shown robust performance across a variety of tasks, one continuing disadvantage of these methods is that, even when a correct answer is selected, there is no clear human-readable justification for that selection. This limits our ability to effectively understand model performance and adjust for errors accordingly.

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 over sets of semantic rules (e.g., Moldovan et al., 2003a, 2007; Balduccini et al., 2008; MacCartney, 2009; Liang et al., 2013; Lewis and Steedman, 2013). Some formal systems have made use of answer-set programming (Tari and Baral, 2006; Baral et al., 2011, 2012; Baral and Liang, 2012) to computationally search through a set of answers to find one for which a formal proof (from question to answer) can be constructed. Still others have constructed semantic graph representations (based on semantic role information) of sentences and questions, attempting to resolve missing roles to find the answer (e.g., Banarescu et al., 2013; Sharma et al., 2015). However, the formal representations used in these systems, e.g., logic forms or semantic graphs, are both expensive to generate and tend to be brittle because they rely extensively on imperfect tools for unsolved problems (such as complete syntactic analysis and word sense disambiguation) as well as incomplete knowledge bases and rule sets.

In Chapter 5, we offer a directed graph representation for sentences (based on a simplification of the sentence syntax, see Section 5.5) as a shallower and consequently more robust approximation of those logical forms. In Section 5.8.4 we show that they are well-suited for the complex questions (see Section 1.2.1) we tackle. This approach allows us to aggregate information from a variety of knowledge sources to create complete, human-readable answer justifications. It is these justifications which we then rank in order to choose the correct answer, using a reranking perceptron extended to be able to learn to rank justifications and answers simultaneously

despite having supervision only for which answer is correct.

While this approach is shallower than many of the approaches based on formal representations, it still requires decomposing free text resources into graph-structured representations and performing a somewhat expensive aggregation step. In practice, this limits its use with extremely large text corpora. To address this limitation, in Chapter 6 we propose a shallower version of this model that eliminates the aggregation and uses a neural network architecture to learn a task-specific embedded representation of each of the question, answer, and candidate justification texts. These representations are then used alongside a small set of explicit features to re-rank the candidate justifications, given the question and answer. By removing the graph-based representation, we increase the robustness, allowing us to use the model with much larger text corpora, and consequently in the domains that require larger them. Additionally, by continuing to re-rank the candidate justifications, we maintain the model interpretability – since the top n justifications returned for each chosen answer provide insight into what the model learned was important for finding correct answers.

In both of these latter approaches, the way we have formulated our justification selection (as a re-ranking of knowledge base sentences) is related to, but yet distinct from the task of answer sentence selection (Wang and Manning, 2010; Severyn and Moschitti, 2012, 2013; Severyn et al., 2013; Severyn and Moschitti, 2015; Wang and Nyberg, 2015, *inter alia*). Answer sentence selection is typically framed as a fully or semi-supervised task for factoid questions (i.e., questions whose answers are limited to one or more facts, such as birth-dates or names), where the goal of the task is to correctly select a sentence that from a corpus that fully contains the answer text. For example, to answer the question *What is the capital of France?*, a system would attempt to select a sentence such as *In the French capital of Paris,* Our QA task, however, is unlike answer-sentence selection. Here, we have a variety of questions, many of which are non-factoid (i.e., questions whose answers are not facts, such as those based on *how* or *why* something happens). Additionally, we have no direct supervision for our justification selection (i.e., no labels as to

which sentences are good justifications for our answers), motivating our distant supervision approach where the performance on our QA task serves as supervision for selecting good justifications. Further, we are not actually looking for sentences that *contain* the answer choice, as with answer sentence selection, but rather sentences which close the "lexical chasm" (Berger et al., 2000) between question and answer. This distinction is demonstrated in the example in Table 1.2, where the correct answer does not overlap lexically with the question and only minimally with the justification. Instead, the justification serves as a bridge between the question and answer, filling in the missing information for the required inference.

CHAPTER 3

Using Free Text to Train Monolingual Alignment Models for Question Answering

For the task of question answering (QA), lexical semantic models have shown promise in bridging Berger et al. (2000)’s ”lexical chasm,” or the lack of lexical overlap between questions and their answers. In general, these models can be classified into alignment models (Echihabi and Marcu, 2003a; Soricut and Brill, 2006; Riezler et al., 2007; Surdeanu et al., 2011; Yao et al., 2013), which are based on the repurposing of machine translation models to operate over aligned question-answer pairs within a single language (and hence require structured training data), and language models (Jansen et al., 2014; Sultan et al., 2014; Yih et al., 2013), which operate over free text. This training data requirement of alignment models is their main limitation, as it is difficult to obtain in specialized domains or low-resource languages. In this Chapter, we focus on closing this gap in resource availability by proposing two inexpensive methods for training alignment models solely using free text by generating artificial question-answer pairs from discourse structures.

3.1 Chapter overview

The remainder of this Chapter is organized as follows. We discuss previous work in Section 3.2. Then, in Section 3.3 we describe our two approaches to generating pairs of artificially aligned text spans. Then, in Section 3.4 we describe our models and model features in more detail. In Section 3.5 we detail the experimental setup used to evaluate the proposed models on two corpora from different genres and domains: one from Yahoo! Answers and one from the biology domain, and two types of non-factoid questions: manner (questions about the way in which something happens, often beginning with “How”) and reason (questions about the reason something happens, typically beginning with “Why”). In Section 3.6 we provide the results, showing

that these alignment models trained directly from discourse structures imposed on free text improve performance considerably over an information retrieval baseline and a neural network language model trained on the same data. Finally, in Section 3.7 we draw conclusions about our approach.

3.2 Related Work

Discourse has been previously applied to QA to help identify answer candidates that contain explanatory text. For example, Verberne et al. (2007) conducted an initial analysis of using discourse features derived from Rhetorical Structure Theory (RST; Mann and Thompson, 1988) for answer candidate selection, and concluded that while discourse features appeared useful, automated discourse parsing tools were required to test the idea on a larger scale. Jansen et al. (2014) proposed an answer reranking model that used both shallow and deep discourse features. The shallow features were based on segmenting text at occurrences of a set of discourse markers (Marcu, 1997) (i.e., words that typically indicate a discourse boundary) such as *and*, *in*, *that*, *for*, *if*, *as*, *not*, *by*, and *but*. The deep features were based on discourse parsing the text. These features were used in combination to identify useful answer structures in large answer collections across different tasks and genres. Here we also use discourse to impose structure on free text, but we use this to create inexpensive knowledge resources for monolingual alignment. However, our work is conceptually *complementary* to that of Jansen et al. – where they explored largely unlexicalized discourse structures to identify explanatory text (i.e., their features do not incorporate the actual text itself), we use discourse to learn lexicalized models that encode semantic similarity.

Our work is conceptually closest to that of Hickl et al. (2006), who also created artificially aligned pairs, but for the task textual entailment (i.e., determining whether or not a given span of text entails another). Taking advantage of the structure of news articles, wherein the first sentence tends to provide a broad summary of the article’s contents, Hickl et al. generated text pairs to train an alignment model

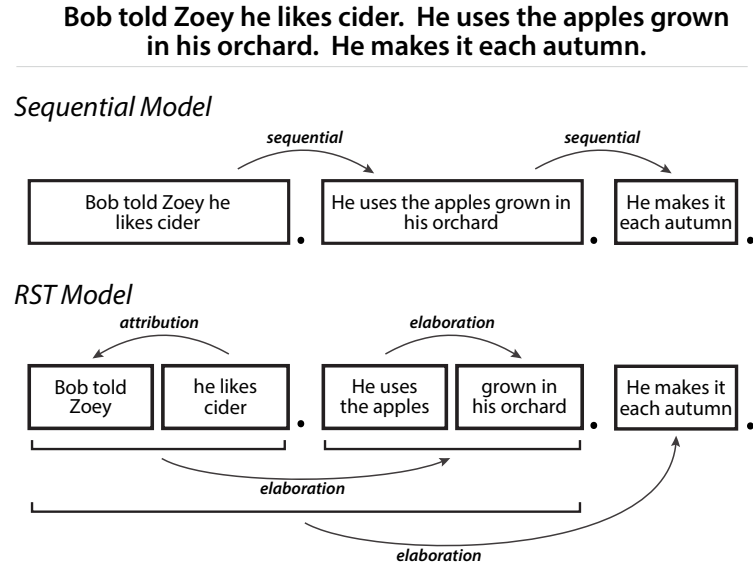


Figure 3.1: An example of the alignments produced by the two discourse models. The sequential model aligns pairs of consecutive sentences, capturing intersentence associations such as *cider-apples*, and *orchard-autumn*. The Rhetorical Structure Theory based model generates alignment pairs from participants in all (binary) discourse relations, capturing both intrasentence and intersentence alignments, including *apples-orchard*, *cider-apples*, and *cider-autumn*.

by aligning the first sentence of each article with its headline. By making use of automated discourse parsing, here we go further and impose alignment structure over an entire text, allowing us to gain more training data from our resources.

3.3 Approach

A written text is not simply a collection of sentences, but rather a flowing narrative where sentences and sentence elements depend on each other for meaning – a concept known as cohesion (Halliday and Hasan, 2014). Here we examine two methods for generating alignment training data from free text that make use of cohesion: a shallow method that uses only intersentence structures, and a deep method that uses both intrasentence and intersentence structures. We additionally attempt to separate the contribution of discourse from that of alignment in general by comparing these models against a baseline alignment model which aligns sentences at

Discourse Relations

attribution, background, cause, comparison, condition, contrast, elaboration, enablement, evaluation, explanation, joint, manner-means, topic-comment, summary, temporal, topic-change, textual-organization, same-unit

Table 3.1: The 18 discourse relation labels that can be assigned by the Rhetorical Structure Theory (RST) parser of Surdeanu et al. (2015).

random.

The first model, the sequential discourse model (SEQ), is based on the intuition that each sentence continues the narrative of the previous one, so for it we create artificial question-answer pairs from all pairs of consecutive sentences. Thus, this model takes advantage of intersentence cohesion by aligning the content words¹ in each sentence with the content words in the following sentence. For example, in the passage in Figure 3.1, this model would associate *cider* in the first sentence with *apples* and *orchard* in the second sentence.

The second model uses Rhetorical Structure Theory (RST) to capture discourse cohesion both within and across sentence boundaries. We extracted RST discourse structures using an in-house parser (Surdeanu et al., 2015), which follows the architecture introduced by Hernault et al. (2010) and Feng and Hirst (2012).

The parser first segments text into elementary discourse units, which may be at sub-sentence granularity, then recursively connects neighboring units with binary discourse relations, such as *Elaboration* or *Contrast* (see Table 3.1 for the full set of discourse relations).

We generate artificial alignment pairs from this imposed structure by aligning the governing text (nucleus) with its dependent text (satellite).² Turning again to the example in Figure 3.1, this RST-based model captures additional alignments that are both intrasentence, e.g., *apples-orchard*, and intersentence, e.g., *cider-autumn*.

¹In pilot experiments, we found that aligning only nouns, verbs, adjectives, and adverbs yielded higher performance.

²Pilot experiments showed that this direction of alignment performed better than aligning from satellite to nucleus.

Alignment Models

Global Alignment Probability:

$p(Q|A)$ according to IBM Model 1 (Brown et al., 1993)

Jenson-Shannon Distance (JSD) Features:

Pairwise JSDs were found between the probability distribution of each content word in the question and those in the answer. The **mean, minimum, and maximum JSD values** were used as features. Additionally, composite vectors were formed which represented the entire question and the entire answer and the **overall JSD** between these two vectors was also included as a feature. See Fried et al. (2015) for additional details.

Embedding Model

Cosine Similarity Features:

Similar to Jansen et al. (2014), we include as features the **maximum and average pairwise cosine similarity** between question and answer words, as well as the **overall similarity** between the composite question and answer vectors.

Table 3.2: Feature descriptions for alignment models and the embedding baseline.

Intuitively, we expect that some discourse relations would correspond with the specific question types: *How* questions may correspond with *elaboration* relations, *Why* questions with *explanation* relations, causal questions with *cause* relations, etc. On the other hand, some relations are more general (such as the *joint* relation, which indicates text contained in a list or disjunction), and would likely be applicable to all questions. While we leave this exploration of relations and question types to future work, we discuss addressing different question types individually with the approach proposed in Chapter 4.

3.4 Models and Features

We evaluate the contribution of these alignment models using a standard reranking architecture, as in Jansen et al. (2014), that first ranks answers with an informa-

tion retrieval (IR) score, then rereanks them using the concatenation of the model features and the IR score. The initial ranking of candidate answers is done using a shallow IR component that uses the cosine similarity between a query vector made from the question lemmas and the vector made from the answer candidate lemmas (each of these vectors is the sum of the one-hot³ vectors for the individual lemmas, weighted using *tf.idf* (Ch. 6, Manning et al., 2008)). Then, these answer candidates are reranked by a more expressive model that utilizes a feature vector composed of alignment features alongside the IR score. As a learning framework we use SVM^{rank} , a Support Vector Machine tailored for ranking.⁴ We compare this alignment-based reranking model against one that uses a state-of-the-art recurrent neural network language model (Mikolov et al., 2010, 2013a), which has been successfully applied to QA previously (e.g., Yih et al., 2013).

Alignment Model: The alignment matrices were generated with IBM Model 1 (Brown et al., 1993) using GIZA++ (Och and Ney, 2003), and they provide the likelihood that a pair of words align. With this method of generation, we consider only unigrams and word order is not used. The corresponding alignment models (i.e., that use the generated matrices) were implemented as per Surdeanu et al. (2011) with a feature for the global alignment probability – the conditional probability of observing an entire question given an entire answer, $P(Q|A)$. This feature is calculated as (equations from Surdeanu et al. (2011)):

$$P(Q|A) = \prod_{q \in Q} P(q|A) \quad (3.1)$$

$$P(q|A) = (1 - \lambda)P(q|A) + \lambda P(q|\mathbf{C}) \quad (3.2)$$

$$P(q|A) = \sum_{a \in A} (T(q|a)P(a|A)) \quad (3.3)$$

³One-hot vectors are so named because they consist of all zeros except for a single element, which has a value of 1.0. Here these are used to indicate the identity of words – the dimension of the vector is equal to the size of the vocabulary, and the location of the 1.0 indicates which vocabulary item the vector represents.

⁴ http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

where the probability of generating a given question word given the answer, $P(q|A)$, is smoothed by the prior probability of generating that question word from the entire collection of answers, \mathbf{C} , using the smoothing parameter λ . $T(q|a)$ is the probability that a given answer word, a , generates a question word, q , as provided by the trained alignment matrix. For more details, please see Surdeanu et al. (2011).⁵ We extend this alignment model with features from Fried et al. (2015) that treat each (source) word’s probability distribution (over destination words) in the alignment matrix as a distributed semantic representation, and make use of the Jensen-Shannon distance (JSD)⁶ between these conditional distributions. A summary of all these features is shown in Table 3.2.

Embed: We learned word embeddings using the `word2vec` recurrent neural network language model of Mikolov et al. (2013a), and include the cosine similarity-based features described in Table 3.2.

3.5 Experiments

We tested our approach in two different domains, open-domain and cellular biology. For consistency we use the same corpora as Jansen et al. (2014), which are described briefly here.

Yahoo! Answers (YA): Ten thousand open-domain *how* questions were randomly chosen from the Yahoo! Answers⁷ community question answering corpus and divided: 50% for training, 25% for development, and 25% for test. Candidate answers for a given question are selected from the corresponding answers proposed by the community (each question has an average of 9 answers).

Biology QA (Bio): 183 *how* and 193 *why* questions in the cellular biology domain were hand-crafted by a domain expert, and paired with gold answers in the

⁵We lightly tuned the smoothing parameter, λ to 0.4 and we redistributed the probability mass for each word such that the probability of a word translating to itself was 0.5.

⁶Jensen-Shannon distance is based on Kullback-Liebler divergence but is a distance metric (finite and symmetric).

⁷<http://answers.yahoo.com>

Campbell’s Biology textbook (Reece et al., 2011). Each paragraph in the textbook was considered as a candidate answer. As there were few questions, five fold cross-validation was used with three folds for training, one for development, and one for test.

In both set-ups, candidate answers were assigned a information retrieval (IR) score based on the lexical similarity of the question to the candidate answer (described in Section 3.4).⁸ That IR score served as both a strong baseline for the models (when used alone) as well as an additional feature to be used alongside the model features described in section 3.4.

Alignment Corpora: To train the alignment models we generated alignment pairs from two different resources: Annotated Gigaword (Napoles et al., 2012a) for YA, and the textbook for Bio. Each was discourse parsed with the RST discourse parser described in Section 3.3, which is implemented in the FastNLPPProcessor toolkit⁹, using the MaltParser¹⁰ for syntactic analysis.

3.6 Results and Discussion

Figure 3.2 shows the performance of the discourse models against the number of documents used to train the alignment model We used the standard implementation for precision at 1 (P@1; Manning et al., 2008) with the adaptations for Bio described in Jansen et al. (2014). We address the following questions.

How does the performance of the Rhetorical Structure Theory (RST) and sequential (SEQ) models compare? Comparing the two principal alignment models, the RST-based model significantly outperforms the sequential (SEQ) model by about 0.5% P@1 in both domains ($p < 0.001$ for Bio and $p < 0.01$ for YA). We report statistics performed at the endpoints, i.e., when all training data is used,

⁸In the Bio task, the IR score also incorporated a similarity between the question and the section containing the candidate answer paragraph.

⁹<http://github.com/sistanlp/processors>

¹⁰<http://www.maltparser.org/>

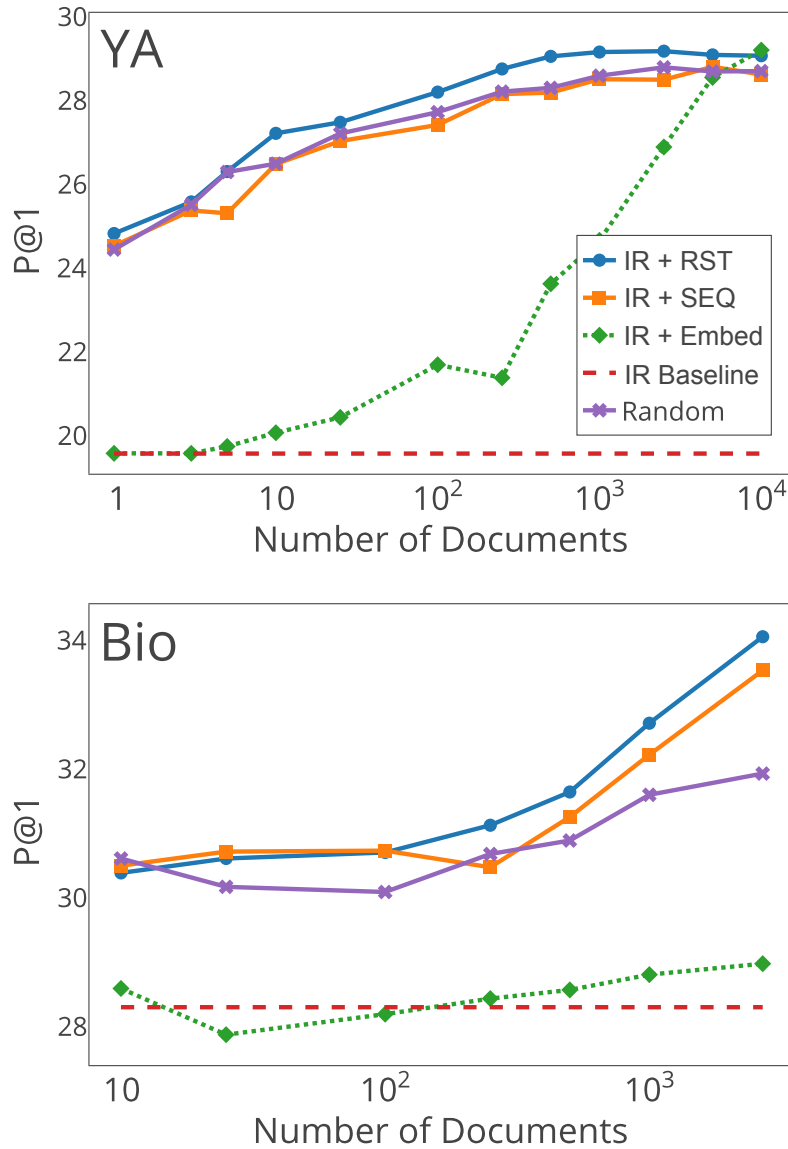


Figure 3.2: Overall performance for the two discourse-based alignment models (the Rhetorical Structure Theory based model and the sequential model), compared against the IR baseline, random baseline, and an embedding-based reranker. The x axis indicates the number of training documents used to construct all models. Each point represents the average of 10 samples of training documents.

using bootstrap resampling¹¹ with 10,000 iterations. This shows that deep discourse

¹¹ In this scenario, bootstrap resampling is implemented such that a sample dataset is randomly drawn (with replacement) from the true test set, and we determine if the experimental model

analysis (as imperfect as it is today) is beneficial.

How does the performance of the RST model compare to a model trained on in-domain question-answer pairs? Both the RST and SEQ results for YA are *higher* than that of an alignment model trained on explicit in-domain question-answer pairs. Fried et al. (2015) trained an identical alignment model using approximately 65k QA pairs from the YA corpus, and report a performance of 27.24% P@1, or nearly 2 points lower than our model trained using 10,000 Gigaword documents. This is an encouraging result, which further demonstrates that: (a) discourse analysis can be exploited to generate artificial semi-structured data for alignment, and (b) the sequential model, which also outperforms Fried et. al, can be used as a reasonable proxy for discourse when a parser is not available.

How does the performance of the RST model compare to previous work?

Comparing our work to Jansen et al. (2014), the most relevant prior work, we notice two trends. First, our discourse-based alignment models outperform their IR + Embed model, which peaks at 26.6% P@1 for YA and 31.7% for Bio *why*. While some of this difference can be assigned to implementation differences (e.g., we consider only content words for both alignment and Embed, where they used all words), this result again emphasizes the value of our approach. Second, the partially lexicalized discourse structures used by Jansen et. al to identify explanatory text in candidate answers perform better than our approach, which relies solely on lexicalized alignment. However, we expect that our two approaches are complementary, because they address different aspects of the QA task (structure vs. similarity).

How do the RST and SEQ models compare to the non-alignment baselines? In Bio, both the RST and SEQ alignment models significantly outperform the Embed and IR baselines ($p < 0.001$). In YA, the RST and SEQ models significantly outperform the IR baseline ($p < 0.001$), and though they considerably out-

performed the baseline model in this sample dataset. This sampling is repeated a large number of times, allowing us to determine if the difference between the models is statistically significant.

	YA	Bio
Discourse Relations	P@1	P@1
All	29.23	34.39
Top 6 (92% of data)	28.98	34.00

Table 3.3: Comparison of performance for the discourse model when all discourse relations are used versus when only the top 6 most frequent (*elaboration*, *attribution*, *background*, *contrast*, *joint*, and *same-unit*) are used.

perform the the Embed baseline for most training document sizes, when all 10,000 documents are used for training, they do not perform better. This shows that alignment models are more robust to little training data, but embedding models catch up when considerable data is available.

How does the SEQ model compare to the random baseline? In Bio, the SEQ model significantly outperforms the random baseline ($p < 0.001$) but in YA it does not. This is likely due to differences in the size of the document which was randomized. In YA, the sentences were randomized within Gigaword articles, which are relatively short (averaging 19 sentences), whereas in Bio the randomization was done at the textbook level. In practice, as document size decreases, the random model approaches the SEQ model.

What is the contribution of the individual discourse relations used in generating the alignment pairs? The RST parser itself performs better on relations which occur more frequently. For our experiments, we use only the relations that comprised at least 1% of the found discourse relations. This amounted to six (of 18 possible) relations: *elaboration*, *attribution*, *background*, *contrast*, *same-unit*, and *joint*. These six relations accounted for 92% of all relations in both domains. As shown in Table 6.3, the bulk of the performance comes from using these six most frequent discourse relations, and using all relations only slightly improves performance by 0.3% P@1.

Why does performance plateau in YA and not in Bio? With Bio, we exploit all of the limited in-domain training data, and continue to see performance improvements. With YA, however, performance asymptotes for the alignment models when trained beyond 10,000 documents, or less than 1% of the Gigaword corpus. Similarly, when trained over the entirety of Gigaword (two orders of magnitude more data), the Embed model improves only slightly, peaking at approximately 30.5% P@1 (or, a little over 1% P@1 higher). We hypothesize that this limitation comes from failing to take context into account. In highly technical domains, such as biology, words like *phosphorylation* are non-ambiguous, whereas in open domains, alignments such as *apple* – *orchard* may interfere with those from different contexts, e.g., *apple* – *computer*, and add noise to the answer selection process. This is empirically supported by the data: in Figure 3.2 the steep slope of the Bio corpus shows no signs of decreasing even beyond 10^3 documents, whereas in YA it begins to plateau after 10^2 documents. This suggests that by incorporating context into alignment models for QA, there is the potential to further capitalize on the remaining training data available.

3.7 Conclusion

We propose two inexpensive methods for training alignment models using solely free text, by generating artificial question-answer pairs from discourse structures. Our experiments indicate that these methods are a viable solution for constructing state-of-the-art QA systems for low-resource domains, or languages where training data is expensive and/or limited. Since alignment models have shown utility in other tasks (e.g. textual entailment), we hypothesize that these methods for creating inexpensive and highly specialized training data could be useful for tasks other than QA. Our generation tool, **Straw2Gold**, is available at: <http://nlp.sista.arizona.edu/releases/straw2gold>.

A limitation of this approach, like most lexical semantic approaches, is its application of the same set of alignments to all questions. As summarized in Table

1.3, however, question sets are not uniform, but rather contain a variety of question types which require different types of inference to answer. For example, consider the follow two questions about flooding: *What causes floods?* and *What are some types of floods?*. Though the questions are minimally different lexically, they are looking for very different answers. The approach described here, as with most lexical semantic approaches, would have identical association scores between the question word *floods* and a candidate answer word such as *rain*, despite the fact that *rain* is likely only appropriate in an answer for the first question.

In Chapter 4 we propose a framework for addressing this limitation while staying within the lexical semantic framework, by training a lexical semantic model for a specific, directed relation. Given the above example about flooding, under this proposed approach we would first identify the question types and then apply the corresponding lexical semantic model. That is, candidate answers to the first question would be scored with a *causal* model, while candidate answers to the second question would be scored with a model that was based on *examples of situations*. In this way, the association score between *flooding* and *rain* would be appropriate to the given question type.

CHAPTER 4

Creating Causal Embeddings for Question Answering with Minimal Supervision

A common model for question answering (QA) is that a good answer is one that is closely related to the question, where relatedness is often determined using general-purpose lexical models such as word embeddings. Here we argue that a better approach is to look for answers that are related to the question in a *relevant way*, according to the information need of the question (that is, the question type). In this way, a QA system might consist of many distinct solvers, each specialized to the inference needs of a particular question type, such as causal inference or reasoning over a process¹. Here we explore one framework that could be used for these solvers, which we use to answer causal questions. We argue that this general framework can be extended easily to handle other question types.

We are not the first to assert the need for specialized solving methods in QA. For example, Oh et al. (2013) incorporate a dedicated causal component into their system, and note that it improves the overall performance. However, their model is limited by the need for explicit lexical overlap between a causal construction found in their knowledge base and the question itself. Here, we develop a causal QA component that exploits *specialized word embeddings* that are task-specific (i.e., specifically learned to model the desired semantic relation, here *causality*). By using word embeddings, we can model the likelihood of a causal link between two texts

¹Not addressed here is *how* questions are to be classified, and it is very much open-ended. Not only is the *number* of classes open-ended, but also the *type* of the classes themselves – for example you could base them on domain (e.g., Biology versus Geology, etc.), or, as discussed here the inference type required (e.g., causal versus definitional, etc.), or some other dimension entirely. Even after determining a classification scheme, developing the classifier itself is certainly non-trivial, made more difficult by the fact that complex questions can potentially belong to several different classes, as described by Jansen et al. (2016). However, this is beyond the scope of the current work.

even when the exact lexical items were never observed together in the knowledge base, increasing the robustness to lexical variation.

To train our embeddings, we start by bootstrapping a large number of cause-effect pairs from free text using a small number of syntactic and surface patterns. We then use these bootstrapped pairs to build several task-specific embedding (and other distributional similarity) models that we then evaluate both directly on a causal-relation identification task and indirectly in a QA task focused on causal questions.

4.1 Chapter overview

The rest of the Chapter is organized as follows. In Section 4.2 we review related work. Then, in Section 4.4 we provide details concerning our semi-supervised methods for gathering a large number of cause-effect text pairs that we use for training our models. In Section 4.5 we describe our primary causal embedding models as well as several other variants. In Section 4.6 we evaluate these models directly on a causal-relation identification task. Then, in Section 4.7 we detail the experimental setup for our indirect evaluation: causal QA over a subset of Yahoo! Answers. We show that explicitly modeling causality improves performance in both the direct and the indirect tasks. In the QA task our best model achieves 37.3% P@1, significantly outperforming a strong baseline by 7.7% (relative). We also show that the results of these two evaluations are *not* identical: a given model’s performance on the direct evaluation does not necessarily transfer to the more complex, real-world QA task, as discussed in our conclusions in Section 4.8.

4.2 Related Work

There has been a vast body of work which demonstrates that word embeddings derived from distributional similarity are useful in many tasks, including question answering (see *inter alia* Fried et al., 2015; Yih et al., 2013). However, Levy et al. (2015) note that there are limitations on the type of semantic knowledge which is en-

coded in these general-purpose similarity embeddings. Therefore, rather than limit ourselves to the general purpose embeddings, here we build task-specific embeddings that are customized for answering causal questions.

Customized embeddings have been created for a variety of tasks. For example, with semantic role labelling, custom embeddings have been learned for semantic frames, their arguments, and the roles of the arguments (e.g., FitzGerald et al., 2015; Woodsend and Lapata, 2015). Riedel et al. (2013) used distant supervision to learn embeddings for binary relations (such as *employed_by*) and argument pairs (such as $\langle \textit{Jane Smith}, \textit{Harvard} \rangle$) in order to perform automated database completion. Similar to Riedel et al., we train embeddings customized for specific relations, but we bootstrap training data using minimal supervision (i.e., a small set of patterns) rather than relying on distant supervision and large existing knowledge bases. Additionally, while Riedel et al. represent *all* relations in a general embedding space, here we train a dedicated embedding space for just the causal relations.

For use in question answering, embeddings have been customized to have question words that are close to either their answer words (Bordes et al., 2014), or to structured knowledge base entries (Yang et al., 2014). While these methods are useful for QA, they do not distinguish between different types of questions, and as such their embeddings are not specific to a given question type.

Additionally, embeddings have been customized to distinguish functional similarity from relatedness (Levy and Goldberg, 2014; Kiela et al., 2015). In particular, Levy and Goldberg train their embeddings by replacing the standard linear context of the target word with context derived from the syntactic dependency graph of the sentence. The resulting embeddings, they argue, model functional similarity rather than topical similarity. For example, when given the word *turing*, whereas a traditional embeddings model returns top associates containing topically related words such as *non-deterministic* and *finite-state*, the dependency-based model returns other tests (i.e., *padding*, *hotelling*, and *hamming*). In this work, we make use of this extension to arbitrary context in order to train our embeddings with contexts derived from binary causal relations. We extract cause-effect text pairs such that

the cause text becomes the *target* text and the effect text serves as the *context*. This raises the issue of multiword target and context texts, which we address in Section 4.5.

Recently, Faruqui et al. (2016) discussed issues surrounding the evaluation of similarity word embeddings, including the lack of correlation between their performance on word-similarity tasks and “downstream” or real-world tasks like QA, text classification, etc. As they advocate, in addition to a direct evaluation of our causal embeddings (see Section 4.6), we also evaluate them independently in a downstream QA task (Section 4.7). We provide the same comparison for two alternative approaches (an alignment model and a convolutional neural network model), confirming that the direct evaluation performance can be misleading without the task-specific, downstream evaluation.

With respect to extracting causal relations from text, Girju et al. 2002 use modified Hearst patterns Hearst (1992) to extract a large number of potential cause-effect tuples, where both causes and effects must be nouns. However, Cole et al. 2005 show that these nominal-based causal relations account for a relatively small percentage of all causal relations, and for this reason, Yang and Mao (2014) allow for more elaborate argument structures in their causal extraction by identifying verbs, and then following the syntactic subtree of the verbal arguments to construct their candidate causes and effects. Additionally, Do et al. 2011 observe that nouns as well as verbs can signal causality. We follow these intuitions in developing our causal patterns by using both nouns and verbs to signal potential participants in causal relations, and then allowing for the entire dominated structures to serve as the cause and/or effect arguments.

4.3 Chapter overview

The rest of the Chapter is organized as follows. In Section 4.4 we provide details concerning our semi-supervised methods for gathering a large number of cause-effect text pairs that we use for training our models. In Section 4.5 we describe

our primary causal embedding models as well as several other variants. In Section 4.6 we evaluate these models directly on a causal-relation identification task. Then, in Section 4.7 we detail the experimental setup for our indirect evaluation: causal QA over a subset of Yahoo! Answers. We show that explicitly modeling causality improves performance in both the direct and the indirect tasks. In the QA task our best model achieves 37.3% P@1, significantly outperforming a strong baseline by 7.7% (relative). We also show that the results of these two evaluations are *not* identical: a given model’s performance on the direct evaluation does not necessarily transfer to the more complex, real-world QA task, as discussed in our conclusions in Section 4.8.

4.4 Extracting Cause-Effect Tuples

Because the success of embedding models depends on large training datasets (Sharp et al., 2015), and such datasets do not exist for open-domain causality, we opted to bootstrap a large number of cause-effect pairs from a small set of patterns. We wrote these patterns using Odin (Valenzuela-Escárcega et al., 2016), a rule-based information extraction framework which has the distinct advantage of being *expressive*. That is, while most open-source rule languages operate over one representation of text (e.g., GATE (Cunningham et al., 2011) generally operates over surface sequences, whereas Semgrex (Chambers et al., 2007) operates over syntactic dependencies), Odin has the flexibility to use either (or both) depending on the user’s need. For this work, we make use of rules that operate over both surface sequences as well as dependency syntax in the grammars introduced in steps (2) and (3) below.

Odin operates as a cascade, allowing us to implement a two-stage approach. First, we identify potential participants in causal relations, i.e., the potential causes and effects, which we term **causal mentions**. A second grammar then identifies actual causal relations that take these causal mentions as arguments.

We consider both noun phrases (NP) as well as entire clauses to be potential causal mentions, since causal patterns form around participants that are syntacti-

cally more complex than flat NPs. For example, in the sentence *The collapse of the housing bubble caused stock prices to fall*, both the cause (*the collapse of the housing bubble*) and effect (*stock prices to fall*) are more complicated nested structures. Reducing these arguments to non-recursive NPs (e.g., *The collapse* and *stock prices*) is clearly insufficient to capture the relevant context.

Formally, we extract our causal relations using the following algorithm:

(1) Pre-processing: Much of the text we use to extract causal relation tuples comes from Annotated English Gigaword (Napoles et al., 2012b). This text is already fully annotated and no further processing is necessary. We additionally use text from the Simple English Wikipedia², which we processed using the Stanford CoreNLP toolkit (Manning et al., 2014a) and the dependency parser of Chen and Manning (2014a).

(2) Causal mention identification: We extract causal mentions (which are able to serve as arguments, i.e., the cause or effect, in our causal patterns) using a set of rules (provided in Appendix 8) designed to be robust to the variety that exists in natural language. Namely, to find causal mentions that are noun phrases, we first find words that are tagged as nouns, then follow outgoing dependency links for modifiers and attached prepositional phrases³, to a maximum depth of two links. To find causal mentions that are clauses, we first find words that are tagged as verbs (excluding verbs which themselves were considered to signal causation⁴), then again follow outgoing dependency links for modifiers and arguments. We used a total of four rules to label causal mentions.

(3) Causal tuple extraction: After causal mentions are identified, a grammar scans the text for causal relations that have causal mentions as arguments. Different

²https://simple.wikipedia.org/wiki/Main_Page. The Simple English version was preferred over the full version due to its simpler sentence structures, which make extracting cause-effect tuples more straightforward.

³The outgoing dependency links from the nouns which we followed were: `nn`, `amod`, `advmod`, `cmod`, `dobj`, `prep_of`, `prep_with`, `prep_for`, `prep_into`, `prep_on`, `prep_to`, and `prep_in`.

⁴The verbs we excluded were: *cause*, *result*, *lead*, *create*.

Corpus	Extracted Tuples
Annotated Gigaword	798,808
Simple English Wikipedia	16,425
Total	815,233

Table 4.1: Number of causal tuples extracted from each corpus.

patterns have varying probabilities of signaling causation (Khoo et al., 1998). To minimize the noise in the extracted pairs, we restrict ourselves to a set of 13 rules designed to find unambiguously causal patterns, such as *CAUSE led to EFFECT*, where *CAUSE* and *EFFECT* are previously found causal mentions. The rules operate by looking for a *trigger* phrase, e.g., *led*, and then following the dependency paths to and/or from the trigger phrase to see if all required causal mention arguments exist.

At step 2, the identification of causal mentions, we place few restrictions on the causal mention syntactic subgraph, favoring recall since in step 3 we use high-precision rules. The rule set for step 4.4, in fact, was winnowed down after early experiments showed that many patterns which *can* indicate causation (e.g., *X happened due to Y*) often are non-causal.

Applying this causal grammar over Gigaword and Simple English Wikipedia produced 815,233 causal tuples, as summarized in Table 4.1. As bootstrapping methods are typically noisy, we manually evaluated the quality of approximately 250 of these pairs selected at random. Of the tuples evaluated, approximately 44% contained some amount of noise. For example, from the sentence

Except for Springer’s show, which still relies heavily on confrontational topics that lead to fistfights virtually every day...

while ideally we would only extract (*confrontational topics* \rightarrow *fistfights*), instead we extract the tuple (*show which still relies heavily on confrontational topics* \rightarrow *fistfights virtually every day*), which contains a large amount of noise: *show*, *relies*, *heavily*, etc. This finding prompted our noise-aware model described at the end of Section 4.5.

4.5 Models

We use the extracted causal tuples to train three distinct distributional similarity models that explicitly capture causality.

Causal Embedding Model (cEmbed): The first distributional similarity model we use is based on the skip-gram word-embedding algorithm of Mikolov et al. (2013b), which has been shown to improve a variety of language processing tasks⁵ including QA (Yih et al., 2013; Fried et al., 2015). In particular, we use the variant implemented by Levy and Goldberg (2014) which modifies the original algorithm to use an arbitrary, rather than linear, context. Our novel contribution is to make this context task-specific: intuitively, the context of a cause is its effect. Further, these contexts are generated from tuples that are themselves bootstrapped, which minimizes the amount of supervision necessary.

The Levy and Goldberg model trains using single-word pairs, while our causal mentions (CM) could be composed of multiple words. For this reason, we decompose each cause-effect tuple, (CM_c, CM_e) , such that each word $w_c \in CM_c$ is paired with each word $w_e \in CM_e$.

After filtering the extracted cause-effect tuples for stop words and retaining only nouns, verbs, and adjectives, we generated over 3.6M (w_c, w_e) word-pairs⁶ from the approximately 800K causal tuples.

The model learns two embedding vectors for each word, one for when the word serves as a target word and another for when the word serves as a context word. Here, since the relation of interest is inherently directional, *both* sets of embeddings are meaningful, and so we make use of both – the target vectors encode the effects of given causes, whereas the context vectors capture the causes of the corresponding effects. As an example, consider the causal relation between the words *wind* and *damage* and the lack of a causal relation between *wind* and *popcorn*. Given the previously described embeddings, after training we expect that the target (cause)

⁵Chris Manning recently called embedding models the Sriracha sauce of NLP.

⁶For all models proposed in this section we used lemmas rather than words.

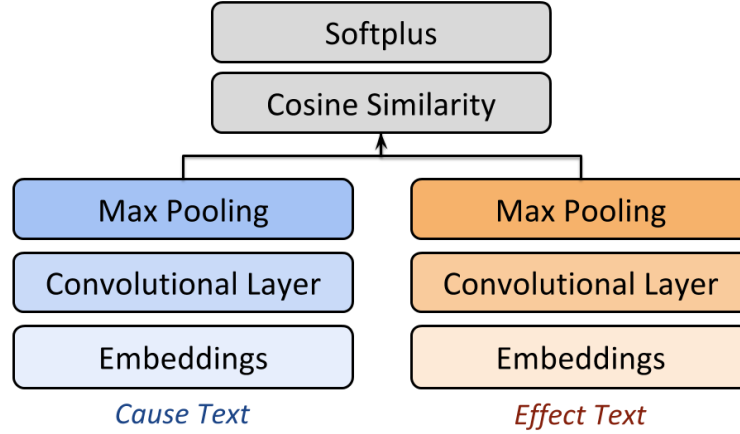


Figure 4.1: Architecture of the causal convolutional network.

vector for *wind* will be closer to the context (effect) vector for *damage* than to context vector for *popcorn*, indicating a more likely causal relation.

Causal Alignment Model (cAlign): Monolingual alignment (or translation) models have been shown to be successful in QA (Berger et al., 2000; Echihabi and Marcu, 2003b; Soricut and Brill, 2006; Riezler et al., 2007; Surdeanu et al., 2011; Yao et al., 2013), and in Chapter 3 we showed that they can be successfully trained with less data than embedding models (see also Sharp et al., 2015).

To evaluate these observations in this context, we train an alignment model that “translates” causes (i.e., the “source language”) into effects (i.e., the “destination language”), using our cause–effect tuples. This is done using IBM Model 1 (Brown et al., 1993) and GIZA++ (Och and Ney, 2003).

Causal Convolutional Neural Network Model (cCNN): Each of the previous models have at their root a bag-of-words representation, which is a simplification of the causality task. To address this potential limitation, we additionally trained a convolutional neural network (CNN) which operates over variable-length texts, and maintains distinct embeddings for causes and effects. The architecture of this approach is shown in Figure 4.1, and consists of two sub-networks (one for cause

text and one for effect text), each of which begins by converting the corresponding text into 50-dimensional embeddings. These are then fed to a convolutional layer,⁷ which is followed by a max-pooling layer of equal length. Then, these top sub-network layers, which can be thought of as a type of phrasal embedding, are merged by taking their cosine similarity. Finally, this cosine similarity is normalized by feeding it into a dense layer with a single node which has a softplus⁸ activation. In designing our CNN, we attempted to minimize architectural and hyperparameter tuning by taking inspiration from Iyyer et al. (2015a), preferring simpler architectures. We train the network using a binary cross entropy objective function and the Adam optimizer (Kingma and Ba, 2014), using the Keras library (Chollet, 2015) operating over Theano (Theano Development Team, 2016), a popular deep-learning framework.⁹

Noise-aware Causal Embedding Model (cEmbedNoise): We designed a variant of our cEmbed approach to address the potential impact of the noise introduced by our bootstrapping method (see discussion in Section 4.4). While training, we weigh the causal tuples by the likelihood that they are truly causal, which we approximate with pointwise mutual information (PMI). For this, we first score the tuples by their causal PMI and then scale these scores by the overall frequency of the tuple (Riloff, 1996), to account for the PMI bias toward low-frequency items. That is, the score S of a tuple, t , is computed as:

$$S(t) = \log \frac{p(\text{tuple}|\text{causal})}{p(\text{tuple})} * \log(\text{freq}(\text{tuple})) \quad (4.1)$$

We then discretize these scores into five quantiles, ascribing a linearly decreasing

⁷The convolutional layer was lightly tuned to contain 100 filters, had a filter length of 2 (i.e., capturing bigram information), and an inner ReLU activation.

⁸The softplus is a smooth approximation of the ReLU, or rectified linear unit.

⁹We also experimented with an equivalent architecture where the sub-networks are implemented using long short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997), and found that they consistently under-perform this CNN architecture. Our conjecture is that CNNs perform better because LSTMs are more sensitive to overall word order than CNNs, which capture only local contexts, and we have relatively little training data, which prevents the LSTMs from generalizing well.

weight during training to data in lower scoring quantiles.

We explore these different relation-specific distributional similarity models to determine their success at representing causal inference. The embedding models as well as the alignment model are simpler than the convolutional neural network, and so seemingly more amenable to the size of our training data, but they are potentially limited by their bag-of-words nature. Recall that the extracted cause-effect tuples may be multi-word phrases, and that the method we have used to convert them to single-word pairs inherently adds some noise (e.g., through *extreme rain* can cause *flooding*, it is not the case that *extreme* causes *flooding*). We evaluate each of these models both directly with a causal-relation identification task (Section 4.6) as well as indirectly in a relevant QA task consisting of causal questions (Section 4.7).

4.6 Direct Evaluation: Ranking Word Pairs

We begin the assessment of our models with a *direct* evaluation to determine whether or not the proposed approaches capture causality better than general-purpose word embeddings and whether their robustness improves upon a simple database look-up. For this evaluation, we follow the protocol of Levy and Goldberg (2014). To demonstrate that their embeddings encoded functional similarity rather than relatedness, they ranked a set of word pairs (each of which reflected one of the two types of similarity) using cosine similarity and showed that their dependency-based embeddings were better able to rank word pairs that were functionally similar higher than those with topical similarity, as compared with standard word embeddings. Here, we use the same setup.

In particular, we create a collection of word pairs, half of which are causally related, with the other half consisting of other relations. These pairs are then ranked by our models and several baselines, with the goal of ranking the causal pairs above the others. The embedding models rank the pairs using the cosine similarity between the target vector for the causal word and the context vector of the effect word. The alignment model ranks pairs using the probability $P(\text{Effect}|\text{Cause})$ given by IBM

Model 1, and the CNN ranks pairs by the value of the output returned by the network.

4.6.1 Data

In order to avoid bias towards our extraction methods, we evaluate our models on an external set of word pairs drawn from the SemEval 2010 Task 8 (Hendrickx et al., 2009), originally a multi-way classification of nine semantic relations¹⁰ between nominals. We used a total of 1730 nominal pairs, 865 of which were from the Cause-Effect relation (e.g., (*dancing* \rightarrow *happiness*)) and an equal number which were randomly selected from the other eight relations included in the SemEval data (e.g., (*juice* \rightarrow *grapefruit*), from the Entity-Origin relation). This set was then randomly divided into equally-sized development and test partitions.

4.6.2 Baselines

We compared our distributional similarity models against three baselines:

Vanilla Embeddings Model (vEmbed): a standard `word2vec` model trained with the skip-gram algorithm and a sliding window of 5, using the original texts from which our causal pairs were extracted.¹¹ As with the cEmbed model, SemEval pairs were ranked using the cosine similarity between the vector representations of their arguments.

Look-up Baseline: a given SemEval pair was ranked by the number of times it appeared in our extracted cause-effect tuples.

Random: pairs were randomly shuffled.

¹⁰The relations included in the SemEval task are (examples taken directly from Hendrickx et al. (2009)): Cause-Effect (e.g., *those cancers were caused by radiation exposures*), Instrument-Agency (*phone operator*), Product-Producer (*a factory manufactures suits*), Content-Container (*a bottle full of honey was weighed*), Entity-Origin (*letters from foreign countries*), Entity-Destination (*the boy went to bed*), Component-Whole (*my apartment has a large kitchen*), Member-Collection (*there are many trees in the forest*) and Message-Topic (*the lecture was about semantics*).

¹¹All embedding models analyzed here, including this baseline and our causal variants, produced embedding vectors of 200 dimensions.

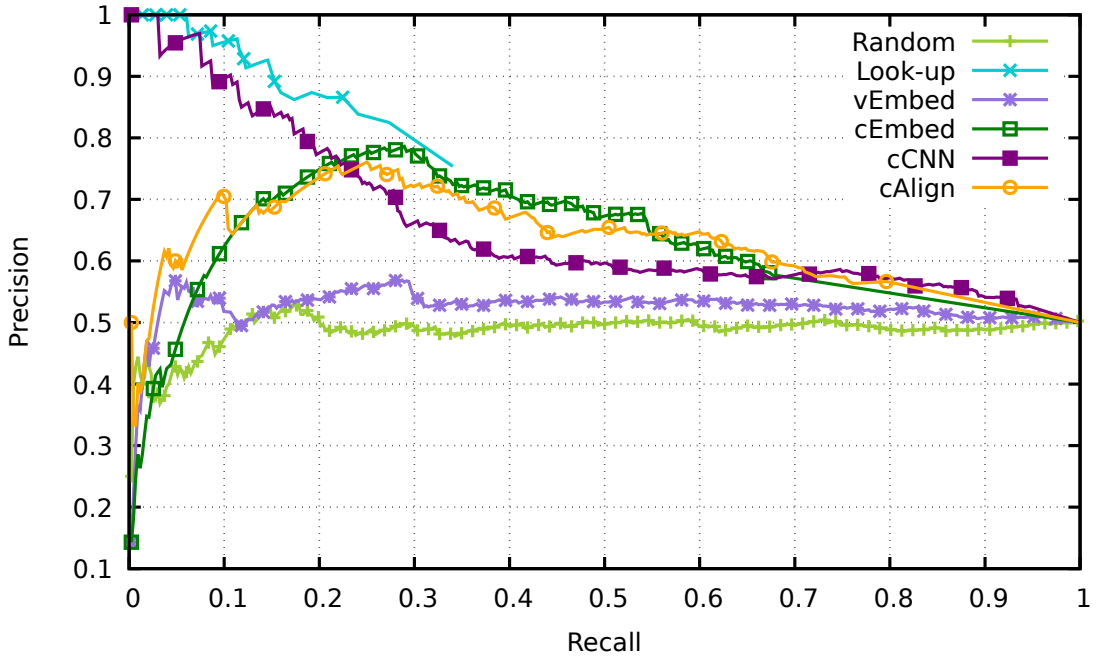


Figure 4.2: Precision-recall curve showing the ability of each model to rank causal pairs above non-causal pairs. The Look-up model has no data points beyond the 35% recall point.

4.6.3 Results

Figure 4.2 shows the precision-recall curve for each of the models and baselines. As expected, the causal models are better able to rank causal pairs than the vanilla embedding baseline (vEmbed), which, in turn, outperforms the random baseline. Our look-up baseline, which ranks pairs by their frequency in our causal database, shows a high precision for this task, but has coverage for only 35% of the causal SemEval pairs.

Some models perform better on the low-recall portion of the curve (e.g., the look-up baseline and cCNN), while the embedding and alignment models have a higher and more consistent performance across the precision-recall curve. We hypothesize that models that better *balance* precision and recall will perform better in a real-world QA task, which may need to access a given causal relation through a variety of

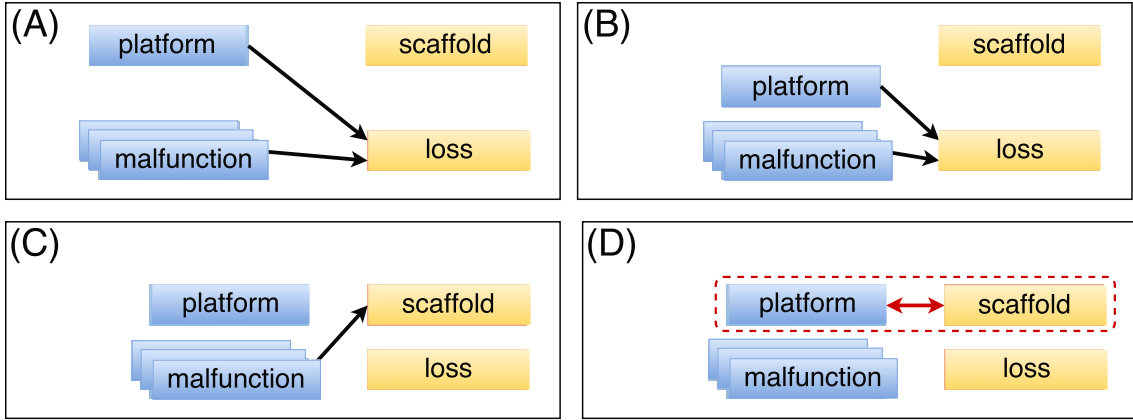


Figure 4.3: Illustration of the indirect learning of associations that facilitates approximate inference. Note that in this illustrated example, the learned associations are noisy, but this mechanism is critical – it is what allows embedding models to generalize beyond the explicit training examples. The blue highlight indicates words embedded in the cause (target) space, and the yellow indicates words embedded in the effect (context) space.

lexical patterns or variations. We empirically validate this observation in Section 4.7.

The precision-recall curve for the causal embeddings shows an atypical dip at low-recall. Though the precision-recall is not expected to be perfectly smooth, typically the low-recall portion of the curve (i.e., the left-most portion) shows the highest precision, as it indicates the data points for which the model was the most confident in its scoring. In this case, these represent the word pairs that received the *highest* causality score, yet the precision is *lower* than anywhere else on the curve – these highest-ranked pairs are, in fact, *not* causal.

To examine this dip, we analyzed its top-ranked 15% of SemEval pairs, and found that incorrectly ranked pairs were not found in the database of causal tuples. Instead, our analysis suggests that these pairs tend to be incorrectly ranked because the cEmbed model performs a form of soft or approximate inference through indirect learning of word-pair relations. This indirect learning is highly desirable (as it is what provides embedding models with their robustness), but it backfired on these data points.

We illustrate this with an example from the analysis, depicted in Figure 4.3.

Consider the top-ranked (incorrect) pair: (*platform* \rightarrow *scaffold*). As with all the incorrectly top-ranked pairs, there were no instances in our causal database where *platform* and *scaffold* were found together in the same cause-effect tuple. Instead, we found that there were three other extracted tuples with *scaffold* in the effect text. These tuple effect texts overlapped lexically with other tuples that had *platform* in the cause texts, which we suspect “pulled” *platform* closer to *scaffold*. For example, there are tuples of the form (... *platform*... \rightarrow ... *loss*...) and of the form (... *malfunction*... \rightarrow ... *loss*...), shown in panel A of Figure 4.3. This common co-occurrence with *loss* indirectly draws *malfunction* and *platform* closer together in the target (cause) embedding space (by drawing them each independently closer to *loss*), shown in panel B of Figure 4.3. Further, *malfunction* occurs in other tuples that have *scaffold* in their effect-text, drawing the target vector for *malfunction* closer to the context vector for *scaffold* (panel C), indirectly bringing the target vector for *platform* closer as well (panel D). While here, this constitutes semantic drift (Curran et al., 2007), as the association between *platform* and *scaffold* is erroneous, this same mechanism of indirectly learning associations is critical for robust models. Without it, we would not be able to generalize beyond our explicit training data, resulting in very poor coverage. In general, we found that these examples of semantic drift occur for low frequency items, whose embeddings could not be robustly estimated due to lack of direct evidence.

Because this sparsity is partially driven by directionality, we implemented a bidirectional embedding model (cEmbedBi) that (a) trains a second embedding model by reversing the input (effects as targets, causes as contexts), and (b) ranks pairs by the *average* of the scores returned by these two unidirectional causal embedding models. Specifically, the final bidirectional score of the pair, (e_1, e_2), where e_1 is the candidate cause and e_2 is the candidate effect, is:

$$s_{bi}(e_1, e_2) = \frac{1}{2}(s_{c \rightarrow e}(e_1, e_2) + s_{e \rightarrow c}(e_2, e_1)) \quad (4.2)$$

where $s_{c \rightarrow e}$ is the score given by the original causal embeddings, i.e., from cause to

effect, and $s_{e \rightarrow c}$ is the score given by the reversed-input causal embeddings, i.e., from effect to cause. The performance of the bidirectional models are shown in Figure 4.4.

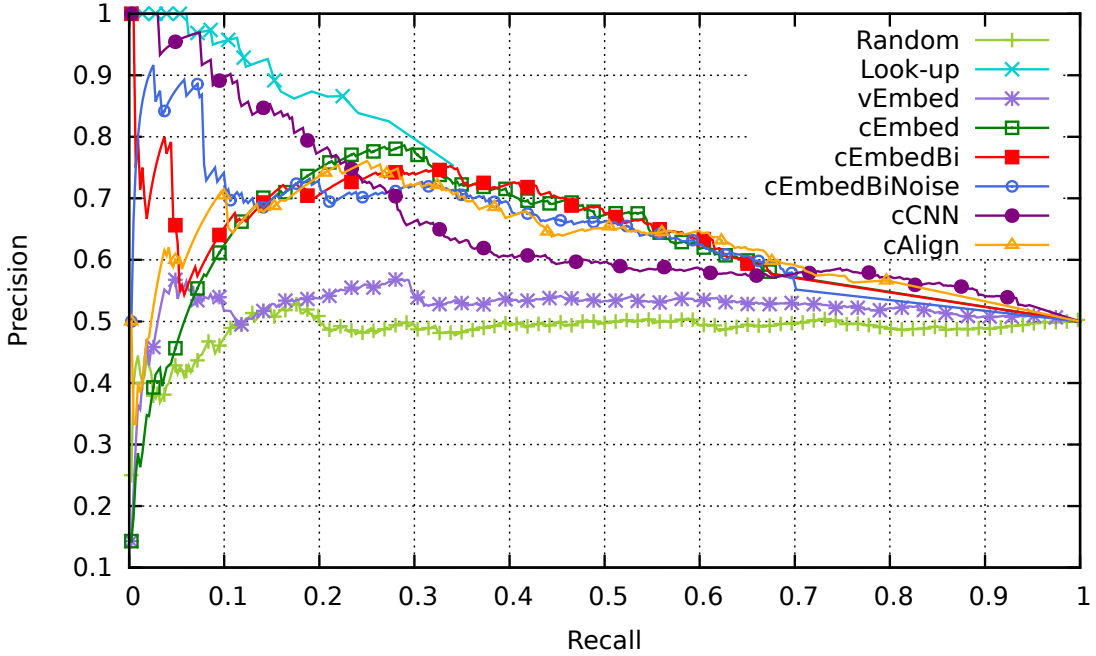


Figure 4.4: Precision-recall curve including the bidirectional variants of the causal models. For clarity, we do not plot the basic noise aware causal embedding model (cEmbedNoise), which performs worse than its bidirectional variant (cEmbedBiNoise, described in Section 4.7.3).

As Figure 4.4 shows, the bidirectional embedding variants consistently outperform their unidirectional counterparts. All in all, the best overall model is cEmbed-BiNoise, which is both bidirectional and incorporates the noise handling approach from Section 4.5. This model substantially improves performance in the low-recall portion of the curve, while also showing strong performance across the curve.

While important, this evaluation is somewhat superficial. A given model’s ability to directly infer causality between two isolated words does not necessarily transfer to a real-world task (Faruqui et al., 2016), where longer texts and noise interfere with prediction. Therefore, in addition to this direct evaluation, we also evaluate

the utility of our various models in the real-world task of question answering, as described in Section 4.7.

4.7 Indirect Evaluation: QA Task

The main objective of our work is to investigate the impact of a customized causal embedding model for question answering (QA). Following our direct evaluation, which solely evaluated the degree to which our models directly encode causality, here we evaluate each of our proposed causal models in terms of their contribution to a downstream real-world QA task.

Our QA system here uses the same standard reranking approach (Jansen et al., 2014) as in Chapter 3. In this architecture, the candidate answers are initially extracted and ranked using a shallow information retrieval (IR) component, then they are re-ranked using a “learning to rank” approach (see Section 3.4 for more details on the IR component and resulting IR score). In particular, we used SVM rank¹², a Support Vector Machines classifier adapted for ranking, and re-ranked the candidate answers with a set of features derived from both the initial IR score and the models we have introduced. For our model combinations (see Table 4.2), the feature set includes the IR score and the features from each of the models in the combination.

4.7.1 Data

We evaluate on a set of causal questions extracted from the Yahoo! Answers corpus¹³ with simple surface patterns such as *What causes ...* and *What is the result of ...*¹⁴. We extracted a total of 3031 questions, each with at least four candidate answers, and we evaluated performance using five-fold cross-validation, with three folds for training, one for development, and one for testing.

¹² http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

¹³Freely available through Yahoo!’s Webscope program (research-data-requests@yahoo-inc.com)

¹⁴We lightly filtered these with stop words to remove non-causal questions, such as those based on math problems and the results of sporting events. Our dataset is freely available, conditioned on users having obtained the Webscope license.

4.7.2 Models and Features

We evaluate the contribution of the bidirectional and noise-aware causal embedding models (cEmbedBi, and cEmbedBiNoise) as well as the causal alignment model (cAlign) and the causal convolutional neural network (cCNN). These models are compared against three baselines: the vanilla embeddings (vEmbed), the lookup baseline (LU), and additionally a vanilla alignment model (vAlign) which is trained over 65k question-answer pairs from Yahoo! Answers.

The features¹⁵ we use for the various models are:

Embedding model features: For both our vanilla and causal embedding models, we use the same set of features as in Section 3.4 (Fried et al., 2015): the maximum, minimum, and average pairwise cosine similarity between question and answer words, as well as the overall similarity between the composite question and answer vectors. When using the causal embeddings, since the relation is directed, we first determine whether the question text is the cause or the effect¹⁶, which in turn determines which embeddings to use for the question text and which to use for the candidate answer texts. For example, in a question such as "*What causes X?*", since *X* is the effect, all cosine similarities would be found using the effect vectors for the question words and the cause vectors for the answer candidate words.

Alignment model features: Again, as in Section 3.4, we use the same global alignment probability, $p(Q|A)$ of Surdeanu et al. (2011). In our causal alignment model, we adapt this to causality as $p(\text{Effect}|\text{Cause})$, and again we first determine the direction of the causal relation implied in the question. We once again include the additional undirected alignment features based on Jensen-Shannon distance, proposed more recently by Fried et al. (2015), in our vanilla alignment model. However, due to the directionality inherent in causality, they do not apply to our causal model so there we omit them.

¹⁵Due to the variety of features used, each feature described here is independently normalized to lie between 0.0 and 1.0.

¹⁶We do this through with simple regular expressions, e.g., "`^[Ww]hat ([a-z]+){0,3}cause.+`"

#	Model	P@1
Baselines		
1	Random	16.43
2	IR	24.31
3	IR + vEmbed	34.61
4	IR + vAlign	19.24
5	IR + Look-up (LU)	29.56
Single Causal Models		
6	IR + cEmbedBi	31.32
7	IR + cEmbedBiNoise	30.15
8	IR + cAlign	23.49
9	IR + cCNN	24.66
Model Combinations		
10	IR + vEmbed + cEmbedBi	37.08*
11	IR + vEmbed + cEmbedBiNoise	35.50*
12	IR + vEmbed + cEmbedBi + LU	36.75*
13	IR + vEmbed + cAlign	34.31
14	IR + vEmbed + cCNN	33.45
Model Stacking		
15	IR + vEmbed + cEmbedBi + cEmbedBiNoise	37.28*

Table 4.2: Performance in the QA evaluation, measured by precision-at-one (P@1). The “Bi” suffix indicates a bidirectional model; the “Noise” suffix indicates a model that is noise aware. * indicates that the difference between the corresponding model and the IR + vEmbed baseline is statistically significant ($p < 0.05$), as determined through a one-tailed bootstrap resampling test with 10,000 iterations.

Look-up feature: For the look-up baseline we count the number of times words from the question and answer appear together in our database of extracted causal pairs, once again after determining the directionality of the questions. If the total number of matches is over a threshold¹⁷, we consider the causal relation to be established and give the candidate answer a score of 1; or a score of 0, otherwise.

¹⁷Empirically determined to be 100 matches. Note that using this threshold performed better than simply using the total number of matches.

4.7.3 Results

The overall results are summarized in Table 4.2. Lines 1–5 in the table show that each of our baselines performed better than IR by itself, except for vAlign, suggesting that the vanilla alignment model does not generate accurate predictions for causal questions. The strongest baseline was IR + vEmbed (line 3), the vanilla embeddings trained over Gigaword, at 34.6% P@1. For this reason, we consider this to be the baseline to “beat”, and perform statistical significance of all proposed models with respect to it, using a one-tailed bootstrap resampling test with 10,000 iterations (see Section 3.6, Footnote 11 for implementation).

Individually, the cEmbedBi model is the best performing of the causal models. While the performance of cAlign in the direct evaluation was comparable to that of cEmbedBi, here it performs far worse (line 6 vs 8), suggesting that the robustness of embeddings is helpful in QA. Notably, despite the strong performance of the cCNN in the low-recall portion of the precision-recall curve in the direct evaluation, here the model performs poorly (line 9).

No individual causal model outperforms the strong vanilla embedding baseline (line 3), likely owing to the reduction in generality inherent to building task-specific QA models. However, comparing lines 6–9 vs. 10–14 shows that the vanilla and causal models are capturing different and complementary kinds of knowledge (i.e., causality vs. association through distributional similarity), and are able to be combined to increase overall task performance (lines 10–12). These results highlight that QA is a complex task, where solving methods need to address the many distinct information needs in question sets, including both causal and direct association relations. This contrasts with the direct evaluation, which focuses strictly on causality, and where the vanilla embedding baseline performs near chance. This observation highlights one weakness of word similarity tasks: their narrow focus may not directly translate to estimating their utility in real-world NLP applications.

Adding in the lookup baseline (LU) to the best-performing causal model does not improve performance (compare lines 10 and 12), suggesting that the bidirectional

Feature	SVM weight
IR	-0.19
vEmbed max similarity	0.021
cEmbed max similarity	0.828
vEmbed min similarity	-1.703
cEmbed min similarity	-0.445
vEmbed avg similarity	-1.842
cEmbed avg similarity	-2.177
vEmbed overall similarity	2.867
cEmbed overall similarity	1.725

Table 4.3: SVM weights learned for each of the features in the combination model IR + vEmbed + cEmbedBi. Recall that feature values themselves are all independently normalized to lie between 0.0 and 1.0.

Error/observation	% Q
Both chosen and gold are equally good answers	45%
Causal max similarity of chosen is higher	35%
Vanilla overall similarity of chosen is higher	35%
Chosen answer is better than the gold answer	25%
The question is very short / lacks content words	15%
Other	10%

Table 4.4: Results of an error analysis performed on a random sample of 20 incorrectly answered questions showing the source of the error and the percentage of questions that were affected. Note that questions can belong to multiple categories.

causal embeddings subsume the contribution of the LU model. cEmbedBi (line 10) also performs better than cEmbedBiNoise (line 11). We conjecture that the “noise” filtered out by cEmbedBiNoise contains distributional similarity information, which is useful for the QA task. cEmbedBi vastly outperforms cCNN (line 14), suggesting that strong overall performance across the precision-recall curve better translates to the QA task. We hypothesize that the low cCNN performance is caused by insufficient training data, preventing the CNN architecture from generalizing well.

Our best performing overall model combines both variants of the causal embedding model (cEmbedBi and cEmbedBiNoise), reaching a P@1 of 37.3%, which shows a 7.7% relative improvement over the strong IR + vEmbed baseline.

4.7.4 Error Analysis

We performed an error analysis to gain more insight into our model as well as the source of the remaining errors. For simplicity, we used the combination model IR + vEmbed + cEmbedBi. Examining the model’s learned feature weights (shown in Table 4.3), we found that the vanilla overall similarity feature had the highest weight, followed by the causal overall similarity and causal maximum similarity features. This indicates that even in causal question answering, the overall *topical* similarity between question and answer is still useful and complementary to the causal similarity features.

To determine sources of error, we randomly selected 20 questions that were incorrectly answered and analyzed them according to the categories shown in Table 4.4. We found that for 70% of the questions, the answer chosen by our system was as good as or better than the gold answer, often the case with community question answering datasets.

Additionally, while the maximum causal similarity feature is useful, it can be misleading due to embedding noise, low-frequency words, and even the bag-of-words nature of the model (35% of the incorrect questions). For example, in the question *What are the effects of growing up with an older sibling who is better than you at everything?*, the model chose the answer *... You are you and they are them - you will be better and different at other things...* largely because of the high causal similarity between (*grow* \rightarrow *better*). While this could arguably be helpful in another context, here it is irrelevant, suggesting that in the future improvement could come from models that better incorporate textual dependencies.

4.8 Conclusion

We presented a framework for creating customized embeddings tailored to the information need of causal questions. We trained three popular models (embedding, alignment, and CNN) using causal tuples extracted with minimal supervision by bootstrapping cause-effect pairs from free text, and evaluated their performance

both directly (i.e., the degree to which they capture causality), and indirectly (i.e., their real-world utility on a high-level question answering task).¹⁸

We showed that models that incorporate a knowledge of causality perform best for both tasks. Our analysis suggests that the models that perform best in the real-world QA task are those that have consistent performance across the precision-recall curve in the direct evaluation. In QA, where the vocabulary is much larger, precision must be balanced with high-recall, and this is best achieved by our causal embedding model. Additionally, we showed that vanilla and causal embedding models address different information needs of questions, and can be combined to improve performance.

Question classification is an unsolved problem, however, given a set of information needs deemed relevant to a domain and a way to classify questions according to these needs, we hypothesize that additional embedding spaces customized to the different information needs of questions would allow for robust performance over a larger variety of questions. We also assert that these customized embedding models would need to be evaluated both directly and indirectly to accurately characterize their performance.

Despite the robustness of this model, and the extensibility to other question types, this model does not particularly provide interpretability. This is demonstrated in the error analysis of Section 4.7.4 – as the only outputs of the model are feature values and learned feature weights, it’s difficult to explain to a human user (particularly a user who isn’t well-versed in machine learning) why the system is choosing the answers it is. This is addressed in Chapters 5 and 6 which follow, as each of these systems return a human-readable justification alongside the chosen answer to provide insight into what the model deems useful in determining a correct answer.

¹⁸All code and resources needed to reproduce this work are available at <http://clulab.cs.arizona.edu/data/emnlp2016-causal/>.

CHAPTER 5

Interpretable Question Answering: Building and Ranking Intersentence Answer Justifications

The ability of a user to understand why a question answering model chooses the answer that it does is critical both for having confidence in the model’s selections as well as for diagnosing its errors. Addressing this need, here we propose a question answering (QA) approach for standardized science exams that both identifies correct answers and produces complete, human-readable justifications for why those answers are correct, such as the example provided in Table 5.1.

Our method first identifies the actual information need in a question (i.e., the focus words, or the portion of the question that is relevant to the inference needed to answer it) using psycholinguistic concreteness norms, human-given ratings of how abstract or concrete a concept is (see Section 5.4). Once these focus words have been identified, we then use this information need to construct answer justifications by aggregating multiple sentences from different knowledge bases using syntactic and lexical information.

While we have labeled training data for which answer is correct, we do not have any for the justification creation. Therefore we model this justification quality as a latent variable to be learned, which we do by using the performance on the QA task as distant supervision. In this way, the model learns to select justifications which best allow it to answer questions correctly, and so the chosen justifications allow us to interpret the model as they provide insight into what the model learns to be informative.

Question: Which organism is a producer?

- (A) frog (C) **grass**
 (B) mushroom (D) lizard

Justification:

Producer is an organism that produces its own food and is food for other organisms: usually a green plant.

Grass is a green, leafy plant that often covers the ground.

Table 5.1: Example of an elementary science question with a justification constructed by our approach (in this case, each sentence comes from a different dictionary resource). Note that while each sentence is relevant to the inference required to answer the question, neither is sufficient without the other. When combined, however, the sentences complete the necessary inference.

5.1 Chapter Outline

The remainder of this chapter is organized in the following way. In Section 5.2 we discuss the relevant previous work. Section 5.3 describes our approach and how the various components of our system work together. Section 5.4 shows how we identify the portions of the question and answer that are relevant for performing the necessary inference – the focus words. Then in Section 5.5 we explain how we aggregate sentences from various sources to form our intersentence answer justifications. In Section 5.6 we detail the features extracted from these justifications that are used in our machine learning framework, which in turn is formally described in Section 5.7. Then, in Section 5.8 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.

Our best configuration answers 44% of the questions correctly, where the top justifications for 57% of these correct answers contain a high-quality human-readable justification, i.e. one that explains the inference required to arrive at the correct answer. The discussion of these results is provided in Section 5.9, then in Section 5.10 we include an error analysis that characterizes 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. Finally, we provide conclusions and future directions in Section 5.11

5.2 Related 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 et al., 1999; Barzilay and McKeown, 2005, *inter alia*), while in QA, aggregation has been used to assemble answers to both factoid questions (Pradhan et al., 2002) and definitional questions (Blair-Goldensohn et al., 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 which 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 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 5.8, this boosts overall

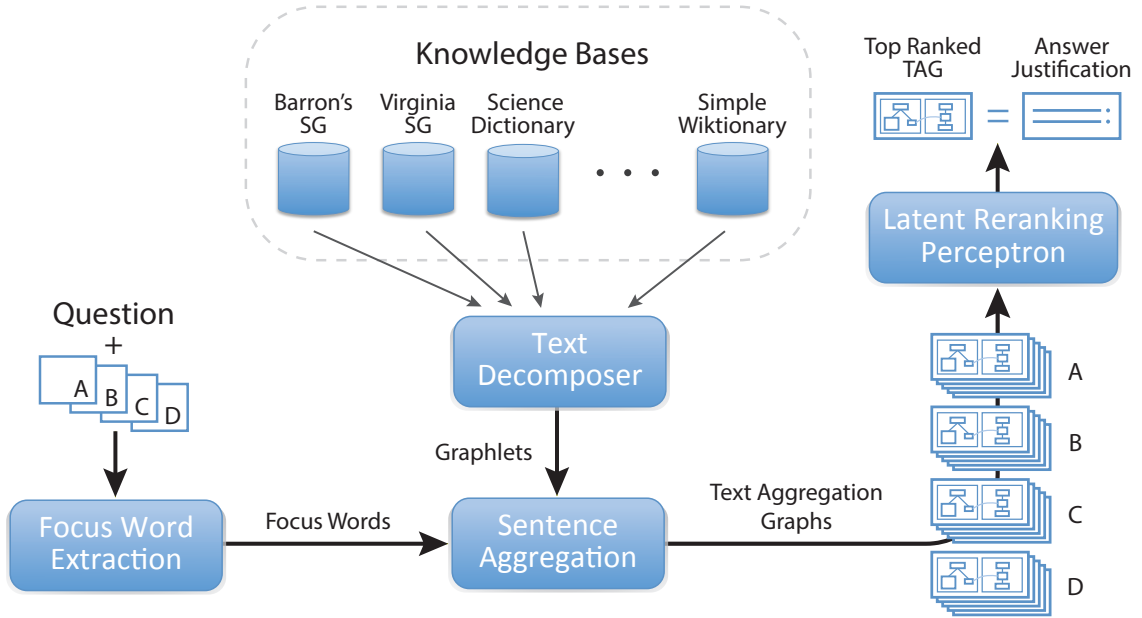


Figure 5.1: Our QA approach, which centers on the construction of answer justifications as text aggregation graphs, and ranking them using a model that treats the justification quality as a latent variable.

performance.

Once the candidate answer justifications are assembled, our method selects the answer which 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), which have been previously used in QA (Surdeanu et al., 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 et al., 2012; Björkelund and Kuhn, 2014), but to our knowledge, we are the first to adapt them to reranking scenarios.

5.3 Approach

The architecture of our proposed QA approach is illustrated in Figure 5.1, 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 (“graphlet”) that splits sentences on clausal and prepositional boundaries (Section 5.5). 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 1.3, and proceeds as follows. First, the questions are fed into a focus word extractor (Section 5.4) 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 5.5) 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 above 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, implemented as the sum of the weights of the focus words present in each justification¹. Using the focus words, we extract a number of features from each candidate justification that measure how well the justification answers the question (Section 5.6), and present this information to a latent-variable perceptron ranker (Section 5.7). 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. While the justification is expressed in the form of a text aggregation

¹We keep ties in this filtered list, which, due to 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.

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

Table 5.2: 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.

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.

5.4 Focus Word Extraction

In the domain of multiple choice elementary science exams, determining the portion of the question that is relevant to the inference required to answer it (i.e. the **focus words**) is not trivial. Often, extra information is included to distract readers and questions are grounded in an example as shown in Table 5.2, where a question about measuring speed is framed in terms of turtles walking along a path. Identifying these focus words, however, is critical for our approach for creating answer justifications through information aggregation – given the previously mentioned example, we would want to aggregate information about *measuring*, not information about *turtles*. Identifying, or extracting, focus words allows us to constrain our aggregation to greatly reduce the degree of justification noise, or semantic drift. Here we provide the description and formal algorithm for our focus word extraction, which is primarily the contribution of Dr. Peter Jansen, and included here only for completeness. For a more detailed discussion, please see Jansen et al. (2017).

We approach the task of selecting **focus words** by borrowing from cognitive

psychology, which suggests that elementary school children reason using concrete concepts (i.e., *turtles* rather than *object*), rather than with the more abstract thinking that develops later (Piaget, 1954). We use this distinction between *concrete* and *abstract* terms to guide our focus word extraction. Intuitively, we focus on words which are not too concrete (as they are likely to be example words, like *turtles*) and which are not too abstract (as they are likely to be too abstract for the focus of elementary-level questions).

To determine how concrete or abstract a given question word is, we make use of a large database of psycholinguistic concreteness norms (Brysbaert et al., 2014), which consists of 40 thousand common English lemmas rated on a numerical scale from 1 (highly abstract) to 5 (highly concrete). As described in Jansen et al. (2017), we empirically observed that 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 the right level of abstraction for isolating the key concepts of our elementary science exam questions².

Formally, we incorporate this observation into our focus word extractor that consists of the following five steps, each aimed at classifying question and answer words into a particular category.

1. **Identify lists and sequences:** We found that lists that occur in questions are often very important terms, and for conjunctive questions, matching all the list words can be critical. Therefore, we identify comma delimited lists (e.g., *sleet, rain, and hail*). Similarly, we identify from/to sequences (e.g., *from a solid to a liquid*, often found in causal or process questions) using paired `prep_from` and `prep_to` Stanford dependencies (De Marneffe and Manning, 2008).
2. **Identify focus words:** Content lemmas (nouns, verbs, adjectives, and adverbs) with concreteness scores between 3.0 and 4.2 in the concreteness norm

²While this “Goldilocks” zone works well for our elementary science questions, we expect that other question types would benefit from usage of these concreteness norms in a classifier to identify focus words (likely with a different critical range) as the distinction between concrete and abstract is fairly orthogonal to both frequency and *tf.idf*, which are commonly used to identify important terms.

database of Brysbaert et al. (2014) are flagged as focus words.

3. **Identify abstract, concrete, and example words:** We then flag (a) highly abstract words (content lemmas with concreteness scores between 1.0 and 3.0), (b) highly concrete words (concreteness scores between 4.2 and 5.0), and (c) example words (named entities that are durations or locations, e.g., *In New York State*).
4. **Identify answer type words:** Answer type words (i.e. *city* in the question *What city hosted the 2000 Olympics?*) are commonly considered to be the focus words in factoid QA as they highly constrain the search space. Here, however, since each of the multiple choice answers is of the same type, they do not help differentiate between correct and incorrect answers so we down-weight them. We flag them using syntactic patterns along with a short list of transparent nouns (e.g., *kind*, *type*, *form*).
5. **Identify stop words:** A list of general and QA-specific stop words and phrases, as well as any remaining words not captured by earlier steps, are flagged as stop words.

5.4.1 Scores and weights

After classifying each question and answer word, we assign each a score that denotes its importance with respect to the inference required to answer the question.

- **Stop words** are given a score of 0 and are excluded from future processing.
- **Answer type words** are given a score of 1.
- **Abstract, concrete, or example words** are sorted by how close they were to one of the concreteness boundaries (3.0 and 4.2). These words are then assigned incrementally increasing scores (starting at 2 for the farthest away word and continuing for all words). In this way, words which were closer to the boundary have higher scores.

- **Focus words** are all uniformly assigned a score that is 10 higher than the highest-scoring abstract, concrete, or example word.
- **List words** are uniformly given a score 1 higher than the focus words.

After all words are scored, the scores are normalized into weights so that they sum to one. An example of this scoring is shown in Table 5.2. While we do not claim that this algorithm is the best method for capturing focus words, we do demonstrate that this has a considerable impact on the performance of our overall QA system in the ablation studies (where individual aspects are systematically removed to determine their impact) discussed in Section 5.8.4.

5.5 Text Aggregation Graphs

In order to answer complex questions requiring inference, we propose a method for aggregating information from multiple knowledge base sources to form multi-sentence justifications that link questions to their answers. These human-readable justifications additionally provide a form of interpretability to the model, allowing the human user to see what the model considers important in identifying correct answers as well as where the model goes wrong. We maintain robustness during the aggregation process by making connections between sentences at the shallower word-level (i.e., we connect based on lexical overlap), rather than requiring more complex overlap. However, since the sentences we aggregate may come from different sources, when joining them together there is a risk of drifting to semantically unrelated concepts (consider if we joined a sentence about a bird’s *wings* and the *wings* of an auditorium). To help our model learn to disprefer incoherent justifications, though we aggregate at the word-level to maintain robustness, we retain the sentence-level context with our structured representation.

We represent our knowledge base sentences as lightly-structured graphs, which we call **graphlets**, and then aggregate them together based on lexical overlap to form our justifications. The graphlet nodes contain sentence text and the edges capture the syntactic dominance between nodes as well as certain specialized roles

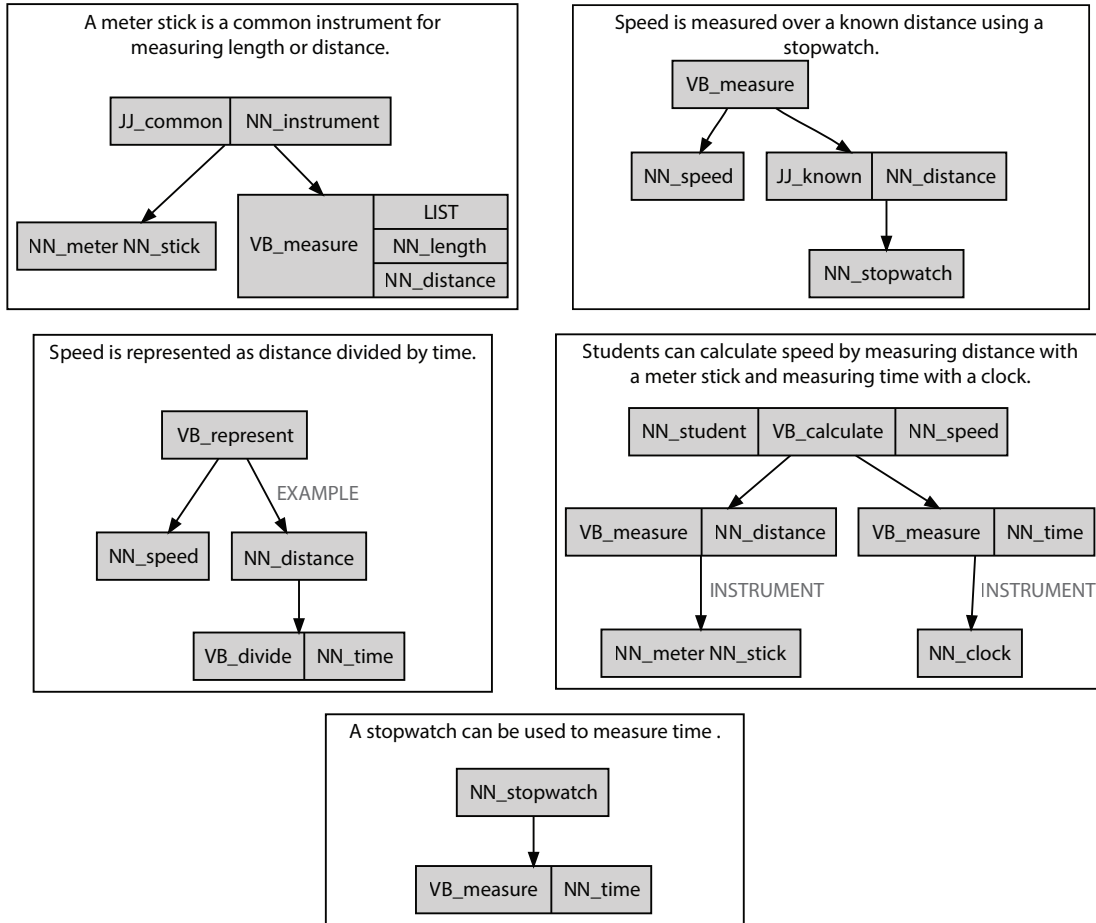


Figure 5.2: Five example graphlets for five sentences that could be aggregated together in different combinations to justifiably answer the question *What tools could determine the speed of turtles walking along a path?* (Answer: *stopwatch and meter stick*). Each graphlet contains one or more information nuggets (grey 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.

(e.g., *instrument* and *cause*), in the form of sparse edge labels. Example sentences and corresponding graphlets are shown in Figure 5.2. Below is a description of how we build our graphlets and then assemble them into our multi-sentence justifications, but for more detail, please see Jansen et al. (2017).

We build our graphlets in two stages. We first decompose knowledge base sentences into smaller units (along clausal and prepositional boundaries) which we call **information nuggets** (Voorhees, 2003). These smaller units reduce the sparsity of the whole sentence while still maintaining clausal context. Then we connect sentence nuggets together if any of their words were originally linked through syntactic dependencies. Formally, we use the following algorithm:

1. **Syntactic parsing:** The text is syntactically parsed using the sentences using the Stanford CoreNLP toolkit (Manning et al., 2014b).
2. **Decompose sentence into information nuggets:** The dependency graph is used to decompose the sentence into information nuggets (i.e., the graphlet nodes) by segmenting along specified dependencies that represent clausal complements (`ccomp`, `xcomp`), adverbial and relative clause modifiers (`advcl`, `rcmod`), reduced non-finite verbal modifiers (`vmod`), and certain empirically determined prepositional phrases (e.g., `prep_with`, etc.)³. The text within the node is segmented into terms, i.e. one or more words. While the default is one word, a term may contain a noun-noun compound or a comma-separated list.
3. **Construct graphlets:** These nuggets are then connected by edges whenever there is an outgoing dependency from one nugget to another. The edges are labeled as either `instrument`, `process`, `example`, `temporal`, `contrast`, or `unknown`, with the vast majority being labeled as `unknown`. These labels are assigned based on specified dependency links (i.e. the dependency link `prep_such_as` corresponds to the label `example`).⁴

³The 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`.

⁴More 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`.

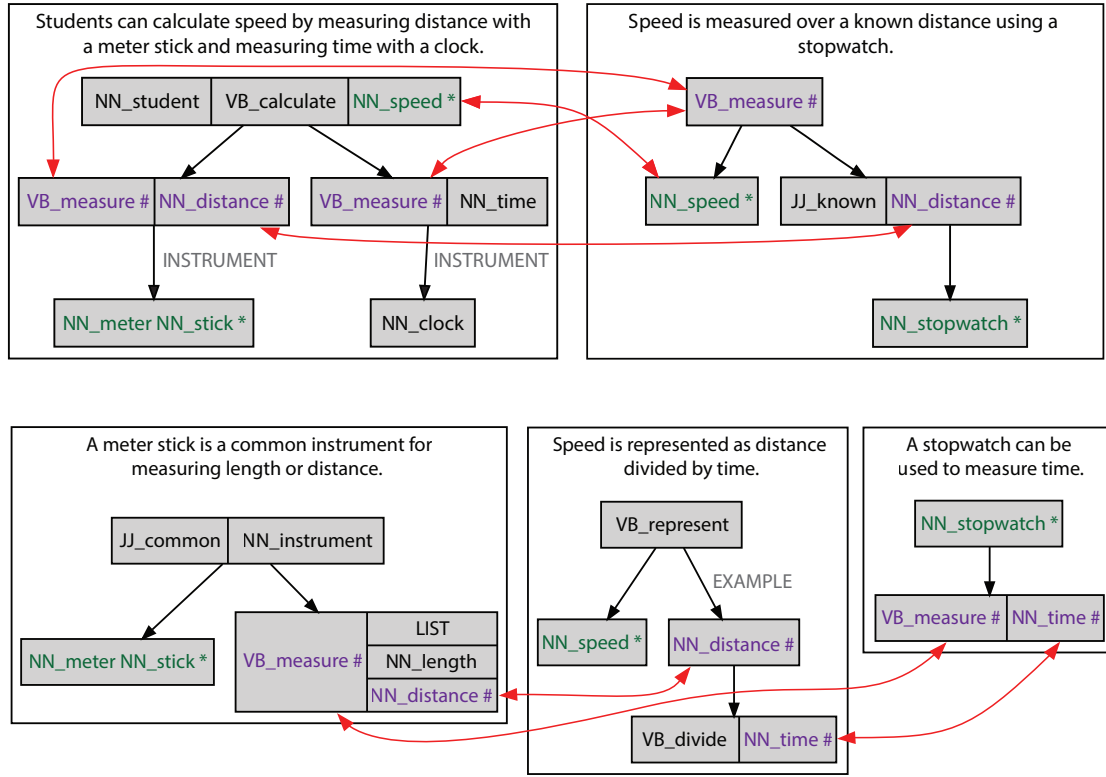


Figure 5.3: Two example Text Aggregation Graphs (TAGs) that justifiably answer the question *What tools could determine the speed of turtles walking along a path* for the answer *a stopwatch and meter stick*. Asterisks (*) denote that a given term is either a question or answer focus word, while pounds (#) 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.

4. **Remove stop words:** Finally, stop words as well as words that are not nouns, verbs, or adjectives are removed.

Once we have created the graphlet representations of our knowledge base sentences, we connect sentence graphlets that have any lexical overlap between their nuggets. The resulting, joined graph structure we call a **text aggregation graph**

(**TAG**) and this TAG serves as a potential answer justification.⁵ Figure 5.3 shows an example of two TAGs. We make use of the multiple levels of TAG structure (clausal, sentence, and intersentence) when designing features that characterize the TAG in terms of its fitness in serving as a justification for a given answer (see Section 5.6).

5.6 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 5.7.

5.6.1 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 5.4) – for example, summing the weight of all question or answer focus words found either within a single graphlet, or across the entire TAG.

A full list of the features and their descriptions can be found in Table 5.3. 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

⁵While conceivably we could construct TAGs of any length, currently we allow our TAGs to consist of up to three graphlets.

Feature	Description
<i>TAG-level features:</i>	
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
<i>Nugget-level features:</i>	
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
<i>Bridge features:</i>	
massMaxBridgeScore	Bridge graphlets are graphlets that contain at least one focus word from both the
massMinBridgeScore	question and answer, signifying that they are highly relevant to the question and the
massDeltaBridgeScore	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.

Table 5.3: Features used to score candidate answer justifications represented as TAGs.

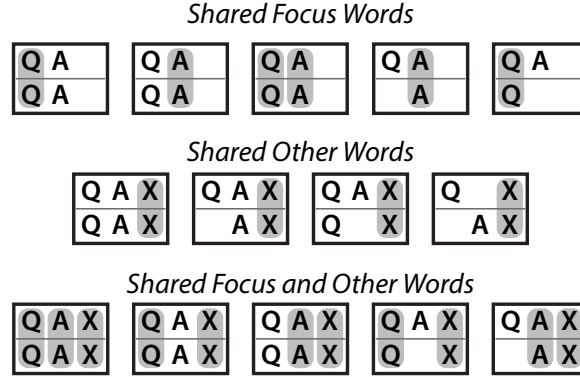


Figure 5.4: Connections between sentences are characterized based on lexical overlap between graphlets. Here, each box represents a two-sentence TAG, with graphlets stacked vertically. The presence of question or answer focus words is marked with *Q* or *A*, while the presence of other non-focus words shared between the two graphlets is marked with *X*. Lexical overlap within a category is highlighted.

(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 *bridge* the question and answer, and include a set of bridge features for evaluating this subset of highly-relevant graphlets.

5.6.2 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 Figure 5.3, in the first TAG both sentences connect on question focus words, while 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 et al., 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 modelled across connection types), whereas others are specific to certain connection types (so they should be modeled separately by type). As shown in Section 5.8, 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 leads to 14 possible connection types, depicted in Figure 5.4.⁶ Second, following Daumé’s method, we generate 15 versions for each of the features introduced in Section 5.6.1: 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 prior⁷.

⁶In practice, for a configuration that allows TAGs of 1 or 2 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.

⁷Finkel and Manning (2010) propose a joint model for domain adaptation (or adapting what is learned by a model in one domain for use in another domain) that makes use of distinct versions of the model’s feature vector for each of the domains. The multiple versions of a given feature, though, are not independent, but rather linked through a shared top-level parameter that leads them to be similar in value across all domains unless there is domain-specific evidence to the

While the features introduced in Table 5.3 apply to justifications containing any number of sentences, characterizing justifications that are longer than two sentences is not straightforward, as the number of connection types (Figure 5.4) 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. While this method would extend to justifications of arbitrary length, justifications longer than three sentences are not currently generated by our system.

5.7 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 TAGs for each of its candidate answers, and then selecting the corresponding answer candidate. This introduces an additional complication during training: while we know which answer candidate is correct, the actual quality of any given TAG is hidden. That is, many TAGs which 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 et al., 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.

contrary. This top-level parameter is the hierarchical prior.

5.7.1 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, \mathbf{A} (in the particular case of the multiple choice test, these are the four multiple choice answers). Each $a_j \in \mathbf{A}$ for a given q_i has a set of TAGs, $\mathbf{X}_{i,j}$, which connect q_i to a_j . Crucially, for a correct answer, we assume that there is *at least one* valid TAG $x_c \in \mathbf{X}_{i,j}$, which 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 .

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 $\mathbf{X}_{i,j}$ that are deemed correct, according to the current model. F averages the scores of these TAGs to compute the overall score of the answer a_j :

$$F(q_i, a_j) = \frac{\sum_{x \in P(q_i, a_j)} \Theta \cdot \Phi(x)}{|P(q_i, a_j)|} \quad (5.1)$$

where the score of a given TAG, x , is computed as the dot product between the model’s weight vector Θ and the individual TAG’s feature vector $\Phi(x)$.

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, Θ , 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, Hoffman et al. implement this function as an edge-cover⁸ 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, 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 Θ . We discuss the other extreme, i.e., using all TAGs, in Section 5.8.4 (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, Θ_{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 perceptron at

⁸ The edge-cover of a graph is a subset of the graph’s edges such that each node in the graph is connected to at least one edge in the subset. For the task of relation extraction, Hoffmann et al. (2011) construct a fully-connected graph with nodes for each supporting sentence from the corpus and nodes for each relation. Their approach to relation extraction learns which relations to extract from the supporting sentences by finding the *best* edge-cover for the graph.

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

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.

```

1: Input:
    $T$  – the number of training epochs
    $\mathcal{Q}$  – 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  $a_j$ 
2: Output:  $\Theta$  – parameter vector
3:  $\Theta = \mathbf{0}$ 
4: for  $t \leftarrow 1$  to  $T$  do
5:   for  $i \leftarrow 1$  to  $n$  do
6:      $k = \arg \max_{j=1,\dots,m} F(q_i, a_j)$ 
7:     if  $k \neq 1$  then
8:       for  $x \leftarrow P(q_i, a_1)$  do
9:          $\Theta = \Theta + \Phi(x)$ 
10:      end for
11:      for  $x \leftarrow P(q_i, a_k)$  do
12:         $\Theta = \Theta - \Phi(x)$ 
13:      end for
14:    end if
15:  end for
16: end for
17: return  $\Theta$ 

```

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

Second, we used a large-margin perceptron, similar to Surdeanu et al. (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 .

5.8 Experiments

5.8.1 Data

Questions: We assembled a corpus of 1,000 third to fifth grade standardized ele-

mentary school science exam questions, consisting of 346 publicly available questions gathered from standardized exams in 12 states, as well as 654 additional questions from an exam generating service that are not publicly available. All questions are multiple choice with four possible answer candidates. Questions vary in length from 1 to 6 sentences, while 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 3 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 Wiktionary¹⁰ 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.

5.8.2 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

¹⁰<http://simple.wiktionary.org>

each QA pair. For our dataset, the mean number of TAGs for each QA pair was 141.

Perceptron Hyperparameters: We investigated the perceptron hyperparameters using a coarse grid search¹¹ 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 epochs), and a learning rate (which dampened the updates to the weight vector) of 0.1. Since 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 by voting – each model casts a single vote for an answer candidate to each question (distributing the vote only in the case of ties), then the answer candidate with the most votes is selected.

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 dataset, 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.

5.8.3 Baselines

We include the following baselines:

Random: selects an answer randomly.

¹¹For each hyperparameter, we systematically tried values differing by an order of magnitude (rather than by small increments) to find our final settings.

Information retrieval (IR): 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 (Manning et al., 2008, Ch. 6) to rank answer candidates in a “learning to rank” (L2R) framework (see also Section 3.4). 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 Lucene¹². 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.

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

¹²<http://lucene.apache.org>

from each model (12 features total) into a ranking perceptron (Shen and Joshi, 2005; Surdeanu et al., 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 and IR model of Jansen et al. (2014), which was shown to perform well for open domain questions. Similar to the IR model, we adapted this model to our task by including six separate lexical semantics (i.e. word embedding) language models (Mikolov et al., 2013a, 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 lexical semantics models are then combined with the two features from each of the IR models (24 features total) as above using a ranking perceptron, with the top-scoring answer candidate taken as correct. Because the lexical semantics features do not easily lend themselves to constructing answer justifications, no additional human-readable justifications were provided by this model.

5.8.4 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 IR models can be combined to increase performance. We use the standard implementation for precision at 1 (P@1; Man-

		P@1		MRR
Model	P@1	Impr.	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%

Table 5.4: 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.

ning et al., 2008) and a tie-aware implementation of mean reciprocal rank (MRR; McSherry and Najork, 2008). All statistical significance tests were performed using one-tailed bootstrap resampling with 10,000 iterations (see Section 3.6, Footnote 11 for implementation).

Justification Length

Table 5.4 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 due to “inference drift” in the graph traversal process (i.e., following the connections from *breakfast* \rightarrow *hashbrowns* \rightarrow *potato* \rightarrow *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 lexical semantic graphs of Fried et al. (2015), and is represented at the level of the sentence or graphlet rather than individual lemmas, we hypothesize based on this previous work that we may be observing the beginning of 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 3 graphlets.

Extending the model to include a knowledge of the connection type (or the type of lexical overlap, see Section 5.6.2) 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.¹³

Combined Models

Where Table 5.4 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 5.5.

Baselines: Where the lexical semantics model of Jansen et al. (2014) outperformed their IR 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)

¹³Each of the 50 models in the ensemble reranker is initialized with random weights, causing the small performance difference between $1G$ and $1G_{CT}$

#	Model	P@1	P@1 Impr.	MRR	MRR Impr.
Baselines					
1	Random	25.00	–	52.08	–
2	IR	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 ^{†4}	+6.7%	63.94%	+2.3%
Combined models that include the IR baseline (Voting)					
6	IR \cup 1G _{CT} \cup 2G _{CT} \cup 3G _{CT}	43.15*	+7.3%	64.51*	+3.2%
7	IR \cup (1G _{CT} + 2G _{CT}) \cup 3G _{CT}	44.46 **	+10.6%	65.53 **	+4.9%

Table 5.5: 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 IR baseline or long (3G) TAG justifications best combined using voting ensembles (denoted with a \cup). 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.

reported that a domain-specific version of their lexical semantic model performed poorly when trained on a 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, see also Chapter 3).

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 to the static-length 2G_{CT} performance of 39.9%).

As the IR baseline and the TAG models are performing inherently different tasks

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

Table 5.6: Example justifications from the IR baseline and their associated ratings.

(information *retrieval* and information *aggregation* respectively), we are able to combine them together in a voting model in order to create a full system that contains the benefits of each. The voting models that incorporate both the IR baseline and TAG models across all justification lengths are included on lines 6 and 7. Both models significantly increase performance over the IR 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 IR 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.

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 (IR) and the best performing TAG model that is independent of IR (i.e., $1G_{CT} + 2G_{CT}$). Where TAG

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)

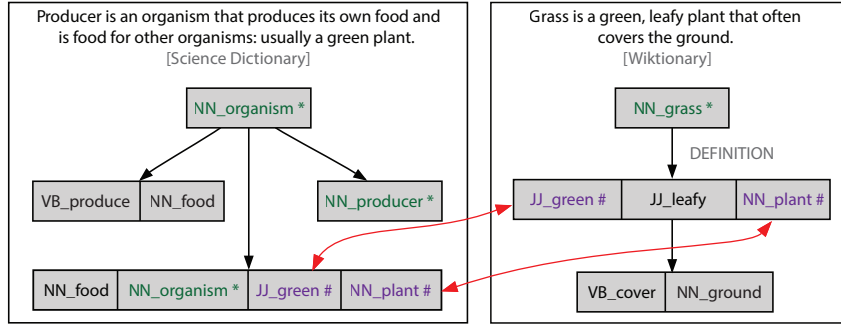


Table 5.7: An example TAG and justification rated as *good*. The two sentences connect on non-focus "other" shared words (e.g., *green*, *plant*) which are not found in the question or answer, but which are highly related to the focus words.

justifications took the form of the sentences being aggregated, justifications for the IR model were taken to be the highest-scoring short passages from each of the six knowledge bases. As the IR model was tuned to a retrieval size of two sentences to maximize P@1 performance, each IR justification was two sentences in length.¹⁴ To facilitate easy comparison, since the IR model provides six justifications (one from each knowledge base), we evaluate the top six scoring TAG justifications.

Two of the authors independently rated the quality of the justifications. 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 5.6 for IR examples, and Figure 5.7 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 5.6. The two final ratings are for justifications that

¹⁴Documents 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.

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

Table 5.8: *At least one* justification performance for both IR and TAG models, reflecting the highest rating attained by at least one of the top six justifications for a given question.

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 justification performance is listed in Table 5.8. For the TAG model, 56.7% of the questions had at least one justification rated as *good*, outperforming IR justifications by 11.4% (absolute).

This experiment supports our original intuition that justifications must be aggregated from multiple resources. While the small window of sentences from the IR 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 IR 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 IR model.

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 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.

5.9 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 multi-sentence 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, i.e., a table for the color of objects contains two rows, one for the object of interest (e.g., “leaf”), and another for it’s 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 2–5 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 modelled as an integer linear programming (ILP) problem which finds paths that maximize QA performance.

The TableILP system reported by Khashabi et al. (2016) contains 69 tables

Resource	Sentences	IR	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%

Table 5.9: Most useful knowledge resources for justifications classified as ”good”.

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 4 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@1¹⁵, compared to the 44.6% P@1 from the best-performing TAG model in Table 5.5. The performance difference between these systems is likely not statistically significant.¹⁶

We view these systems as complementary, converging, and with each capable of exploring different aspects of graph-based inference for science QA. While 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 5.4 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 allow us to investigate the graph-building process in isolation, reducing inference drift due to noise, and further moving this area forward.

Which knowledge resources are generating the most useful answer justifications? Shown in Table 5.9, the Barron’s Study Guide (SG) contributes more of the *good* justification sentences than any other source, followed by the science dictionary, then the other resources. Interestingly, the Simple Wiktionary contributes

¹⁵We wish to thank Khashabi et al. (2016) for providing us with this performance figure.

¹⁶We did not have access to system output. However, in our experiments on this dataset, we observed that only differences in P@1 scores of 3% or higher (absolute) tend to be statistically significant at $p < 0.05$.

the fewest sentences to the *good* justifications for the IR system (7.5%), but for the TAG system it is the third largest contributor (14.8%). That is, while the IR 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, while 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, given our data and resources.

How orthogonal is the performance of the TAG model when compared to IR? Both the TAG and IR 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 5.5 appear to support this, where combining the TAG and IR 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.

Comparing the TAG ($1G_{CT} + 2G_{CT}$) and IR 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 while the TAG and IR models share about half of their performance, each model is sensitive to different kinds of questions, suggesting that further combination strategies between TAG and IR are worth exploring.

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

Table 5.10: Proportion of good justifications with a given number of connecting word categories (Q, A, X) for both correct and incorrect answers. (Top 6)

5.10 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 5.6.2), 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.¹⁷

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, while we

¹⁷Our current focus word extractor is not yet well suited to complex multisentence questions. While the question corpus as a whole consists of 62% single-sentence questions, 25% two-sentence questions, and 14% questions which 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.

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

Table 5.11: 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.

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.

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 5.10 shows that for questions answered correctly, in 69% of cases good justifications are connected on 2 or 3 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 while 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 detecting well connected answer justifications, which correlate with good answers, but the features aren’t 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. (2013a), 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

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%

Table 5.12: 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.

the error analysis based on the inference type required to answer them correctly. This analysis, shown in Table 5.11, 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 6, 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.

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 <i>twist</i> , but some example words (<i>bruce</i> , <i>open</i> and <i>top</i>) are not suppressed, and receive the same weight as <i>twist</i> .

Table 5.13: Example of failure to extract appropriate focus words from the question.

5.10.1 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 knowledge bases, and other errors. The distribution of these errors is shown in Table 5.12 and detailed below.

FOCUS WORD ERRORS: Extracting focus words from questions and answers can be difficult, as often these questions are grounded in complex examples. Moreover, while 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 5.13.

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.

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.

NOISE IN THE KNOWLEDGE BASE: We included nearly all sentences contained in each of the five science-domain corpora. Though uncommon, 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.

COMPLEX INFERENCE: Despite the elementary-level of our question set, some questions require complex inference to be answered correctly (a recent analysis of a very similar elementary science question set indicated that as many as 77% of questions required a form of complex inference to solve (Jansen et al., 2016)). 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. While 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 5.14.

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 <i>bears</i> eat <i>fish</i> , <i>eating</i> an animal makes you a <i>predator</i> and it your <i>prey</i> , and a decrease in <i>prey</i> population also causes a decrease in <i>predator</i> 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 5.14: Example of a question that needs more than two sentences to answer.

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: <i>evaporation, condensation, precipitation</i> . The focus word extractor does not currently detect causal or process markers, and as such the focus words for the question do not include <i>after</i> .
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)

Table 5.15: Example of a question that requires reasoning over a causal structure or process.

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 <i>large</i> supply causes <i>less</i> competition. The focus word extractor does not currently detect quantifiers, and as such the focus words for the correct answer do not include <i>large</i> .
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 5.16: Example of a question that requires an understanding of the quantifiers in both the question and the answers.

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)

Table 5.17: Example of a question that requires an understanding of negation.

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 5.15, 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 5.16, our current system does not implement a knowledge of quantifiers, nor their relationship to each other (e.g., *some* is less than *most*, *smaller* is less than *big*, etc.).

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 5.17, these sentences can be misused by our system.

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 <i>condensing</i> in the question with <i>condensation</i> 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)

Table 5.18: Example of a failure to recognize relatedness or equivalence of words.

MISCELLANEOUS:

Semantic Cluster Matching: While 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 5.18), 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.

Other: While 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. Due to 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.

5.10.2 Summary of Errors

To summarize, this error analysis suggests that a majority of errors are not caused by fundamental limitations of this approach to question answering for science exams, but rather by surface issues such as focus-word extraction errors, noise in the knowledge base, knowledge coverage, and semantic cluster matching. Were we to solve these surface issues, 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 implemented dedicated domain-specific methods for the difficult problems left unsolved by our approach.

5.11 Conclusion

We have proposed an approach for QA that emphasizes model interpretability, such that producing human-readable justifications for answers, and evaluating answer justification quality, is the critical component¹⁸. 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, to maintain robustness, 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 explain quality, 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 complete, human-readable justifications for those answers. We further validated our three critical contributions, showing that: (a) modeling the high-level task of determining

¹⁸To increase reproducibility, all the code behind this effort is released as open-source software¹⁹, which allows other researchers to use our entire science QA system as is, or to explore adapting the various components to other tasks.

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.

Here we focus on elementary science exams, and even at this level the problem is not solved. We determined that questions with complex inference still present difficulty for our system, and presumably this difficulty would only increase were we to look at secondary or post-secondary questions. In order for our approach to perform well in those settings, we hypothesize that in addition to level-appropriate background knowledge, we would also need to be able to aggregate more information to perform more complex inference and abstract reasoning. However, as more information is aggregated, the amount of noise relative to the signal quickly becomes problematic. Thus, in order to realistically use deeper aggregation, noise handling would likely need to be addressed directly, potentially through compartmentalizing and having specialized methods for different inference types. While we do not address adult-level questions here, in Chapter 6, we do evaluate a related approach on a more difficult set of questions from 8th grade science exams.

This approach seeks to balance robustness with interpretability by utilizing our lightly-structured graphlets for creating answer justifications. While here we demonstrate the success of this approach in this domain, performing this structured sentence decomposition and aggregation over larger corpora (such as those required for more advanced exams) can become quite cumbersome. For this reason, in Chapter 6, where we move from elementary to 8th grade level questions, we propose another model which is designed to be even more robust without losing interpretability. That is, we explore a similar model which continues to rank answer justifications (maintaining interpretability), but which does not perform aggregation and which makes use of only the shallower free-text representations of the knowledge base sentences and accordingly uses a shallower set of explicit feature (rather than those based on focus words and graphlet structure, increasing robustness).

CHAPTER 6

Robust and interpretable: A neural approach

Once again, we are concerned with developing interpretable models for question answering. In Chapter 5 we proposed a method for question answering that created and ranked human-readable answer justifications using performance on the QA task as the only form of supervision. We demonstrated that not only did it outperform a strong information retrieval baseline in terms of correctly answering questions, but it also produced a higher percentage of good justifications, as rated by human annotators. The main drawback to the system was simply in the somewhat heavy cost of the sentence processing. While this processing was able to be done offline (and so only once), the time and size burden still prevents its practical use over extremely large corpora.

Here, we propose an approach that once again uses answer ranking as distant supervision for learning how to select informative justifications. As with the previous approach, our justifications serve as inferential connections between the question and the correct answer while often containing little lexical overlap with either. However, here in this modified approach we use a shallower representation of knowledge base sentences and we do not perform aggregation, facilitating usage over much larger resources. We additionally extend the linear learning framework used in Chapter 5 through the use of deep learning (specifically, a non-linear feed-forward neural network), as described in Section 6.3. In making these modifications to increase the robustness of the approach, we purposefully maintain our interpretability – we continue to rank and return human-readable answer justifications for our model’s chosen answers.

6.1 Chapter Outline

The remainder of the chapter is organized as follows. First, we review related work in Section 6.2. In Section 6.3 we describe our overall neural approach for QA that reranks answer justifications as an intermediate (and human-interpretable) step in answer selection. In Section 6.4 we describe our neural network in more detail. In that same section, we also describe our set of features designed to combine both learned representations (i.e. embeddings) and explicit features to capture the connection between questions, answers, and answer justifications. Then, in Section 6.5 we provide the details of our experimental setup and in Section 6.6 we provide our results – with this end-to-end approach we are able to significantly improve upon a strong IR baseline in both justification ranking (+9% rated highly relevant) and answer selection (+6% P@1). In Section 6.6.3 we utilize our model-returned justifications in an error analysis. Finally, in Section 6.7 are our conclusions.

6.2 Related Work

In many ways, deep learning has become the canonical example of the "black box" of machine learning and many of the approaches to explaining it can be loosely categorized into two types: approaches that try to interpret the parameters themselves (e.g., with visualizations and heat maps (Zeiler and Fergus, 2014; Hermann et al., 2015; Li et al., 2016)), and approaches that generate human-interpretable information that is ideally correlated with what is being learned inside the model (e.g., Lei et al. (2016)). Our approach falls into the latter type – we use our model’s reranking of human-readable justifications to give us insight into what the model considers informative for answering questions. This allows us to see where we do well (Section 6.6.2), and where we can improve (Section 6.6.3).

Deep learning has been successfully applied to many recent QA approaches and related tasks (Bordes et al., 2015; Hermann et al., 2015; He and Golub, 2016; Dong et al., 2015; Tan et al., 2016, inter alia). However, large quantities of data are needed to train the millions of parameters often contained in these models. Recently,

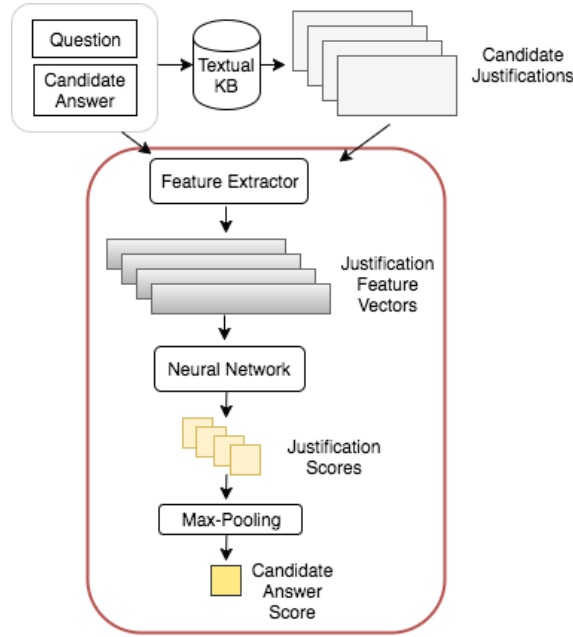


Figure 6.1: Architecture of our question answering approach. Given a question, candidate answer, and a free-text knowledge base as inputs, we generate a pool of candidate justifications, from which we extract feature vectors. We use a neural network to score each and then retain *only* the highest-scoring justification from the pool of candidates (i.e., max-pooling), thus selecting the current best justification. This justification score serves as the score for the candidate answer itself. The red border indicates the components that are trained online.

simpler model architectures have been proposed that greatly reduce the number of parameters while maintaining high performance (e.g., Iyyer et al., 2015b; Chen et al., 2016; Parikh et al., 2016). We take inspiration from this trend and propose a simple neural architecture for our task to offset the limited available training data.

Another way to mitigate sparse training data is to include higher-level explicit features. Like Sachan et al. (2016), we make use of explicit features alongside features from distributed representations (see Section 6.4.2) to capture connections between questions, answers, and supporting text. However, we use a simpler set of features and while they use structured and semi-structured knowledge bases, we use only free-text.

6.3 Approach

One of the primary difficulties with the explainable QA task addressed here is that, while we have supervision for the correct answer, we do not have annotated answer justifications. Here we tackle this challenge by using the QA task performance as supervision for the justification reranking, allowing us to learn to choose both the correct answer and a compelling, human-readable justification for that answer.

Additionally, similar to the strategy Chen and Manning (2014b) applied to parsing, we combine representation-based features with explicit features that capture additional information that is difficult to model through embeddings, especially with limited training data.

The architecture of our approach is summarized in Figure 6.1. Given a question and a candidate answer, we first query an textual knowledge base (KB) to retrieve a pool of potential justifications for that answer candidate. For each justification, we extract a set of features designed to model the relations between questions, answers, and answer justifications based on word embeddings, lexical overlap with the question and answer candidate, discourse, and information retrieval (IR) (Section 6.4.2). These features are passed into a simple neural network to generate a score for each justification, given the current state of the model. A final max-pooling layer filters the input, retaining only the top-scoring justification for the candidate answer and this max score is used also as the score for the answer candidate. In relation to the latent variable perceptron proposed in Section 5.7.1, this max-pooling layer is functionally identical to the latent P function that learns to identify the good justifications for a given answer (recall that we implemented P such that it selects exactly one justification at each iteration – the best-performing, given the current state of the model). The final candidate answer score (also shown in Figure 6.2) essentially takes the place of $F(q_i, a_j)$ in Equation 5.1. The system is trained using correct-incorrect answer pairs with a pairwise margin ranking loss objective function to enforce that the correct answer be ranked higher than any of the incorrect answers.

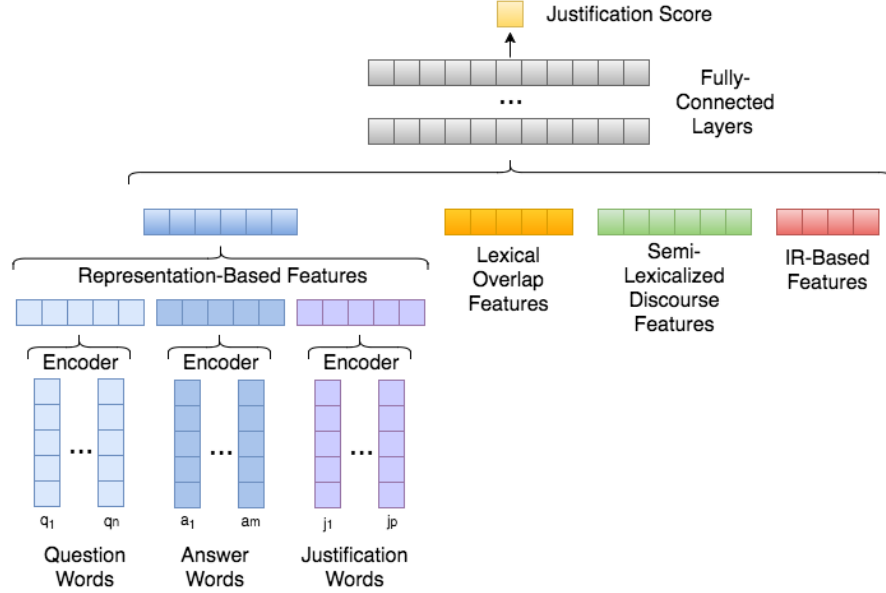


Figure 6.2: Detailed architecture of the model’s scoring component. The question, candidate answer, and justification are encoded (by summing their word embeddings) to create vector representations of each. These representations are combined in several ways to create a set of representation-based similarity features that are concatenated to additional explicit features capturing lexical overlap, discourse and IR information and fed into a feed-forward neural network. The output layer of the network is a single node that represents the score of the justification candidate.

With this end-to-end approach, the model learns to select justifications that allow it to correctly answer questions. We hypothesize that, as with the system in Chapter 5, this approach enables the model to indirectly learn to choose justifications that provide good explanations as to why the answer is correct. We empirically test this hypothesis in Section 6.6, where we show that indeed the model learns to correctly answer questions, as well as to select justifications that fully explain the connection between those answers to the corresponding questions.

6.4 Model and Features

Our approach consists of three main components: (a) the retrieval of a pool of candidate answer justifications (Section 6.4.1); (b) the extraction of features for

each (Section 6.4.2); and (c) the scoring of the answer candidate itself based on this pool of justifications (Section 6.4.3). The architecture of this latter scoring component is shown in Figure 6.2.

<hr/> Representation-Based Features (Emb) <hr/>	
$sim(Q, A)$, $sim(Q, J)$, and $sim(Q, J)$	The pairwise cosine similarities between the question, answer, and justification representations.
$sim(Q, uniqueJ)$ and $sim(A, uniqueJ)$	The cosine similarities between each of the question and answer representations and the representation for the terms in the justification which don't overlap with either.
$dist(Q + J, A)$	The euclidean distance between the sum of the question and justification representations and that of the answer (see Section 6.4.2 for motivation and details).
<hr/> Lexical Overlap Features (LO) <hr/>	
$qCoverage$, $aCoverage$, and $qaCoverage$	The proportion of question words, of answer words, and of the combined set of question and answer words that also appear in the justification.
$jNovelty$	The proportion of justification words that do not appear in either the question or the answer.
$numWords$	Length of the justification in words. ¹
<hr/> Semi-Lexicalized Discourse Features (lexDisc) <hr/>	
$lexDisc_0, \dots, lexDisc_n$	A set of indicator-like features that capture the presence of semi-lexicalized discourse relations (see Section 6.4.2 for details and example).
<hr/> IR-Based Features (IR^{++}) <hr/>	
IR_{max} , IR_{sum}	The reciprocal rank of the answer candidate based on (a) the highest scoring IR-retrieved document and (b) the weighted sum of all the IR-retrieved documents, using the basic IR query (see 6.4.2 for details).
$IR_{max}^{boosted}$, $IR_{sum}^{boosted}$	The reciprocal rank of the answer candidate based on (a) the highest scoring IR-retrieved document and (b) the weighted sum of all the IR-retrieved documents, using the boosted IR query (see 6.4.2 for details).

Table 6.1: Summary of the features calculated for each candidate justification.

¹We normalized this value by the maximum justification length.

6.4.1 Candidate Justification Retrieval

The first step in our process is to use standard information retrieval (IR) methods to retrieve a set of candidate justifications for each candidate answer to a given question. To do this, we build a bag-of-words query using the content lemmas for the question and answer candidate, boosting the answer lemmas to have four times more weight². We used Lucene³ with a *tf-idf* based scoring function to return the top-scoring documents from the knowledge base. Each of these indexed documents consists of a single sentence from our corpora, and serves as one potential justification.

6.4.2 Feature Extraction

For each retrieved candidate justification, we extract a set of features based on (a) distributed representations (i.e., word embeddings) of the question, candidate answer, and justification terms; (b) strict lexical overlap; (c) discourse relations present in the justification; and (d) the IR scores for the justification. Each of these features is described in detail below, and a summary is provided in Table 6.1.

Representation-based features (Emb): To model the similarity between the text of each question (Q), candidate answer (A), and candidate justification (J), i.e. the answer passage from the IR model, we include a set of features that utilize distributed representations of the words found in each. First we encode each by summing the word embedding vectors for each of their words.⁴ We then compute $\text{sim}(Q, A)$, $\text{sim}(Q, J)$, and $\text{sim}(A, J)$ using cosine similarity. Using another vector representation of only the *unique* words in the justification, i.e., the words that do not occur in either the question or the candidate answer, we also compute $\text{sim}(Q, \text{unique}J)$ and $\text{sim}(A, \text{unique}J)$.

²We empirically found this answer term boosting to ensure retrieval of documents which were relevant to the particular answer candidate.

³<https://lucene.apache.org>

⁴While this bag-of-words approach is not ideal in many ways, it performed equivalently to far more complicated approaches such as LSTMs and GRUs, also noted by (Iyyer et al., 2015b), likely due to the limited training data in this domain.

To create a feature which captures the relationship between the question, answer, and justification, we take inspiration from TransE, a popular relation extraction framework (Bordes et al., 2013). TransE is based on the premise that if two entities, e_1 and e_2 are related by a relation r , then a mapping into k dimensions, $m(x) \in \mathbb{R}^k$ can be learned such that $m(e_1) + m(r) \approx m(e_2)$. Here, we modify this intuition for QA by suggesting that given the vectorized representations of the question, answer candidate, and justification above, $Q + J \approx A$, i.e., a question combined with a strong justification will point towards an answer. Here we model this as an explicit feature, the euclidean distance between $Q + J$ and A , and hypothesize that as a consequence the model will learn to select passages that maximize the quality of the justifications. This makes a total of six features based on distributed representations.

Lexical overlap features (LO): We additionally characterize each justification in terms of a simple set of explicit features designed to capture the size of the justification, as well as the lexical overlap (and difference) between the justification and the question and answer candidate. We include these five features: the proportion of question words, of answer words, and of the combined set of question and answer words that also appear in the justification; the proportion of justification words that do not appear in either the question or the answer; and the length of the justification in words.⁵

Semi-Lexicalized Discourse features (lexDisc): These features use the discourse structure of the justification text, which has been shown to be useful for QA (Jansen et al., 2014; Sharp et al., 2015; Sachan et al., 2016, see also Chapter 3).

We use the discourse parser of Surdeanu et al. (2015) to first fragment the text into elementary discourse units (EDUs) and then recursively connects neighboring EDUs together, assigning each a binary discourse relation label such as *Elaboration* or *Contrast*. Each of these discourse relations consists of a head, a relation label, and a modifier. For each of the 18 possible relation labels (listed in Table 3.1 in Section 3.3), we create a set of semi-lexicalized discourse features that indicate the presence

⁵We normalized this value by the maximum justification length.

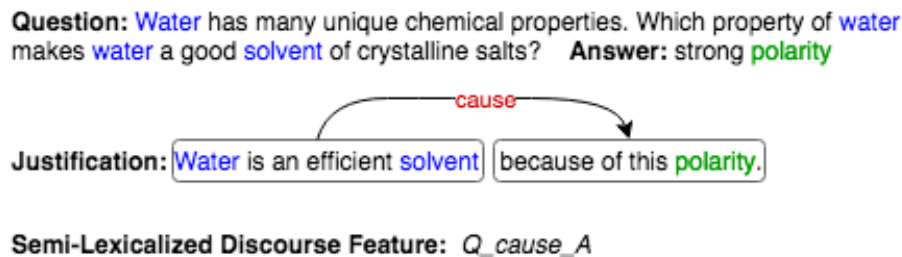


Figure 6.3: Example showing the discourse feature extracted from a question, answer, and justification. Words occurring in the question are shown in blue and words occurring in the answer are shown in green. The boxes around the justification text indicate the elementary discourse units identified by the discourse parser, and the arrow indicates the found discourse relation (with the assigned relation label given in red).

of a given discourse relation as well as whether or not the head and modifier texts contain words from the question and/or the answer.

Consider, for example, the question Q : *What makes water a good solvent...?* A : *strong polarity* shown in Figure 6.3. The discourse parser identified a causal relation in the justification: $[Water\ is\ an\ efficient\ solvent]_{e1}\ [because\ of\ this\ polarity.]_{e2}$. From this discourse relation, we create the semi-lexicalized feature Q_cause_A , because there is a *Cause* relation between EDUs $e1$ and $e2$, $e1$ overlaps lexically with the question (i.e. *water* and *solvent*, and $e2$ overlaps lexically with the answer (i.e., *polarity*). Since there are 18 possible discourse relation labels, and the prefix and suffix can be any of Q , A , QA or *None*, this creates a set of 288 indicator features⁶, though in practice many combinations are never found.

IR-based features (IR^{++}): Finally, we also use a set of four IR-based features which are assigned at the level of the answer candidate (i.e., these features are identical for each of the candidate justifications for that answer choice). Using the same IR query method as described in Section 6.4.1, we query the indexed corpus using question and answer terms (boosting the importance of the answer terms), to

⁶In practice these features can be greater than 1.0. We wanted to maintain the indicator-like quality of the features while still capturing if a given justification contains several of a certain type of discourse feature. Thus, we use the fourth-root of the count of each of the discourse features present in the justification as the feature value.

retrieve a set of indexed documents for each question and answer candidate. Using the *tf-idf* based retrieval scores of these returned documents, $s(d_i)$ for $d_i \in D$, we rank the answer candidates using two methods:

- by the maximum retrieved document score for each candidate, and
- by the weighted sum of all retrieved document scores⁷:

$$\sum_{d_i \in D} \frac{1}{i} s(d_i) \quad (6.1)$$

We repeat this process using an unboosted query as well, for a total of four rankings of the answer candidates. We then use these rankings to make a set of four reciprocal rank features, IR_0^{++} , ..., IR_3^{++} , for each answer candidate (i.e., $IR_0^{++} = 1.0$ for the top-ranked candidate in the first ranking, $IR_0^{++} = 0.5$ for the next candidate, etc.)

6.4.3 Neural Network

As shown in Figure 6.2, the extracted features for each candidate justification are concatenated and passed into a fully-connected feed-forward neural network. The output layer is a single node representing the justification score. We then use max-pooling over these scores to select the current best justification for the answer candidate, and use its score as the score for the answer candidate itself. For training, the correct answer for a given question is paired with each of the incorrect answers, and each are scored as above. We compute the pair-wise margin ranking loss for each training pair:

$$L = \max(0, m - F(a^+) + F(a^-)) \quad (6.2)$$

where $F(a^+)$ and $F(a^-)$ are the model scores for a correct and incorrect answer candidate and m is the margin, and backpropagate the gradients. At testing time,

⁷Weighted sum was based on the IR scores used in the winning Kaggle system from user Cardal (https://github.com/Cardal/Kaggle_AllenAIScience)

we use the trained model to score each answer choice (again using the maximum justification score) and select the highest-scoring.

As we are interested in not only correctly answering questions, but also selecting valid justification for those answers, we keep track of the scores of *all* justifications and use this information to return the top k justifications for each answer choice. These are evaluated along with the answer selection performance in Section 6.6.

6.5 Experiments

6.5.1 Data and Setup

We evaluated our model on the set of 8th grade science questions that was provided by the Allen Institute for Artificial Intelligence (AI2) for a recent Kaggle challenge. The training set contained 2,500 questions, each with 4 answer candidates. For our test set, we used the 800 publicly-released questions that were used as the validation set in the actual evaluation.⁸ We tuned our model architectures and hyper-parameters on the training data using five-fold cross-validation (training on 4 folds, validating on 1). During testing, we froze the model architecture and all hyperparameters and re-trained on all the training data, setting aside a random 15% of training questions to facilitate early stopping. We implemented early stopping with a patience parameter initially set to 5. For a given training epoch, if the model performed worse on the held-out question set (the development fold during training and the random 15% of training questions during testing) than it did the epoch prior, then the patience was decreased by 1. Otherwise, if the model performed better on the held-out set, the patience was increased by 1 (capped at 5). If the model’s patience reached 0 before the maximum number of epochs, the training was stopped. At the end of training, we used the version of the model that performed best on the held-out set.

⁸The official testing dataset is not publicly available.

6.5.2 Baselines

In addition to previous work, we compare our model against two strong IR baselines:

- **IR Baseline:** For this baseline, we rank answer candidates by the maximum *tf.idf* document retrieval score using an unboosted query of question and answer terms (see Section 6.4.1 for retrieval details).
- **IR⁺⁺:** This baseline uses the same architecture as the full model, as described in Section 6.4.3, but with only the IR⁺⁺ feature group.

6.5.3 Corpora

For our pool of candidate justifications (as well as the scores for our IR baselines) we used the corpora that were cited as being most helpful to the top-performing systems of the Kaggle challenge. These consisted of short, flash-card style texts gathered from two online resources: about 700K sentences from StudyStack⁹ and 25K sentences from Quizlet¹⁰. From these corpora, we use the top 50 sentences retrieved by the IR model as our set of candidate justifications. All of our corpora were annotated using the Stanford CoreNLP toolkit (Manning et al., 2014b), the dependency parser of Chen and Manning (2014b), and the discourse parser of Surdeanu et al. (2015).

While our model is able to learn a set of embeddings, we found performance was improved when using pre-trained embeddings, and in this low-data domain, fixing these embeddings to not update during training substantially reduced the amount of model over-fitting. In order to pre-train domain-relevant embeddings for our vocabulary, we used the documents from the StudyStack and Quizlet corpora, supplemented by the newly released Aristo MINI corpus (December 2016 release)¹¹, which contains 1.2M science-related sentences from various web sources. The training was done using the `word2vec` algorithm (Mikolov et al., 2010, 2013a) as implemented

⁹<https://www.studystack.com/>

¹⁰<https://quizlet.com/>

¹¹<http://allenai.org/>

#	Model	P@1 Val	P@1 Test
1	Random	25	25
2	IR Baseline	47.2	47
3	IR ⁺⁺	50.7**	36.35
4	Iyyer et al. (2015b)	—	32.52
5	Khot et al. (2017)	—	46.17
6	Our approach w/o IR	50.54*	48.66
7	Our approach	54.0**††	53.3**†

Table 6.2: Performance on the AI2 Kaggle questions, measured by precision-at-one (P@1). *s indicate that the difference between the corresponding model and the IR baseline is statistically significant (* indicates $p < 0.05$ and ** indicates $p < 0.001$) and †s indicate significance compared to IR⁺⁺. All significance values were determined through a one-tailed bootstrap resampling test with 100,000 iterations.

by Levy and Goldberg (2014), such that the context for each word in a sentence is composed of all the other words in the same sentence. We used embeddings of size 50 as we did not see a performance improvement with higher dimensionality.

6.5.4 Model Tuning

The neural model was implemented in Keras (Chollet, 2015) using the Theano (Theano Development Team, 2016) backend. For our feed-forward component, we use a shallow neural network that we lightly tuned to have a single fully-connected layer containing 10 nodes, glorot uniform initialization, a *tanh* activation, and an L2-regularization of 0.1. We trained with the RMSProp optimizer (Tieleman and Hinton, 2012), a learning rate of 0.001, 100 epochs, a batch size of 32, and early stopping with a patience of 5 epochs. Our loss function used a margin of 1.0.

We experimented with burn-in, i.e., using the best justification chosen by the IR model for the first mini-batches, but found that models without burn-in performed better, indicating that the model benefited from being able to select its own justification.

Ablated Model	P@1 Val
IR ⁺⁺ + LO	53.4 ^{**††}
IR ⁺⁺ + LO + lexDisc	53.6 ^{**††}
Full Model (IR ⁺⁺ + LO + lexDisc + Emb)	54.0^{**††}

Table 6.3: Ablation of feature groups results, measured by precision-at-one (P@1) on validation data. Emb indicates our embedding-based features, LO indicates our lexical overlap features, lexDisc signifies our semi-lexicalized discourse features, and IR⁺⁺ indicates our information retrieval based features. Significance is indicated as in Table 6.2.

6.6 Results

Rather than seeking to outperform all other systems at selecting the correct answer to a question, here we aimed to construct a system that can produce substantially better justifications for why the answer choice is correct to a human user (i.e., that is interpretable), without unduly sacrificing accuracy on the answer selection task (i.e., that maintains robustness). Accordingly, we evaluate our system both in terms of it’s ability to correctly answer questions (Section 6.6.1), as well as provide high-quality justifications for those answers (6.6.2). Additionally, we perform an error analysis (Section 6.6.3), taking advantage of the insight the reranked justifications provide into what the model is learning.

6.6.1 QA Performance

We evaluated the accuracy of our system as well as the baselines on the held-out 800 set of test questions. Performance, measured in precision at 1 (P@1; Manning et al., 2008), is shown in Table 6.2 for both the validation (i.e., cross validation on training) and test partitions. Because NNs are sensitive to initialization, each experimental result shown is the average performance across five runs, each using different random seeds.

The best performing baseline on the validation data was a model using only IR⁺⁺ features (line 3), but its performance dropped substantially when evaluated on test due to the failure of several random seed initializations to learn. For this

reason, we assessed significance of our model combinations with respect to both the IR baseline as well as the IR^{++} (indicated by * and †s, respectively). All significance values were determined through a one-tailed bootstrap resampling test with 100,000 iterations (see Section 3.6, Footnote 11 for implementation).

Our full model that combines IR^{++} , lexical overlap, discourse, and embeddings-based features, has a P@1 of 53.3% (line 7), an absolute gain of 6.3% over the strong IR baseline despite using the same background knowledge.

Comparison to Previous Work: We compared our performance against another model that achieves state of the art performance on a different set of 8th grade science questions, TUPLEINF(T+T') (Khot et al., 2017). TUPLEINF(T+T') uses Integer Linear Programming to find support for questions via tuple representations of KB sentences¹². On our test data, TUPLEINF(T+T') achieves 46.17% P@1 (line 5). As this model is independent of an IR component, we compare its performance against our full system without the IR-based features (line 6), whose performance is 48.66% P@1, an absolute improvement of 2.49% P@1 (5.4% relative) despite our unstructured text inputs and the far smaller size of our knowledge base (three orders of magnitude).

Sachan et al. (2016) also tackle the AI2 Kaggle question set with an approach that learns alignments between questions and structured and semi-structured KB data. They use only the training questions (splitting them into training, validation, and testing partitions), supplemented by questions found in online study guides, and report an accuracy of 47.84%. By way of a loose comparison (since we are evaluating on different data partitions), our model has approximately 5% higher performance despite our simpler set of features and unstructured KB.

We also compare our model to our implementation of the basic Deep-Averaged Network (DAN) Architecture of Iyyer et al. (2015b). We used the same 50-dimensional embeddings in both models, so with the reduced embedding dimension,

¹²Notably, one portion of the tuple KB used was constructed based on a different 8th grade question set than the one we use here.

we reduced the size of each of the DAN dense layer to 50 as well. For simplicity, we also did not implement their word-dropout, a feature that they reported as providing a performance boost. Using this implementation, the performance on the test set was 31.50% P@1. To help with observed overfitting, we tried removing the dense layers and received a small boost to 32.52% P@1 (line 4). The lower performance of their model, which relies exclusively on latent representations of the data, underscores the benefit of including explicit features alongside latent features in a deep-learning approach for this domain.

In comparison to other systems that competed in the Kaggle challenge, our system comes in in 7th place out of 170 competitors (top 4%).¹³ Compared with the systems which disclosed their methods, we use a subset of their corpora and substantially less hyperparameter tuning, and yet we achieve competitive results.

Feature Contribution: To evaluate the contribution of the individual feature groups, we additionally performed an ablation experiment by removing one feature group at a time and then re-evaluating. The results of this experiment are shown in Table 6.3. Each of our ablated models performed significantly better than the IR baseline on the validation set, including our simplest model that includes only the lexical overlap and IR-based feature, IR⁺⁺+LO¹⁴.

6.6.2 Justification Performance

One of our key claims is that our approach addresses the related, but more challenging problem of performing *explainable* question answering, i.e., providing a high-quality, compelling justification for the chosen answer. To evaluate this claim, we evaluated a random set of 100 test questions that both the IR baseline and our full system answered correctly. For each question, the quality of each of the top five justi-

¹³Based on the public leaderboard (<https://www.kaggle.com/c/the-allen-ai-science-challenge/leaderboard>). The best scoring submission had an accuracy of 59.38%. Note that for the systems that participated, this set served as *validation* while for us it was test, and thus it is likely that these scores are slightly overfitted to this dataset, but for us it was blind. As such this is a conservative comparison, and in reality the difference is likely to be smaller.

¹⁴As we consider the fully ablated IR⁺⁺ to be a baseline, we do not include it here.

Question	
Q: Scientists use ice cores to help predict the impact of future atmospheric changes on climate. Which property of ice cores do these scientists use?	
A: The composition of ancient materials trapped in air bubbles	
Rating	Example Justification
<i>Good</i>	Ice cores: cylinders of ice that scientist use to study trapped atmospheric gases and particles frozen with in the ice in air bubbles
<i>Half</i>	Ice core: sample from the accumulation of snow and ice over many years that have recrystallized and have trapped air bubbles from previous time periods
<i>Topical</i>	Vesicular texture formation [has] trapped air bubbles.
<i>Off-topic</i>	Physical change: change during which some properties of material change but ...

Table 6.4: Example justifications from the our model and their associated ratings.

fications was assessed by one of the authors¹⁵. For IR, these were the highest-scoring retrieved documents, and for our system, these were the top-scoring justifications as re-ranked by our model. Each of these justifications was composed of a single sentence from our corpus, though a future version could use multi-sentence passages, or aggregate several sentences together, as in Chapter 5 (Jansen et al., 2017).

As in the justification evaluation in Chapter 5, each justification was rated as either *Good* (if the connection between the question and correct answer was fully covered), *Half* (if there was a missing link), *Topical* (if the justification was simply of the right topic), or *Off-Topic* (if the justification was completely unrelated to the question). Examples of each rating are provided in Table 6.4.

Results of this analysis are shown using three evaluation metrics in Table 6.5. The first two columns show the percentage of questions which had a *Good* justification at position 1 (Good@1), and within the top 5 (Good@5). Note that 61% of the top-ranked justifications from our system were rated as *Good* as compared to 52% from the IR baseline (a gain of 9%), despite the systems using identical corpora.

¹⁵While annotation by only one author is non-ideal, all justifications assessed and the scores they received are available upon request.

Model	Good@1	Good@5	NDCG@5
IR Baseline	0.52	0.64	0.55
Our Approach	0.61	0.74	0.62**

Table 6.5: Percentage of questions that have at least one *good* justification within the top 1 (Good@1) and the top 5 (Good@5) justifications, as well as the normalized discounted cumulative gain at 5 (NDCG@5) of the ranked justifications. Significance indicated as in Table 6.2.

We also evaluated the justification ratings using normalized discounted cumulative gain at 5 (NDCG@5) as formulated in Manning et al. (2008), p.163. In our context, let Q be our set of questions and $R(q, j)$ be the relevance score (gain) given to justification j for question q , then the normalized discounted cumulative gain at k is:

$$NDCG(Q, k) = \frac{1}{|Q|} \sum_{q=1}^{|Q|} Z_{kq} \sum_{m=1}^k \frac{2^{R_{(q,m)}} - 1}{\log_2(1 + m)} \quad (6.3)$$

where Z_{kq} is the normalization factor, calculated so that a perfect justification ranking (for us, this would be all *Good* justifications) for question q would have an NDCG@ k of 1. k is the number of results to consider (here we use the top 5).

In terms of gains, we assigned *Good* justifications a gain of 3.0, *Half* a gain of 2.0, *Topical* a gain of 1.0, and *Off-Topic* a gain of 0.0. With this formulation, our system had a NDCG@5 of 0.62 while the IR baseline had a significantly lower NDCG@5 of 0.55 ($p < 0.001$), shown in the third column of Table 6.5.

Contribution of Learning to Rerank Justifications: The main assertion of this work is that through learning to rank answers and justifications for those answer candidates in an end-to-end manner, we both answer questions correctly and provide justifications as to why the answer is correct. To confirm that this is the case, we also ran a version of our system that does not re-rank justifications, but uses the top-ranked justification retrieved by IR. This configuration dropped our performance on test to 48.7% P@1, a decrease of 4.6%, and we additionally lose all justification improvements from our system (see Section 6.6.2), demonstrating that learning this

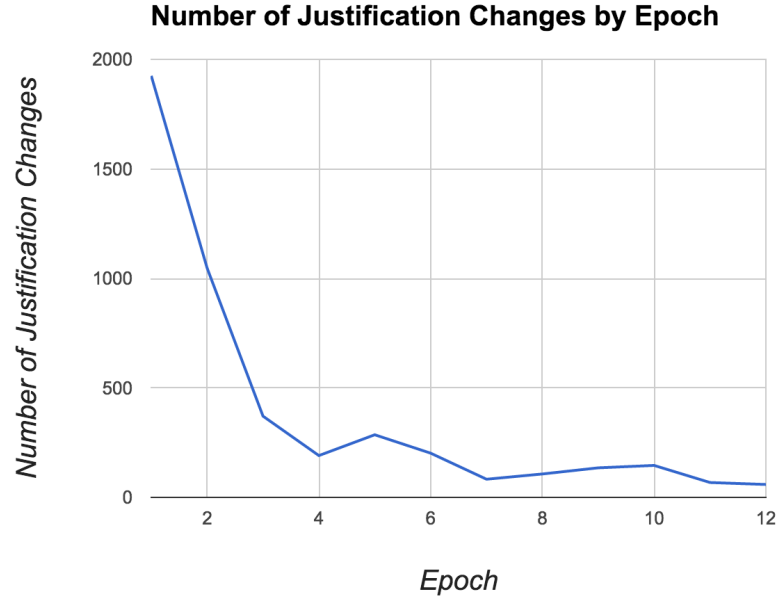


Figure 6.4: Number of questions for which our complete model chooses a new justification at each epoch during training. While this is for a single random seed, we see essentially identical graphs for each random initialization.

reranking is key to our approach.

Additionally, we tracked the number of times a new justification was chosen by the model as it trained. We found that our system converges to a stable set of justifications during training, shown in Figure 6.4.

6.6.3 Error Analysis

To better understand the limitations of our current system, we performed an error analysis of 30¹⁶ incorrectly answered questions. We examined the top 5 justifications returned for both the correct and chosen answers. Notably, 50% of the questions analyzed had one or more good justifications in the top 5 returned by our system, but for a variety of reasons, summarized in Table 6.6, the system incorrectly ranked another justification higher.

¹⁶These were randomly selected from a total of 372 questions that were answered incorrectly by the majority of the models from the five random seeds.

Error Type	Percent
Short justification/High lexical overlap	53.3%
Complex inference required	43.3%
Knowledge Base Noise	6.7%
Word order necessary	6.7%
Coverage	6.7%
Negation	3.3%
Other	6.7%

Table 6.6: Summary of the findings of the 30 question error analysis. Examples of several categories are provided in separate tables. Note that a given question may fall into more than one category.

Type:	Short justification/High lexical overlap
Question:	The length of time between night and day on Earth varies throughout the year. This time variance is explained primarily by ____.
Correct:	Earth 's angle of tilt <i>... the days are very short in the winter because the sun's rays hit the earth at an extreme angle ... due to the tilt of the earth's axis.</i>
Chosen:	Earth 's distance from the Sun <i>Is light year time or distance? Distance</i>

Table 6.7: Example of the system preferring a justification for which all the terms were found in either the question or answer candidate, while the justification for the correct answer contained additional information necessary to fully explain the answer. (Justifications shown in italics)

The table shows that the most common form of error was the system's apparent preference for short justifications with a large degree of lexical overlap with the question and answer choice itself, shown by the example in Table 6.7. The effect was magnified when the correct answer required more explanation to connect the question to the answer. This suggests that the system has learned that generally many unmatched words are indicative of an incorrect answer. While this may typically be true, extending the system to be able to prefer the *opposite* with certain types of questions would potentially help with these errors.

The second largest source of errors came from questions requiring complex infer-

Type:	Complex inference required
Question:	Mr. Harris mows his lawn twice each month. He claims that it is better to leave the clippings on the ground. Which long term effect will this most likely have on his lawn?

Correct: It will provide the lawn with needed nutrients.

Table 6.8: Example of a question for which complex inference is required. In order to answer the question, you would need to assemble the event chain: cut grass left on the ground \rightarrow grass decomposes \rightarrow decomposed material provides nutrients.

Type:	Knowledge base noise
Question:	If an object traveling to the right is acted upon by an unbalanced force from behind it the object will ____.
Correct:	speed up
Chosen	change direction <i>Unbalanced force: force that acts on an object that will change its direction</i>

Table 6.9: Example of a question for which knowledge base noise (here, in the form of over-generalization) was an issue.

ence (causal, process, quantitative, or model-based reasoning) as with the question shown in Table 6.8. This demonstrates not only the difficulty of the question set but also the need for systems that can robustly handle a variety of question types and their corresponding information needs.

Aside from these primary sources of error, there were some smaller trends: 7% of the incorrectly chosen answers actually had justifications which “validated” them due to noise in the knowledge base (e.g., the example shown in Table 6.9), 7% required word-order to answer (e.g., *mass divided by acceleration* vs. *acceleration divided by mass*), another 7% of questions suffered from lack of coverage of the question concept in the knowledge base, and 3% failed to appropriately handle negation (i.e., questions of the format *Which of the following are NOT ...*).

6.7 Conclusion

Here we propose an end-to-end question answering (QA) model that learns to correctly answer questions as well as provide compelling, human-readable justifications for its answers, despite not having access to labels for justification quality. We do this by using the question answering task as a form of distant supervision for learning justification re-ranking. Similar in nature to the model proposed in Chapter 5, we differ primarily in the shallower representation of our knowledge base texts (sentences versus graphlets) and the lack of aggregation in forming justifications. These differences render this model more versatile – they allow us to utilize larger corpora and consequently handle more challenging question sets that require more diverse background information to answer. We show that our accuracy and justification quality are significantly better than a strong IR baseline, while maintaining near state-of-the-art performance for the answer selection task as well. That is, without unduly sacrificing accuracy, with this approach we are able to provide explanations that complete the inference needed to connect questions and their answers.

Of the different types of features included in the model, we found that the explicit lexical overlap features were the most useful, and that the discourse and word embedding-based features each provided only a slight performance boost. For the embedding-based features, this could be due to the limited amount of training data (often, neural networks that successfully learn feature representations operate over orders of magnitude more data than what we have available here). Indeed, we found that when we attempted to update the word embeddings we ran into issues with overfitting.

Regarding the discourse features, with this approach we do not particularly address the different information needs of distinct question types (see 1.1.2; cf., Chapter 4), but intuitively different discourse relations may be more or less relevant to each of them. For example, a *cause* relation may be more relevant to a causal question than it is to a definitional question. Since our approach here does not allow for specialization of the model to different question types, this could be the reason

for the underperformance of this discourse-based feature group. Also, as discussed in the error analysis in Section 6.6.3, of the incorrectly answered questions analyzed, 43.3% required a form of complex inference such as *causal* or *quantitative* reasoning. We hypothesize that by attempting to apply the same model to *all* question types, we aren’t able to specialize enough to handle these more complex information needs. While this framework can be extended to allow the model to learn different priorities for justification selection given different question information needs (perhaps similarly to the domain adaptation-based method that is used in Section 5.6.2), we leave that extension to future work.

Additionally, not performing aggregation to construct justifications constitutes a trade-off – the approach is more lightweight (i.e., faster to run, and able to handle larger corpora within a given resource limit), but it becomes less likely that there exists a *complete* justification for more complicated questions. Likely there is a way to better balance the two – perhaps by performing only selected aggregation during pre-processing, for topics that are more likely to need more than a single sentence to express. Alternatively, perhaps reinforcement learning could be used to learn *when* to add another sentence to an existing candidate justification for a given question and answer. Again, we leave this exploration to future work.

CHAPTER 7

CONCLUSION

In this work, we address the task of question answering (QA), which in simple terms comes down to querying a knowledge base, which could contain unstructured text or some structured information, to get a desired answer. With this work we focus on knowledge bases composed of only unstructured text, the more common form of information available. Beyond the specific data we use here, QA holds promise as a proving ground for developing tools and techniques to be able to navigate the vast sea of text knowledge that is now available, whose sheer size requires automated tools for usage.

A successful QA approach must go beyond simple lexical lookup to perform inference – anticipating the desired result based solely on the content of the question and the given background knowledge (i.e., the knowledge base). As the majority of queries and knowledge base documents are written in natural language, this becomes a bit tougher – developing semantic representations of entire phrases or sentences is extremely difficult due to language variety, and yet when considered in isolation (i.e., a bag-of-words), some words are critical for the inference, some are distracting, and some lose much of their meaning without their context.

Additionally, within a given question set there exists a wide range of question types with different forms of inference needed to solve them (i.e., different information needs). Consider the difference between a question asking for an *example* of a storm versus a question asking about the *result* of a storm – each requires a different form of inference to solve. Consider also a question about the process by which a student might test the effect of sunlight on plants. Unless there happens to be a single sentence explaining the entire process, many separate facts would need to be connected together to provide a complete explanation. While this natural language inference is fairly trivial for a human to do, for an automated system it is very much

a difficult, unsolved problem.

For QA methods which tackle the problem of natural language inference, we assert that to be truly useful, they must be **robust**, in that they should be able to handle a large range of questions with varied vocabulary and phrasing and they should be able to operate over large and diverse knowledge bases to allow for ample coverage. Additionally, we suggest that these methods should be **interpretable**, or able to explain to a non-expert human user *why* the selected answer was selected. Without this ability, even when correct, users will hesitate to trust the output of the system. When incorrect, interpretability is helpful too, as it gives insight into what changes need to be made in order to improve the performance.

Here we proposed a series of approaches that address these two desired attributes, robustness and interpretability, to varying degrees. We began with an approach that focuses primarily on robustness – a monolingual alignment approach in Chapter 3 that we extended to be able to train over any free-text resource, rather than only over aligned question-answer pairs, by aligning the words associated with discourse relations found in the text (Section 3.3). This makes this approach more robust than a traditional QA alignment model, as it extends the corpora over which it can operate, and we demonstrated its success in Section 3.6. This approach is not particularly interpretable, however, as for selected answers the model provides only a set of learned weights and word associations. Additionally, this approach does not, however, address the different information needs present in a question set, rather it attempts to use the same set of learned word association scores for all words in all questions. Following the intuition that certain discourse relations are better suited to certain information needs (e.g., the *explanation* relation might be more relevant to *Why* questions), this approach could be extended to see whether separate sets of alignments, each individualized to correspond to a single discourse relation, could be leveraged to address different types of questions. While this is an investigation we leave to future work, we do propose a different approach that uses a model designed to address a specific relation.

In Chapter 4 we describe a framework for QA that would address different ques-

tion information needs with different models, each tailored to a single need. We train and evaluate a model for one of these needs, *causality*, that makes use of word embeddings. Word embedding models are robust to lexical variation as associations are learned for words not seen together during training, though since they provide only word association scores as output, they are not easily interpretable by users. Additionally, though we demonstrate that this relation-specific model is able to significantly improve upon a general-use model (Section 4.7.3), in terms of extending this method for use with a variety of questions, we do not particularly address the open-ended issue of determining *what* relations would need to be handled. Determining the needed relations by examining a given question set would guarantee some degree of relevancy, but it inherently adds bias and may limit the ability to generalize to unseen questions. In practice, however, this may be the best way to proceed. Even once a working set of relations is identified, classifying questions for which relation is appropriate would also be necessary and non-trivial, as the expense of manual labels would require using machine learning to train a classifier.

Additionally, while it was fairly straightforward to find cause-effect pairs to train the model (since causal language is often fairly explicit, as with phrases such as *X led to Y* or *Y was the result of X*), finding training data for many other relations may be trickier. This suggests that extracted data for other relations may be noisier than what we extracted here for causality, which was itself a bit noisy. While we proposed a simple method for mitigating noise, we hypothesize that improving the way that noise is handled would be very beneficial to any system that extends this approach to operate over more relations.

We also include two approaches that emphasize model interpretability. For each of these we learn how to select both an answer to a question and a human-readable justification for why the answer is correct. As we have no labeled training data for justifications, we use the model’s performance in the QA task as supervision for both answer and justification selection. That is, we score all potential justifications for each answer candidate and use the highest scoring as the current justification. We use the intuition that the best justification for the correct answer should be

better than the best justification for the other, incorrect answers (that is, the correct answer should be better supported by the knowledge base than the incorrect answers). Thus, features from these top-scoring justifications are used to then rank the answer candidates themselves and update the model. We use this basic framework to rerank answers and learn to select justifications in each of our latter two approaches, detailed in Chapters 5 and 6, and in each we demonstrate that the justifications selected are better than those returned by a strong information retrieval baseline, as rated by humans.

In Chapter 5, we make use of a directed graph representation of knowledge base sentences based on a simplification of the sentence syntax, then aggregate multiple sentences together. On the other hand, in Chapter 6 we use only single sentences for our justifications and we use deep learning to learn vector representations of our questions, answers, and justifications. As a result, with the first approach we are able to use finer-grained features that make use of the structure of the justification graphs themselves, as well as features that characterize the lexical overlap between the aggregated sentences. In the second approach, our features are coarser – based only on the surface forms of the sentences (the sequence of words) and the representations learned over them. However, the shallower representation of the knowledge base sentences means faster processing, which in practice allows for usage over the larger text resources which are arguably necessary for the more complicated questions we handle in Chapter 6 (8th grade versus 3rd-5th grade).

Each of the models we propose in this work utilizes a bag-of-words approach. As mentioned above, often the context of a word is necessary to determine the intended meaning, and many times word order is critical. For example, consider the difference between a question asking for the state change of *solid to liquid* versus *liquid to solid*. We are currently unable to handle these questions, though we hope to explore methods that go beyond bag-of-words to robustly handle phrases, sentences, and perhaps even paragraphs to handle arbitrarily more complex questions as well as larger answer and justification texts. Much work has been done on determining how to represent longer spans of text, (inter alia, Le and Mikolov, 2014; Sutskever

et al., 2014; Pagliardini et al., 2017), but many of the popular techniques (including using Long-Short Term Memory Networks (LSTMs; Hochreiter and Schmidhuber, 1997)) require a very large amount of training data and struggle to generalize when presented infrequent or unseen vocabulary and phrasing. Future work will explore available options for text representation, and perhaps explore new methods. We hope to also explore deep learning as a means for learning (a) fixed-length representations of the graphlet structures described in Section 5.5 (Jansen et al., 2017), and then hopefully (b) alignments between a graphlet representation of questions and answers and the graphlets from potential knowledge base sentence justifications.

While we have proposed several methods to address natural language inference for QA, there is still much to be done for this unsolved problem. In these methods described here, as well as those we develop in the future, we plan to continue to invest in models which are robust to question variety, so as to maximize the utility of the model, and which are interpretable, and thus able to encourage confidence in the model predictions through providing a human-readable explanation by which a user can evaluate the provided answer.

CHAPTER 8

Causal Extraction Rules Appendix

The ODIN (Valenzuela-Escárcega et al., 2016) taxonomy and rules used by the approach described in Section 4.4.

Listing 8.1: ODIN taxonomy which provides the hierarchy of extracted entities and events.

```
taxonomy:
- Entity:
  - TransparentChunk
  - NP
  - CM:
    - CM_NP
    - CM_event
  - VP
  - Predicate
- Event:
  - Causal
```

Listing 8.2: Rules for grammatical primitives (e.g. noun phrases, prepositional phrases, etc). These are used in a later step for finding the causal mentions.

```
- name: entity_rule
  label: Entity
  priority : 1
  type: token
  pattern: |
    [word=/.+/]

- name: np_rule_2
  label: NP
  priority : 2
  type: token
  unit: "tag"
  pattern: |
    /^NN/
```

```

- name: vp_rule_1
  label: VP
  priority : 2
  type: token
  unit: "tag"
  pattern: |
    /^VB/

```

Listing 8.3: Rules for extracting causal mentions (CMs), which are the potential arguments for causal events.

```

- name: CM_expansion
  label: CM_event
  priority : 4
  pattern: |
    trigger = [mention=VP & !lemma=/(cause|result|lead|create)/]
    deps:Entity* = /^(nsubj|nn|amod|advmod|ccmod|prep_(with|for|into|on|to)|dobj|
      xcomp|ccomp|agent|vmod)/{1,3}

- name: CM_expansion_2
  label: CM_NP
  priority : 3
  pattern: |
    trigger = [mention=NP & !lemma=/(cause|result)/]
    deps:Entity* = /^(nn|amod|advmod|ccmod|prep_(of|with|for|into|on|to|in)|dobj)
      /{1,3}

- name: CM_expansion_3_copular
  label: CM_event
  priority : 3
  pattern: |
    trigger = [lemma=be]
    deps:Entity* = /^(nsubj|prep_(in|on|with|for|of)|dobj|nn|amod)/+

- name: nounphrase_with_property_3
  label: CM_NP
  priority : 5
  example: "... will cause huge <np> damage to the reef <\np>."
  pattern: |
    trigger = [mention=CM & !lemma=/(cause|result|lead|create)/]
    modifier:CM = <dobj [lemma=cause & tag=^VB/] >/(prep_(to|for|of))|xcomp/

```

Listing 8.4: Rules for causal relations (events). Note that the cause and effect arguments are previously found causal mentions.

```

- name: PosReg_syntax_1_verb
  label: Causal
  priority: 9
  example: ""
  pattern: |
    trigger = [lemma=cause]
    controlled:CM+ = prepc_by? (dobj | xcomp | ccomp)
    controller:CM+ = <xcomp? (nsubj | agent | <vmod) /appos|nn|conj_|cc|prep_of|
      prep_in/{,2}

- name: causal_2
  label: Causal
  priority: 9
  example: "X will cause huge <np> damage to the reef <\np>."
  pattern: |
    trigger = [lemma=cause & tag=/^VB/]
    controlled:CM+ = dobj
    controller:CM+ = nsubj

- name: causal_3
  label: Causal
  priority: 9
  example: "X cause Y to Z"
  pattern: |
    trigger = [lemma=cause & tag=/^VB/]
    controlled:CM_event+ = xcomp
    controller:CM+ = nsubj

- name: causal_4
  label: Causal
  priority: 9
  example: "X was the cause of Y"
  pattern: |
    trigger = [lemma=cause & tag=/^NN/]
    controlled:CM+ = prep_of
    controller:CM+ = nsubj

- name: causal_5
  label: Causal
  priority: 9
  example: "X was caused by Y"

```

- pattern: |
 trigger = [lemma=cause & tag=/^VB/]
 controlled:CM+ = nsubjpass
 controller:CM+ = agent
- name: causal_6
 label: Causal
 priority: 9
 example: "They did not provide a motive for the alleged crime , which caused the
 city 's first fire -related deaths in more than a decade ."
 pattern: |
 trigger = [lemma=cause & tag=/^VB/]
 controlled:CM_NP = <rcmod (?= >rcmod >nsubj [tag=WDT])
 controller:CM+ = dobj
- name: causal_7
 label: Causal
 priority: 9
 example: "After the exam grades begin to count , students often start taking them
 more seriously , which causes passage rates to increase , Jennings said ."
 pattern: |
 trigger = [lemma=cause & tag=/^VB/]
 controlled:CM_NP = <advcl (?= >advcl >nsubj [tag=WDT])
 controller:CM+ = dobj
- name: result_1
 label: Causal
 priority: 9
 example: "X resulted in Y"
 pattern: |
 trigger = [lemma=result & tag=/^VB/]
 controlled:CM+ = prep_in
 controller:CM+ = nsubj
- name: result_2
 label: Causal
 priority: 9
 example: "X as a result of Y"
 pattern: |
 trigger = [lemma=result & tag=/^NN/]
 controlled:CM+ = <prep_as
 controller:CM+ = prep_of

- name: result_3
label: Causal
priority : 9
example: "X was the result of Y"
pattern: |
trigger = [lemma=result & tag=/^NN/]
controlled:CM+ = nsubj
controller:CM+ = prep_of

- name: result_4
label: Causal
priority : 9
example: "Jet lag results when the body's internal clock is out of sync with
daily life , making people sleepy when they want to be awake and wakeful
when they want to sleep."
pattern: |
trigger = [lemma=result & tag=/^VB/]
controlled:CM+ = nsubj
controller:CM+ = /(advcl|prep_from)/

- name: led_to_1
label: Causal
priority : 9
example: "X led to Y"
pattern: |
trigger = [lemma=lead & tag=/^(VB|NN)/] to
controller:CM+ = nsubj
controlled:CM+ = prep_to

- name: create_1
label: Causal
priority : 9
example: "X was created by Y"
pattern: |
trigger = [lemma=create & tag=/^VB/]
controlled:CM+ = nsubjpass
controller:CM+ = agent

REFERENCES

- arXiv (2017). arXiv monthly submission rates. [Online; accessed 24-July-2017].
- Balduccini, M., C. Baral, and Y. Lierler (2008). Knowledge representation and question answering. *Foundations of Artificial Intelligence*, **3**, pp. 779–819.
- Banarescu, L., C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider (2013). Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pp. 178–186. Association for Computational Linguistics.
- Baral, C. and S. Liang (2012). From Knowledge Represented in Frame-Based Languages to Declarative Representation and Reasoning via ASP. In *KR*.
- Baral, C., S. Liang, and V. Nguyen (2011). Towards deep reasoning with respect to natural language text in scientific domains. In *DeepKR Workshop*. Citeseer.
- Baral, C., N. H. Vo, and S. Liang (2012). Answering Why and How questions with respect to a frame-based knowledge base: a preliminary report. In *ICLP (Technical Communications)*, pp. 26–36. Citeseer.
- Barzilay, R. and K. R. McKeown (2005). Sentence fusion for multidocument news summarization. *Computational Linguistics*, **31**(3), pp. 297–328.
- Barzilay, R., K. R. McKeown, and M. 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*, pp. 550–557. Association for Computational Linguistics.
- Berger, A., R. Caruana, D. Cohn, D. Freytag, and V. 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, A. and J. 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, S., K. McKeown, and A. H. Schlaikjer (2003). A hybrid approach for answering definitional questions. *Technical Report CUCS-006-03, Columbia University*.

- Bordes, A., S. Chopra, and J. Weston (2014). Question Answering with Subgraph Embeddings. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP) 2014*.
- Bordes, A., N. Usunier, S. Chopra, and J. Weston (2015). Large-scale Simple Question Answering with Memory Networks. *CoRR*, **abs/1506.02075**.
- Bordes, A., N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko (2013). Translating Embeddings for Modeling Multi-relational Data. In *NIPS*.
- Brown, P. F., S. A. D. Pietra, V. J. D. Pietra, and R. L. Mercer (1993). The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, **19**(2), pp. 263–311.
- Brysbaert, M., A. Warriner, and V. Kuperman (2014). Concreteness ratings for 40 thousand generally known English word lemmas. *Behavior Research Methods*, **46**(3), pp. 904–911. doi:10.3758/s13428-013-0403-5.
- Chambers, N., D. Cer, T. Grenager, D. Hall, C. Kiddon, B. MacCartney, M.-C. De Marneffe, D. Ramage, E. Yeh, and C. D. Manning (2007). Learning alignments and leveraging natural logic. In *Proc. of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pp. 165–170.
- Chen, D., J. Bolton, and C. D. Manning (2016). A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task. In *Association for Computational Linguistics (ACL)*.
- Chen, D. and C. D. Manning (2014a). A Fast and Accurate Dependency Parser using Neural Networks. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.
- Chen, D. and C. D. Manning (2014b). A Fast and Accurate Dependency Parser using Neural Networks. In *Empirical MNLP*, pp. 740–750.
- Chollet, F. (2015). Keras. <https://github.com/fchollet/keras>.
- Chu-Carroll, J., K. Czuba, J. M. Prager, A. Ittycheriah, and S. Blair-Goldensohn (2004). IBM’s PIQUANT II in TREC 2004. In *Text Retrieval Conference (TREC)*.
- Clark, P. (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, January 25-30, 2015, Austin, Texas, USA.*, pp. 4019–4021.
- Clark, P., P. Harrison, and N. Balasubramanian (2013a). 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*, pp. 37–42.

- Clark, P., P. Harrison, and N. Balasubramanian (2013b). A study of the knowledge base requirements for passing an elementary science test. In *Proc. of the 2013 workshop on Automated Knowledge Base Construction (AKBC)*, pp. 37–42.
- Cole, S. V., M. D. Royal, M. G. Valtorta, M. N. Huhns, and J. B. Bowles (2005). A lightweight tool for automatically extracting causal relationships from text. In *SoutheastCon, 2006. Proc. of the IEEE*, pp. 125–129.
- Collins, M. (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, pp. 1–8. Association for Computational Linguistics, Stroudsburg, PA, USA. doi:10.3115/1118693.1118694.
- Craven, M. W. and J. W. Shavlik (1996). Extracting tree-structured representations of trained networks. *Advances in neural information processing systems*, pp. 24–30.
- Cunningham, H., D. Maynard, K. Bontcheva, V. Tablan, N. Aswani, I. Roberts, G. Gorrell, A. Funk, A. Roberts, D. Damjanovic, T. Heitz, M. A. Greenwood, H. Saggion, J. Petrak, Y. Li, and W. Peters (2011). *Text Processing with GATE (Version 6)*. ISBN 978-0956599315.
- Curran, J. R., T. Murphy, and B. Scholz (2007). Minimising semantic drift with mutual exclusion bootstrapping. In *Proc. of the 10th Conference of the Pacific Association for Computational Linguistics*, volume 3.
- Daumé III, H. (2007). Frustratingly Easy Domain Adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 256–263. Association for Computational Linguistics, Prague, Czech Republic.
- De Marneffe, M.-C. and C. D. Manning (2008). Stanford typed dependencies manual. Technical report, Technical report, Stanford University.
- Do, Q. X., Y. S. Chan, and D. Roth (2011). Minimally supervised event causality identification. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 294–303.
- Dong, L., F. Wei, M. Zhou, and K. Xu (2015). Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of Association for Computational Linguistics*, pp. 260–269.
- Du, X., J. Shao, and C. Cardie (2017). Learning to Ask: Neural Question Generation for Reading Comprehension. In *Association for Computational Linguistics (ACL)*.

- Echihabi, A. and D. Marcu (2003a). A noisy-channel approach to question answering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pp. 16–23. Association for Computational Linguistics.
- Echihabi, A. and D. Marcu (2003b). A Noisy-Channel Approach to Question Answering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 16–23. Sapporo, Japan.
- Etzioni, O. (2011). Search needs a shake-up. *Nature*, **476**(7358), pp. 25–26.
- Faruqui, M., Y. Tsvetkov, R. Rastogi, and C. Dyer (2016). Problems With Evaluation of Word Embeddings Using Word Similarity Tasks. *arXiv preprint arXiv:1605.02276*.
- Feng, V. W. and G. Hirst (2012). Text-level Discourse Parsing with Rich Linguistic Features. In *Proceedings of the Association for Computational Linguistics*.
- Fernandes, E. R., C. N. Dos Santos, and R. L. 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*, pp. 41–48. Association for Computational Linguistics.
- Ferrucci, D., E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, et al. (2010). Building Watson: An overview of the DeepQA project. *AI magazine*, **31**(3), pp. 59–79.
- Finkel, J. R. and C. 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*, pp. 720–728. Association for Computational Linguistics.
- FitzGerald, N., O. Täckström, K. Ganchev, and D. Das (2015). Semantic Role Labeling with Neural Network Factors. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 960–970.
- Fried, D., P. Jansen, G. Hahn-Powell, M. Surdeanu, and P. Clark (2015). Higher-order Lexical Semantic Models for Non-factoid Answer Reranking. *Transactions of the Association for Computational Linguistics*, **3**, pp. 197–210.
- Girju, R. and D. I. Moldovan (2002). Text mining for causal relations. In *FLAIRS Conference*, pp. 360–364.
- Halliday, M. A. K. and R. Hasan (2014). *Cohesion in english*. Routledge.

- Harabagiu, S., D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Girju, V. Rus, and P. Morarescu (2000). Falcon: Boosting Knowledge for Answer Engines. In *Proceedings of the Text REtrieval Conference (TREC)*. Gaithersburg, MD, USA.
- He, X. and D. Golub (2016). Character-Level Question Answering with Attention. In *EMNLP*.
- Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *Proc. of the 14th conference on Computational linguistics (COLING)*, pp. 539–545.
- Heilman, M. and N. A. Smith (2010). Good Question! Statistical Ranking for Question Generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 609–617. Association for Computational Linguistics.
- Hendrickx, I., S. N. Kim, Z. Kozareva, P. Nakov, D. Ó Séaghdha, S. Padó, M. Pennacchiotti, L. Romano, and S. Szpakowicz (2009). Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proc. of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pp. 94–99.
- Hermann, K. M., T. Kočiský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom (2015). Teaching Machines to Read and Comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.
- Hernault, H., H. Prendinger, D. duVerle, and M. Ishizuka (2010). HILDA: A Discourse Parser Using Support Vector Machine Classification. *Dialogue and Discourse*, 1(3), pp. 1–33.
- Hickl, A., J. Williams, J. Bensley, K. Roberts, B. Rink, and Y. Shi (2006). Recognizing textual entailment with LCCs GROUNDHOG system. In *Proceedings of the Second PASCAL Challenges Workshop*.
- Hochreiter, S. and J. Schmidhuber (1997). Long short-term memory. *Neural computation*, 9(8), pp. 1735–1780.
- Hoffmann, R., C. Zhang, X. Ling, L. Zettlemoyer, and D. 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*, pp. 541–550. Association for Computational Linguistics.

- Iyyer, M., V. Manjunatha, J. Boyd-Graber, and H. Daumé III (2015a). Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the Association for Computational Linguistics*.
- Iyyer, M., V. Manjunatha, J. Boyd-Graber, and H. Daumé III (2015b). Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *Association for Computational Linguistics*.
- Jansen, P., N. Balasubramanian, M. Surdeanu, and P. Clark (2016). What’s in an Explanation? Characterizing Knowledge and Inference Requirements for Elementary Science Exams. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 2956–2965. The COLING 2016 Organizing Committee, Osaka, Japan.
- Jansen, P., R. Sharp, M. Surdeanu, and P. Clark (2017). Framing QA as Building and Ranking Intersentence Answer Justifications. *Computational Linguistics*.
- Jansen, P., M. Surdeanu, and P. 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, D., T. Khot, A. Sabharwal, P. Clark, O. Etzioni, and D. 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)*, pp. 1145–1152.
- Khoo, C. S., J. Kornfilt, R. N. Oddy, and S. H. Myaeng (1998). Automatic extraction of cause-effect information from newspaper text without knowledge-based inferencing. *Literary and Linguistic Computing*, **13**(4), pp. 177–186.
- Khot, T., A. Sabharwal, and P. Clark (2017). Answering Complex Questions Using Open Information Extraction. In *Proceedings of Association for Computational Linguistics (ACL)*.
- Kiela, D., F. Hill, and S. Clark (2015). Specializing Word Embeddings for Similarity or Relatedness. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Kim, B., J. A. Shah, and F. Doshi-Velez (2015). Mind the Gap: A Generative Approach to Interpretable Feature Selection and Extraction. In *NIPS*.
- Kingma, D. and J. Ba (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Le, Q. V. and T. Mikolov (2014). Distributed Representations of Sentences and Documents. In *ICML*.

- Lei, T., R. Barzilay, and T. S. Jaakkola (2016). Rationalizing Neural Predictions. In *EMNLP*.
- Letham, B., C. Rudin, T. H. McCormick, D. Madigan, et al. (2015). Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, **9**(3), pp. 1350–1371.
- Levy, O. and Y. Goldberg (2014). Dependency-Based Word Embeddings. In *Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 302–308.
- Levy, O., S. Remus, C. Biemann, I. Dagan, and I. Ramat-Gan (2015). Do supervised distributional methods really learn lexical inference relations. In *Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.
- Lewis, M. and M. Steedman (2013). Combining Distributional and Logical Semantics. *Transactions of the Association for Computational Linguistics*, **1**, pp. 179–192.
- Li, J., X. Chen, E. H. Hovy, and D. Jurafsky (2016). Visualizing and Understanding Neural Models in NLP. In *HLT-NAACL*.
- Liang, P., A. Bouchard-Côté, D. Klein, and B. 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*, pp. 761–768. Association for Computational Linguistics.
- Liang, P., M. I. Jordan, and D. Klein (2013). Learning dependency-based compositional semantics. *Computational Linguistics*, **39**(2), pp. 389–446.
- MacCartney, B. (2009). *Natural language inference*. Ph.D. thesis, Citeseer.
- Mann, W. C. and S. A. Thompson (1988). Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text*, **8**(3), pp. 243–281.
- Manning, C. D., P. Raghavan, and H. Schütze (2008). *Introduction to Information Retrieval*. Cambridge University Press.
- Manning, C. D., M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky (2014a). The Stanford CoreNLP Natural Language Processing Toolkit. In *Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

- Manning, C. D., M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky (2014b). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55–60.
- Marcu, D. (1997). *The Rhetorical Parsing, Summarization, and Generation of Natural Language Texts*. Ph.D. thesis, University of Toronto.
- McSherry, F. and M. Najork (2008). Computing Information Retrieval Performance Measures Efficiently in the Presence of Tied Scores. In *30th European Conference on IR Research (ECIR)*. Springer-Verlag.
- Mikolov, T., K. Chen, G. Corrado, and J. Dean (2013a). Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
- Mikolov, T., M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur (2010). Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*.
- Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119.
- Moldovan, D., C. Clark, S. Harabagiu, and D. Hodges (2007). Cogex: A semantically and contextually enriched logic prover for question answering. *Journal of Applied Logic*, **5**(1), pp. 49–69.
- Moldovan, D., C. Clark, S. Harabagiu, and S. 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*, pp. 87–93. Association for Computational Linguistics.
- Moldovan, D., M. Paşca, S. Harabagiu, and M. Surdeanu (2003b). Performance Issues and Error Analysis in an Open-domain Question Answering System. *ACM Trans. Inf. Syst.*, **21**(2), pp. 133–154. ISSN 1046-8188. doi:10.1145/763693.763694.
- Moldovan, D. I. and V. Rus (2001). Logic form transformation of wordnet and its applicability to question answering. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pp. 402–409. Association for Computational Linguistics.

- Napoles, C., M. Gormley, and B. Van Durme (2012a). Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX '12, pp. 95–100. Association for Computational Linguistics, Stroudsburg, PA, USA.
- Napoles, C., M. Gormley, and B. Van Durme (2012b). Annotated gigaword. In *Proc. of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pp. 95–100.
- Och, F. J. and H. Ney (2003). A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, **29**(1), pp. 19–51.
- Oh, J.-H., K. Torisawa, C. Hashimoto, M. Sano, S. De Saeger, and K. Ohtake (2013). Why-Question Answering using Intra-and Inter-Sentential Causal Relations. In *The 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1733–1743.
- Pagliardini, M., P. Gupta, and M. Jaggi (2017). Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. *CoRR*, **abs/1703.02507**.
- Parikh, A. P., O. Täckström, D. Das, and J. Uszkoreit (2016). A Decomposable Attention Model for Natural Language Inference. In *EMNLP*.
- Park, J. H. and W. B. 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*, SIGIR '15, pp. 927–930. ACM, New York, NY, USA. ISBN 978-1-4503-3621-5. doi:10.1145/2766462.2767768.
- Piaget, J. (1954). *The Construction of Reality in the Child*. Basic Books.
- Pradhan, S. S., V. Krugler, S. Bethard, W. Ward, D. Jurafsky, J. H. Martin, S. Blair-Goldensohn, A. H. Schlaikjer, E. Filatova, P. A. Duboué, et al. (2002). Building a Foundation System for Producing Short Answers to Factual Questions. In *TREC*.
- Real Time Statistics Project (2017). Internet Live Stats. [Online; accessed 25-July-2017].
- Reece, J., L. Urry, M. Cain, S. Wasserman, and P. Minorsky (2011). *Campbell Biology*. Pearson Benjamin Cummings. ISBN 9780321558237.
- Ribeiro, M. T., S. Singh, and C. Guestrin (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *HLT-NAACL Demos*.
- Riedel, S., L. Yao, A. McCallum, and B. M. Marlin (2013). Relation extraction with matrix factorization and universal schemas. In *Proc. of Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

- Riezler, S., A. Vasserman, I. Tsochantaridis, V. Mittal, and Y. 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)*, pp. 464–471.
- Riloff, E. (1996). Automatically generating extraction patterns from untagged text. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pp. 1044–1049.
- Sachan, M., A. Dubey, and E. P. Xing (2016). Science Question Answering using Instructional Materials. In *The 54th Annual Meeting of the Association for Computational Linguistics*, p. 467.
- Serban, I. V., A. García-Durán, Ç. Gülçehre, S. Ahn, S. Chandar, A. Courville, and Y. Bengio (2016). Generating Factoid Questions With Recurrent Neural Networks: The 30M Factoid Question-Answer Corpus. In *54th Annual Meeting of the Association for Computational Linguistics*, p. 588. Association for Computational Linguistics.
- Severyn, A. and A. Moschitti (2012). Structural relationships for large-scale learning of answer re-ranking. In *SIGIR*.
- Severyn, A. and A. Moschitti (2013). Automatic Feature Engineering for Answer Selection and Extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Severyn, A. and A. Moschitti (2015). Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In *SIGIR*.
- Severyn, A., M. Nicosia, and A. Moschitti (2013). Learning Adaptable Patterns for Passage Reranking. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning (CoNLL)*.
- Sharma, A., N. H. Vo, S. Aditya, and C. 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, R., P. Jansen, M. Surdeanu, and P. 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*, pp. 231–237. Association for Computational Linguistics, Denver, Colorado.

- Shen, L. and A. K. Joshi (2005). Ranking and Reranking with Perceptron. *Machine Learning. Special Issue on Learning in Speech and Language Technologies*, **60**(1), pp. 73–96.
- Soricut, R. and E. Brill (2006). Automatic Question Answering Using the Web: Beyond the Factoid. *Journal of Information Retrieval - Special Issue on Web Information Retrieval*, **9**(2), pp. 191–206.
- Sultan, M. A., S. Bethard, and T. Sumner (2014). Back to Basics for Monolingual Alignment: Exploiting Word Similarity and Contextual Evidence. *Transactions of the Association for Computational Linguistics*, **2**, pp. 219–230.
- Sun, X., T. Matsuzaki, D. Okanohara, and J. Tsujii (2009). Latent Variable Perceptron Algorithm for Structured Classification. In *IJCAI*, volume 9, pp. 1236–1242.
- Surdeanu, M., M. Ciaramita, and H. Zaragoza (2011). Learning to Rank Answers to Non-Factoid Questions from Web Collections. *Computational Linguistics*, **37**(2), pp. 351–383.
- Surdeanu, M., T. Hicks, and M. A. Valenzuela-Escárcega (2015). Two Practical Rhetorical Structure Theory Parsers. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL): Software Demonstrations*.
- Sutskever, I., O. Vinyals, and Q. V. Le (2014). Sequence to Sequence Learning with Neural Networks. In *NIPS*.
- Tan, M., C. N. dos Santos, B. Xiang, and B. Zhou (2016). Improved Representation Learning for Question Answer Matching. In *ACL*.
- Tari, L. and C. Baral (2006). Using AnsProlog with Link Grammar and WordNet for QA with deep reasoning. In *Information Technology, 2006. ICIT'06. 9th International Conference on*, pp. 125–128. IEEE.
- Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, **abs/1605.02688**.
- Tieleman, T. and G. Hinton (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude.
- Valenzuela-Escárcega, M. A., G. Hahn-Powell, and M. Surdeanu (2016). Odin’s Runes: A Rule Language for Information Extraction. In *Proc. of the 10th International Conference on Language Resources and Evaluation (LREC)*.

- Verberne, S., L. Boves, N. Oostdijk, P.-A. Coppen, et al. (2007). Discourse-based answering of why-questions. *Traitement Automatique des Langues, Discours et document: traitements automatiques*, **47**(2), pp. 21–41.
- Voorhees, E. 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, pp. 109–111. Association for Computational Linguistics, Stroudsburg, PA, USA. doi:10.3115/1073483.1073520.
- Wang, D. and E. Nyberg (2015). A long short-term memory model for answer sentence selection in question answering. *ACL, July*.
- Wang, M. and C. D. Manning (2010). Probabilistic Tree-Edit Models with Structured Latent Variables for Textual Entailment and Question Answering. In *COLING*.
- Wikipedia (2017). Wikipedia:Statistics. [Online; accessed 24-July-2017].
- Woodsend, K. and M. Lapata (2015). Distributed Representations for Unsupervised Semantic Role Labeling. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Yang, M.-C., N. Duan, M. Zhou, and H.-C. Rim (2014). Joint Relational Embeddings for Knowledge-based Question Answering. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 645–650.
- Yang, X. and K. Mao (2014). Multi level causal relation identification using extended features. *Expert Systems with Applications*, **41**(16), pp. 7171–7181.
- Yao, X., B. Van Durme, C. Callison-Burch, and P. Clark (2013). Semi-Markov Phrase-based Monolingual Alignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Yih, W.-t., M.-W. Chang, C. Meek, and A. Pastusiak (2013). Question Answering Using Enhanced Lexical Semantic Models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Zeiler, M. D. and R. Fergus (2014). Visualizing and Understanding Convolutional Networks. In *ECCV*.
- Zettlemoyer, L. S. and M. 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)*, pp. 678–687.