REASONING-DRIVEN QUESTION-ANSWERING FOR NATURAL LANGUAGE UNDERSTANDING

Daniel Khashabi

A DISSERTATION

in

Computer and Information Sciences

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy 2019

Supervisor of Dissertation

Dan Roth, Professor of Computer and Information Science

Graduate Group Chairperson

Rajeev Alur, Professor of Computer and Information Science

Dissertation Committee

Dan Roth, Professor, Computer and Information Science, University of Pennsylvania

Mitch Marcus, Professor of Computer and Information Science, University of Pennsylvania

Zachary Ives, Professor of Computer and Information Sciences, University of Pennsylvania

Chris Callison-Burch, Associate Professor of Computer Science, University of Pennsylvania

Ashish Sabharwal, Senior Research Scientist, Allen Institute for Artificial Intelligence

REASONING-DRIVEN QUESTION-ANSWERING
FOR NATURAL LANGUAGE UNDERSTANDING

Dedicated to the loving memory of my gramma, An'nah

Your patience and kindness will forever stay with me.

ACKNOWLEDGEMENT

I feel incredibly lucky to have Dan Roth as my advisor. I am grateful to Dan for trusting me, especially when I had only a basic understanding of many key challenges in natural language. It took me a while to catch up with *what is important* in the field and be able to communicate the challenges effectively. During these years, Dan's vision has always been the guiding principle to many of my works. His insistence on focusing on the long-term progress, rather than "easy" wins, shaped the foundation of many of the ideas I pursued. This perspective pushed me to think differently than the popular trends. It has been a genuine privilege to work together.

I want to thank my thesis committee at UPenn, Mitch Marcus, Zach Ives and Chris Callison-Burch for being a constant source of invaluable feedback and guidance. Additionally, I would like to thanks the many professors who have touched parts of my thinking: Jerry DeJong, for encouraging me read the classic literature; Chandra Chekuri and Avrim Blum, for their emphasis on intuition, rather than details; and my undergraduate advisor Hamid Sheikhzadeh Nadjar, for encouraging me to work on important problems.

A huge thank you to the Allen Institute for Artificial Intelligence (AI2) for much support during my PhD studies. Any time I needed any resources (computing resources, crowdsourcing credits, engineering help, etc), without any hesitation, AI2 has provided me what was needed. Special thanks to Ashish Sabhwaral and Tushar Khot for being a constant source of wisdom and guidance, and investing lots of time and effort. They both have always been present to listen to my random thoughts, almost on a weekly basis. I am grateful to other members of AI2 for their help throughout my projects: Oren Etzioni, Peter Clark, Oyvind Tafjord, Peter Turney, Ingmar Ellenberger, Dirk Groeneveld, Michael Schmitz, Chandra Bhagavatula and Scott Yih. Moreover, I would like to remember Paul Allen (1953-2018): his vision and constant generous support has tremendously changed our field (and my life, in particular).

My collaborators, especially past and present CogComp members, have been major contributors and influencers throughout my works. I would like to thank Mark Sammons, Vivek Srikumar, Christos Christodoulopoulos, Erfan Sadeqi Azer, Snigdha Chaturvedi, Kent Quanrud, Amirhossein Taghvaei, Chen-Tse Tsai, and many other CogComp members. Furthermore, I thank Eric Horn and Jennifer Sheffield for their tremendous contributions to many of my write-ups. And thank you to all the friends I have made at Penn, UIUC, and elsewhere, for all the happiness you've brought me. Thanks to Whitney, for sharing many happy and sad moments with me, and for helping me become a better version of myself.

Last, but never least, my family, for their unconditional sacrifice and support. I wouldn't have been able to go this far without you.

ABSTRACT

REASONING-DRIVEN QUESTION-ANSWERING FOR NATURAL LANGUAGE UNDERSTANDING

Daniel Khashabi

Dan Roth

Natural language understanding (NLU) of text is a fundamental challenge in AI, and it has received significant attention throughout the history of NLP research. This primary goal has been studied under different tasks, such as Question Answering (QA) and Textual Entailment (TE). In this thesis, we investigate the NLU problem through the QA task and focus on the aspects that make it a challenge for the current state-of-the-art technology. This thesis is organized into three main parts:

In the first part, we explore multiple formalisms to improve existing machine comprehension systems. We propose a formulation for abductive reasoning in natural language and show its effectiveness, especially in domains with limited training data. Additionally, to help reasoning systems cope with irrelevant or redundant information, we create a supervised approach to learn and detect the essential terms in questions.

In the second part, we propose two new challenge datasets. In particular, we create two datasets of natural language questions where (i) the first one requires reasoning over multiple sentences; (ii) the second one requires *temporal common sense* reasoning. We hope that the two proposed datasets will motivate the field to address more complex problems.

In the final part, we present the first formal framework for multi-step reasoning algorithms, in the presence of a few important properties of language use, such as incompleteness, ambiguity, etc. We apply this framework to prove fundamental limitations for reasoning algorithms. These theoretical results provide extra intuition into the existing empirical evidence in the field.

TABLE OF CONTENTS

[1]Terms originally made by Roger Schank to characterize two different camps: the first group that represented commonsense knowledge in the form of large amorphous semantic networks, as opposed to another from the camp of whose work was based on logic and formal extensions of logic.

LIST OF TABLES

LIST OF ILLUSTRATIONS

PUBLICATION NOTES

1. Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Peter Clark, Oren Etzioni, and Dan Roth. Question answering via integer programming over semi-structured knowledge. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI), 2016*. URL `http://cogcomp.org/page/publication_view/786`.

2. Daniel Khashabi, Tushar Khot, Ashish Sabharwal, and Dan Roth. Learning what is essential in questions. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL), 2017*. URL `http://cogcomp.org/page/publication_view/813`.

3. Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), 2018a*. URL `http://cogcomp.org/page/publication_view/833`.

4. Daniel Khashabi, Tushar Khot, Ashish Sabharwal, and Dan Roth. Question answering as global reasoning over semantic abstractions. In *Proceedings of the Fifteenth Conference on Artificial Intelligence (AAAI), 2018b.* URL `http://cogcomp.org/page/publication_view/824`.

5. Daniel Khashabi, Erfan Sadeqi Azer, Tushar Khot, Ashish Sabharwal, and Dan Roth. On the capabilities and limitations of reasoning for natural language understanding, 2019. URL `https://arxiv.org/abs/1901.02522`

6. Ben Zhou, Daniel Khashabi, Qiang Ning, and Dan Roth. "Going on a vacation" takes longer than "Going for a walk": A Study of Temporal Commonsense Understanding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2019.*

CHAPTER 1 : Introduction

> "To model this language understanding process in a computer, we need a program which combines grammar, semantics, and reasoning in an intimate way, concentrating on their interaction."
>
> — T. Winograd, Understanding Natural Language, 1972

1.1. Motivation

The purpose of *Natural Language Understanding* (NLU) is to enable systems to interpret a given text, as close as possible to the many ways humans would interpret it.

Improving NLU is increasingly changing the way humans interact with machines. The current NLU technology is already making significant impacts. For example, we can see it used by speech agents, including Alexa, Siri, and Google Assistant. In the near future, with better NLU systems, we will witness a more active presence of these systems in our daily lives: social media interactions, in financial estimates, during the course of product recommendation, in accelerating of scientific findings, etc.

The importance of NLU was understood by many pioneers in Artificial Intelligence (starting in the '60s and '70s). The initial excitement about the field ushered a decade of activity in this area (McCarthy, 1963; Winograd, 1972; Schank, 1972; Woods, 1973; Zadeh, 1978). The beginning of these trends was overly positive at times, and it took years (if not decades) to comprehend and appreciate the real difficulty of language understanding.

1.2. Challenges along the way to NLU

We, humans, are so used to using language that it's almost impossible to see its complexity, without a closer look into instances of this problem. As an example, consider the story shown in Figure 1, which appeared in an issue of the New York Times (taken from McCarthy (1976)). With relatively simple wording, this story is understandable to English speakers. Despite the simplicity, many nuances have to come together to form a coherent understanding of this story.

A 61-year-old furniture salesman was pushed down the shaft of a freight elevator yesterday in his downtown Brooklyn store by two robbers while a third attempted to crush him with the elevator car because they were dissatisfied with the $1,200 they had forced him to give them.
The buffer springs at the bottom of the shaft prevented the car from crushing the salesman, John J. Hug, after he was pushed from the first floor to the basement. The car stopped about 12 inches above him as he flattened himself at the bottom of the pit.
Mr. Hug was pinned in the shaft for about half an hour until his cries attracted the attention of a porter. The store at 340 Livingston Street is part of the Seamans Quality Furniture chain.
Mr. Hug was removed by members of the Police Emergency Squad and taken to Long Island College Hospital. He was badly shaken, but after being treated for scrapes of his left arm and for a spinal injury was released and went home. He lives at 62-01 69th Lane, Maspeth, Queens.
He has worked for seven years at the store, on the corner of Nevins Street, and this was the fourth time he had been held up in the store. The last time was about one year ago, when his right arm was slashed by a knife-wielding robber.

Figure 1: A sample story appeared on the New York Times (taken from McCarthy (1976)).

We flesh out a few general factors which contribute to the complexity of language understanding in the context of the story given in Figure 1:

- *Ambiguity* comes along when trying to make sense of a given string. While an average human might be good at this, it's incredibly hard for machines to map symbols or characters to their actual meaning. For example, the mention of "car" that appears in our story has multiple meanings (see Figure 2; left). In particular, this mention in the story refers to a sense other than its usual meaning (here refers to *the elevator*

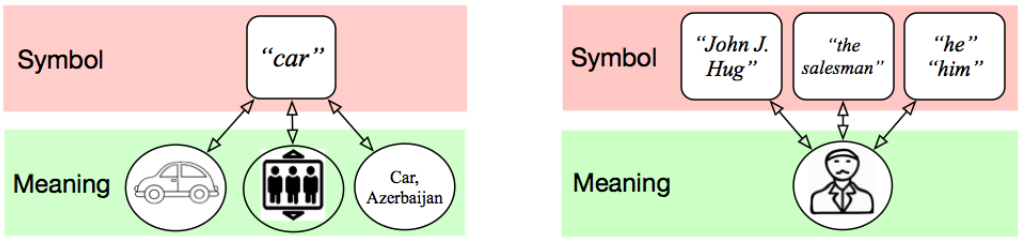


Figure 2: *Ambiguity* (left) appears when mapping a raw string to its actual meaning; *Variability* (right) is having many ways of referring to the same meaning.

cabin; the usual meaning is a *road vehicle*).

- *Variability* of language means that a single idea could be phrased in many different ways. For instance, the same character in the story, "Mr. Hug," has been referred to in different ways: "the salesman," "he," "him," "himself," etc. Beyond lexical level, there is even more variability in bigger constructs of language, such as phrases, sentences, paragraphs, etc.

- Reading and understanding text involves an implicit formation of a mental structure with many elements. Some of these elements are directly described in the given story, but a significant portion of the understanding involves information that is *implied* based on a readers' background knowledge. *Common sense* refers to our (humans) understanding of everyday activities (e.g., sizes of objects, duration of events, etc), usually shared among many individuals. Take the following sentence from the story:

> The car stopped about 12 inches above him as he flattened himself at the bottom of the pit.

  There is a significant amount of imagination hiding in this sentence; each person after reading this sentence has a mental picture of the incident. And based on this mental picture, we have implied meanings: *we know he is lucky to be alive now; if he didn't flatten himself, he would have died; he had nowhere to go at the bottom of the pit; the car is significantly heavier than the man*; etc. Such understanding is common and easy for humans and rarely gets direct mention in text, since they are considered trivial (for humans). Humans are able to form such implicit understanding as a result of our own world model and past shared experiences.

- *Many small bits combine to make a big picture.* We understand that "downtown Brooklyn" is probably not a safe neighborhood, since "this was the fourth time he had been held up here." We also understand that despite all that happened to "Mr.

Question 1: Where did the robbers push Mr. Hug? **Answer 1:** down the shaft of a freight elevator
Question 2: How old is Mr. Hug? **Answer 2:** 61 years old
Question 3: On what street is Mr. Hug's store located? **Answer 3:** 340 Livingston Street, on the corder of Nevins Street
Question 4: How far is his house to work? **Answer 4:** About 30 minutes train ride
Question 5: How long did the whole robbery take? **Answer 5:** Probably a few minutes
Question 6: Was he trapped in the elevator car, or under? **Answer 6:** under
Question 7: Was Mr. Hug conscious after the robbers left? **Answer 7:** Yes, he cried out and his cries were heard.
Question 8: How many floors does Mr. Hug's store have? **Answer 8:** More than one, since he has an elevator

Table 1: Natural language questions about the story in Figure 1.

Hug," he likely goes back to work after treatment because similar incidents have happened in the past. Machines don't really make these connections (for now!).

Challenges in NLU don't end here; there are many other aspects to language understanding that we skip here since they go beyond the scope of this thesis.

1.3. Measuring the progress towards NLU via Question Answering

To measure machines' ability to understand a given text, one can create numerous questions about the story. A system that better understands language should have a higher chance of answering these questions. This approach has been a popular way of measuring NLU since its early days (McCarthy, 1976; Winograd, 1972; Lehnert, 1977).

Table 1 shows examples of such questions. Consider **Question 1**. The answer to this question is directly mentioned in text and the only thing that needs to be done is creating a representation to handle the variability of text. For instance, a reoresentation of the meaning that are conveyed by *verb* predicates, since a major portion of meanings are centered around verbs. For example, to understand the various elements around a verb "push," one has to

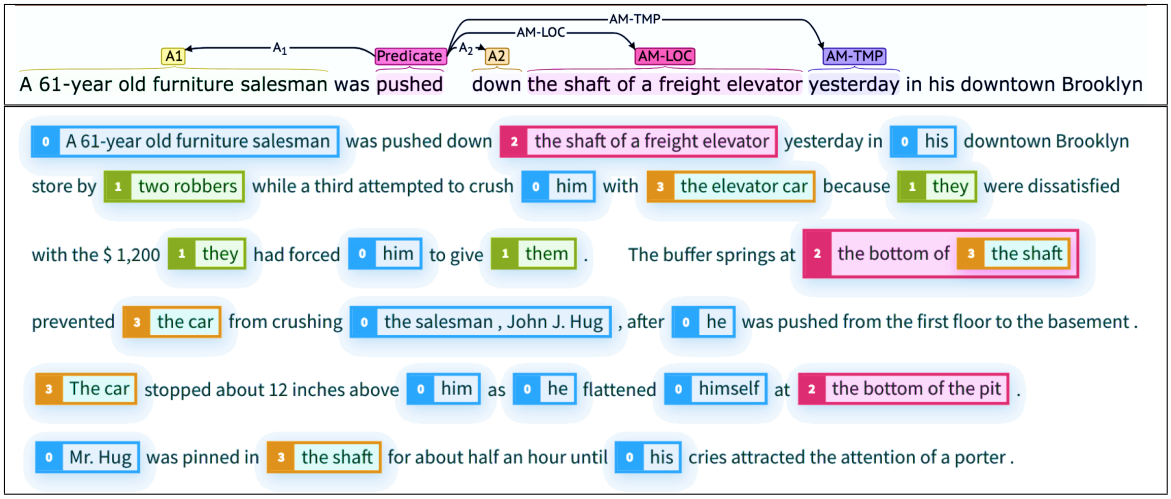


Figure 3: Visualization of two semantic tasks for the given story in Figure 1. Top figure shows *verb semantic roles*; bottom figure shows clusters of *coreferred* mentions. The visualizations use CogCompNLP (Khashabi et al., 2018c) and AllenNLP (Gardner et al., 2018).

figure out *who pushed, who was pushed, pushed where*, etc. The subtask of **semantic role labeling** (Punyakanok et al., 2004) is dedicated to resolving such inferences (Figure 3; top). The output of this annotation of indicates that *the location pushed to* is "the shaft of a freight elevator." In addition, the output of the **coreference** task (Carbonell and Brown, 1988; McCarthy, 1995) informs computers about such equivalences between the mentions of the main character of the story (namely, the equivalence between "Mr. Hug" and "A 61-year-old furniture salesman").

Similarly the answers to **Question 2 and 3** are directly included in the paragraph, although they both require some intermediate processing like the coreference task. The system we introduce in Chapter 4 uses such representations (coreference, semantic roles, etc) and in principle should be able to answer such questions. The dataset introduced in Chapter 6 also motivates addressing questions that require chaining information from multiple pieces of text. In a similar vein, Chapter 8 takes a theoretical perspective on the limits of chaining information.

The rest of the questions in Table 1 are harder for machines, as they require information

beyond what is directly mentioned in the paragraph. For example, **Question 4** requires knowledge of the distance between "Queens" and "Brooklyn," which can be looked up on the internet. Similarly, **Question 5** requires information beyond text; however, it is unlikely to be looked up easily on the web. Understanding that "the robbery" took only a few minutes (and not hours or days) is part of our common sense understanding. The dataset that we introduce in Chapter 7 motivates addressing such understanding (temporal common sense). **Question 6 and 7** require different forms of common sense understanding, beyond the scope of this thesis.

In this thesis we focus on the task of Question Answering (QA), aiming to progress towards NLU. And for this goal, we study various representations and reasoning algorithms. In summary, this thesis is centered around the following statement:

Thesis Statement. *Progress in automated question answering could be facilitated by incorporating the ability to reason over natural language abstractions and world knowledge. More challenging, yet realistic QA datasets pose problems to current technologies; hence, more opportunities for improvement.*

1.4. Thesis outline

In the thesis we use QA as a medium to tackle a few important challenges in the context of NLU. We start with an in-depth review of past work and its connections to our work in Chapter 2. The main content of the thesis is organized as follows (see also Figure 4):

- Part 1: *Reasoning-Driven QA System Design*
  - Chapter 3 discusses TABLEILP, a model for abductive reasoning over natural language questions, with internal knowledge available in tabular representation.
  - Chapter 4 presents SEMANTICILP, an extension of the system in the previous chapter to function on raw text knowledge.

Category	Sub-category	Chapter 3	Chapter 4	Chapter 5	Chapter 6	Chapter 7	Chapter 8
Contribution type	*system design*	✓	✓	✓			
	dataset				✓	✓	
	theory						✓
Challenges addressed	*Ambiguity (grounding)*		✓				✓
	Variability		✓	✓	✓	✓	✓
	Combining information	✓	✓		✓		✓
	Common-sense understanding	✓			✓	✓	

Figure 4: An overview of the contributions and challenges addressed in each chapter of this thesis.

  - Chapter 5 studies the notion of *essential question terms* with the goal of making QA solvers more robust to distractions and irrelevant information.

- Part 2: *Moving the Peaks Higher: More Challenging QA datasets*

  - Chapter 5 presents MultiRC, a reading comprehension challenge which requires combining information from multiple sentences.

  - Chapter 6 presents TacoQA, a reading comprehension challenge which requires the ability to resolve temporal common sense.

- Part 3: *Formal Study of Reasoning in Natural Language*

  - Chapter 7 presents a formalism, in an effort to provide theoretical grounds to the existing intuitions on the limits and possibilities in reasoning, in the context of natural language.

CHAPTER 2 : Background and Related Work

"Whoever wishes to foresee the future must consult the past."

— Nicolo Machiavelli, 1469-1527

2.1. Overview

In this chapter, we review the related literature that addresses different aspects of *natural language understanding.*

Before anything else, we define the terminology (Section 2.2). We divide the discussion into multiple interrelated axes: Section 2.3 discusses various evaluation protocols and datasets introduced in the field. We then provide an overview of the field from the perspective of *knowledge representation and abstraction* in Section 2.4. Building on the discussion of representation, we provide a survey of *reasoning* algorithms, in Section 2.5. We end the chapter with a short section on the technical background necessary for the forthcoming chapters (Section 2.6).

To put everything into perspective, we show a summary of the highlights of the field in Figure 5. Each highlight is color-coded to indicate its contribution type. In the following sections, we go over a select few of these works and explain the evolution of the field, especially those directly related to the focus of this work.

2.2. Terminology

Before starting our main conversation, we define the terminology we will be using throughout this document.

- Propositions are *judgments* or *opinions* which can be true or false. A proposition is not necessarily a sentence, although a sentence can express a proposition (e.g., "cats cannot fly").
- A concept is either a physical entity (like a *tree, bicycle, etc*) or an abstract idea (like

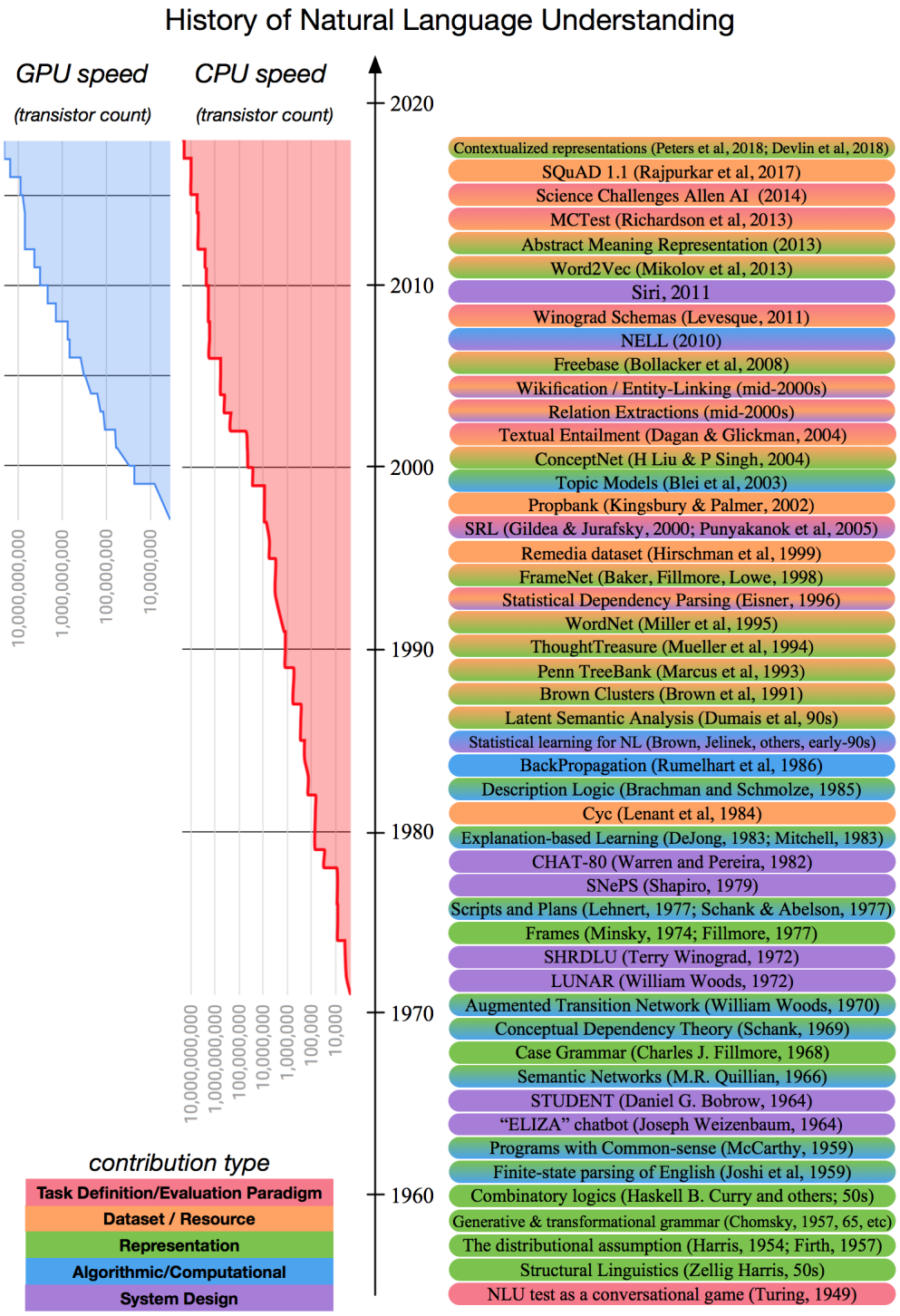


Figure 5: Major highlights of NLU in the past 50 years (within the AI community). For each work, its contribution-type is color-coded. To provide perspective about the role of the computational resources available at each period, we show the progress of CPU/GPU hardware over time.

happiness, thought, betrayal, etc).

- a belief is an expression of faith and/or trust in the truthfulness of a proposition. We also use *confidence* or *likelihood* to refer to the same notion.

- Knowledge is information, facts, understanding acquired through experience or education. The discussion on the philosophical nature of knowledge and its various forms is studied under *epistemology* (Steup, 2014).

- Representation is a medium through which knowledge is provided to a system. For example, the number 5 could be represented as the string "5", as bits `101`, or Roman numeral "V", etc.

- Abstraction defines the level of granularity in a representation. For example, the mentions "New York City", "Buenos Aires", "Maragheh" could all be *abstracted* as `city`.

- Knowledge acquisition is the process of identifying and acquiring the relevant knowledge, according to the representation.

- Reasoning is the process of drawing a conclusion based on the given information. We sometimes refer to this process as *decision-making* or *inference*.

2.3. Measuring the progress towards NLU

2.3.1. Measurement protocols

Evaluation protocols are critical in incentivizing the field to solve the right problems. One of the earliest proposals is due to Alan Turing: if you had a pen-pal for years, you would not know whether you're corresponding to a human or a machine (Turing, 1950; Harnad, 1992). A major limitation of this test (and many of its extensions) is that it is "expensive" to compute (Hernandez-Orallo, 2000; French, 2000).

The protocol we are focusing on in this work is through answering natural language ques-

tions; if an actor (human or computer) understands a given text, it should be able to answer any questions about it. Throughout this thesis, we will refer to this protocol as **Question Answering** (QA). This has been used in the field for many years (McCarthy, 1976; Winograd, 1972; Lehnert, 1977). There are few other terms popularized in the community to refer the same task we are solving here. The phrase **Reading Comprehension** is borrowed from standardized tests (SAT, TOEFL, etc.), usually refers to the scenario where a paragraph is attached to the given question. Another similar phrase is **Machine Comprehension**. Throughout this thesis, we use these phrases interchangeably to refer to the same task.

To make it more formal, for an assumed scenario described by a paragraph P, a system f equipped with NLU should be able to answer any questions Q about the given paragraph P. One can measure the expected performance of the system on a set of questions $\mathcal{D}$, via some distance measure $d(.,.)$ between the predicted answers $f(Q;P)$ and the correct answers $f^*(Q;P)$ (usually a prediction agreed upon by multiple humans):

$$\mathfrak{R}(f;\mathcal{D}) = \mathbb{E}_{(Q,P)\sim\mathcal{D}}\Big[d\Big(f\left(Q;P\right), f^*\left(Q;P\right)\Big)\Big]$$

A critical question here is the choice of question set $\mathcal{D}$ so that $\mathfrak{R}(f;\mathcal{D})$ is an effective measure of f's progress towards NLU. Denote the set of all the possible English questions as $\mathcal{D}_u$. This is an enormous set and, in practice it is unlikely that we could write them all in one place. Instead, it might be more practical to sample from this set. In practice, this sampling is replaced with *static* datasets. This introduces a problem: datasets are hardly a uniform subset of $\mathcal{D}_u$; instead, they are heavily skewed towards more simplicity.

Figure 6 depicts a hypothetical high-dimensional manifold of all the natural language questions in terms of an arbitrary representation (bytes, characters, etc.) Unfortunately, datasets are usually biased samples of the universal set $\mathcal{D}_u$. And they are often biased towards simplicity. This issue makes the dataset design of extra importance since performance results

on a single set might not be a true representative of our progress. Two chapters of this work are dedicated to the construction of QA datasets.

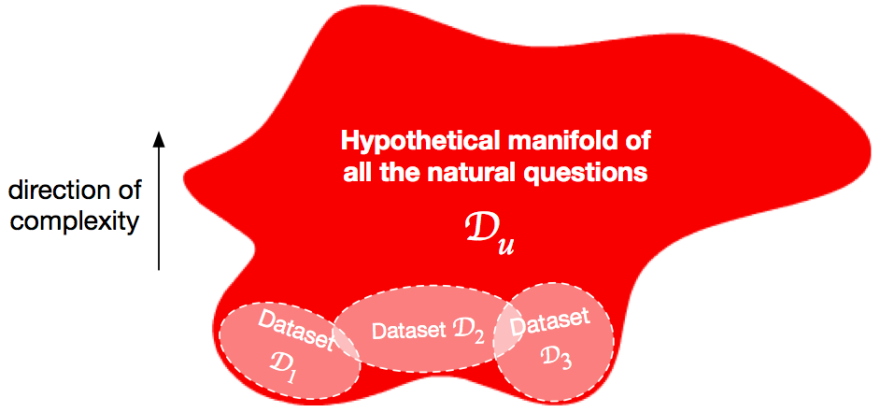


Figure 6: A hypothetical manifold of all the NLU instances. Static datasets make it easy to evaluate our progress but since they usually give a biased estimate, they limit the scope of the challenge.

There are few flavors of QA in terms of their answer representations (see Table 2): (i) questions with multiple candidate-answers, a subset of which are correct; (ii) extractive questions, where the correct answer is a substring of a given paragraph; (iii) Direct-answer questions; a hypothetical system has to *generate* a string for such questions. The choice of answer-representation has direct consequences for the representational richness of the dataset and ease of evaluation. The first two settings (multiple-choice and extractive questions) are easy to evaluate but restrict the richness of the dataset. Direct-answer questions can result in richer datasets but are more expensive to evaluate.

Datasets make it possible to automate the evaluation of the progress towards NLU and be able to compare systems to each other on fixed problems sets. One of the earliest NLU datasets published in the field is the Remedia dataset (Hirschman et al., 1999) which contains short-stories written in simple language for kids provided by Remedia Publications. Each story has 5 types of questions (*who, when, why, where, what*). Since then, there has been many suggestions as to what kind of question-answering dataset is a better test of NLU. Brachman et al. (2005) suggests SAT exams as a challenge for AI. Davis (2014) proposes

Multiple-choice	Dirk Diggler was born as Steven Samuel Adams on April 15, 1961 outside of Saint Paul, Minnesota. His parents were a construction worker and a boutique shop owner who attended church every Sunday and believed in God. Looking for a career as a male model, Diggler dropped out of school at age 16 and left home. He was discovered at a falafel stand by Jack Horner. Diggler met his friend, Reed Rothchild, through Horner in 1979 while working on a film. **Question:** How old was Dirk when he met his friend Reed? **Answers:** *(A) 18 (B) 16 (C) 22
Extractive	The city developed around the Roman settlement Pons Aelius and was named after the castle built in 1080 by Robert Curthose, William the Conqueror's eldest son. The city grew as an important centre for the wool trade in the 14th century, and later became a major coal mining area. The port developed in the 16th century and, along with the shipyards lower down the River Tyne, was amongst the world's largest shipbuilding and ship-repairing centres. **Question:** Who built a castle in Newcastle in 1080? **Answers:** "Robert Curthose"
Direct-answer	**Question:** Some birds fly south in the fall. This seasonal adaptation is known as migration. Explain why these birds migrate. **Answers:** "A(n) bird can migrate, which helps cope with lack of food resources in harsh cold conditions by getting it to a warmer habitat with more food resources."

Table 2: Various answer representation paradigms in QA systems; examples selected from Khashabi et al. (2018a); Rajpurkar et al. (2016); Clark et al. (2016).

multiple-choice challenge sets that are easy for children but difficult for computers. In a similar spirit, Clark and Etzioni (2016) advocate elementary-school science tests. Many science questions have answers that are not explicitly stated in text and instead, require combining information together. In Chapter 2, 3 we use elementary-school science tests as our target challenge.

While the field has produced many datasets in the past few years, many of these datasets are either too restricted in terms of their linguistic richness or they contain annotation biases (Gururangan et al., 2018; Poliak et al., 2018). For many of these datasets, it has been pointed out that many of the high-performing models neither need to 'comprehend' in order to correctly predict an answer, nor learn to 'reason' in a way that generalizes across datasets (Chen et al., 2016; Jia and Liang, 2017; Kaushik and Lipton, 2018). In Section 3.4.4 we show that adversarially-selected candidate-answers result in a significant drop in performance of a few state-of-art science QA systems. To address these weaknesses,

in Chapter 4, 5 we propose two new challenge datasets which, we believe, pose better challenges for systems.

A closely related task is the task of Recognizing Textual Entailment (RTE) (Khashabi et al., 2018c; Dagan et al., 2013), as QA can be cast as entailment (Does P entail $Q + A$? (Bentivogli et al., 2008)). While we do not directly address this task, in some cases we use it as a component within out proposed QA systems (in Chapter 3 and 4).

2.4. Knowledge Representation and Abstraction for NLU

The discussion of knowledge representation has been with AI since its beginning and it is central to the progress of language understanding. Since directly dealing with the raw input/output complicates the reasoning stage, historically researchers have preferred to devise a middleman between the raw information and the reasoning engine. Therefore, the need for an intermediate level seems to be essential. In addition, in many problems, there is a significant amount of knowledge that is not mentioned directly, but rather implied from the context. Somehow the extra information has to be provided to the reasoning system. As a result, the discussion goes beyond just creating formalism for information, and also includes issues like, how to *acquire*, *encode* and *access* it. The issue of representations applies to both *input* level information and the *internal* knowledge of a reasoning system. We refer to some of the relevant debates in the forthcoming sections.

2.4.1. Early Works: "Neats vs Scruffies"[1]

An early trend emerged as the family of symbolic and logical representations, such as propositional and 1st-order logic (McCarthy, 1963). This approach has deep roots in *philosophy* and *mathematical logic*, where the theories have evolved since Aristotle's time. Logic, provided a general purpose, clean and uniform language, both in terms of representations and reasoning.

[1]Terms originally made by Roger Schank to characterize two different camps: the first group that represented commonsense knowledge in the form of large amorphous semantic networks, as opposed to another from the camp of whose work was based on logic and formal extensions of logic.

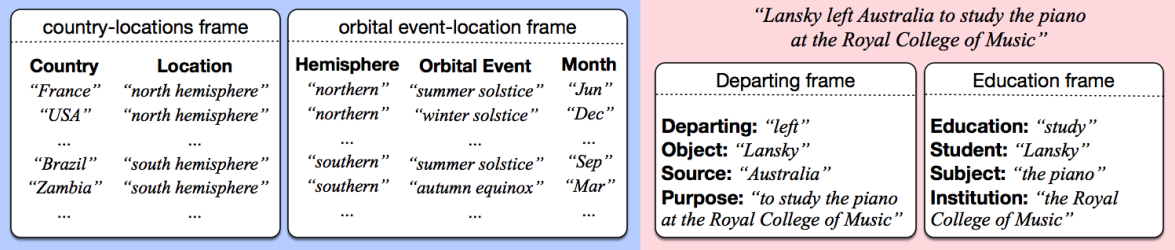


Figure 7: Example frames used in this work. Generic basic science frames (left), used in Chapter 3; event frames with values filled with the given sentence (right), used in Chapter 4.

The other closely-related school of thought evolved from *linguistics* and *psychology*. This trend was less concerned with mathematical rigor, but more concerned with richer psychological and linguistic motivations. For example, *semantic networks* (Quillan, 1966), a network of concepts and links, was based on the idea that memory consists of associations between mental entities. In Chapter 8 we study a formalism for reasoning with such graph-like representations. *Scripts* and *plans* are representational tools to model frequently encountered activities; e.g., going to a restaurant (Schank and Abelson, 1975; Lehnert, 1977). Minsky and Fillmore, separately and in parallel, advocated *frame*-based representations (Minsky, 1974; Fillmore, 1977). The following decades, these approaches have evolved into fine-grained representations and hybrid systems for specific problems. One of the first NLU programs was the STUDENT program of Bobrow (1964), written in LISP (McCarthy and Levin, 1965), which could read and solve high school algebra problems expressed in natural language.

Intuitively, a *frame* induces a grouping of concepts and creates abstract hierarchies among them. For example, *"Monday", "Tuesday", ...* are distinct concepts, but all members of the same conceptual frame. A frame consists of a group of slots and fillers to define a stereotypical object or activity. A slot can contain values such as rules, facts, images, video, procedures or even another frame (Fikes and Kehler, 1985). Frames can be organized hierarchically, where the default values can be inherited the value directly from parent frames. This is part of our underlying representation in Chapter 3, where the reasoning is done over tables of information (an example in Figure 7, left). Decades later after

its proposal, the frame-based approach resulted in resources like FrameNet (Baker et al., 1998), or tasks like Semantic Role Labeling (Gildea and Jurafsky, 2002; Palmer et al., 2005; Punyakanok et al., 2004). This forms the basis for some of the key representations we use in Chapter 4 (see Figure 7, right).

2.4.2. Connectionism

There is another important trend inspired by the apparent brain function emergent from interconnected networks of neural units (Rosenblatt, 1958). It lost many of its fans after Minsky and Papert (1969) showed representational limitations of shallow networks in approximating few functions. However, a series of events reinvigorated this thread: Notably, Rumelhart et al. (1988) found a formalized way to train networks with more than one layer (nowadays known as Back-Propagation algorithm). This work emphasized the *parallel* and *distributed* nature of information processing and gave rise to the "connectionism" movement. Around the same time (Funahashi, 1989) showed the universal approximation property for feed-forward networks (any continuous function on the real numbers can be uniformly approximated by neural networks). Over the past decade, this school has enjoyed newfound excitement by effectively exploiting parallel processing power of GPUs and harnessing large datasets to show progress on certain datasets.

2.4.3. Unsupervised representations

Unsupervised representations are one of the areas that have shown tangible impacts across the board. A pioneering work is Brown et al. (1992) which creates binary term representations based on co-occurrence information. Over the years, a wide variety of such representations emerged; using Wikipedia concepts (Gabrilovich and Markovitch, 2007), word co-occurrences (Turney and Pantel, 2010), co-occurrence factorization (Mikolov et al., 2013; Levy and Goldberg, 2014; Pennington et al., 2014; Li et al., 2015), and using context-sensitive representation (Peters et al., 2018; Devlin et al., 2018). In particular, the latter two are inspired by the connectionist frameworks in the 80s and have shown to be effective

across a wide range of NLU tasks. In this thesis we use unsupervised representations in various ways: In Chapter 3 and 4 we use such representations for phrasal semantic equivalence within reasoning modules. In Chapter 5 we use as features of our supervised system. In Chapter 6, 7 we create NLU systems based on such representations in order to create baselines for the datasets we introduce.

A more recent highlight along this path is the emergence of new unsupervised representations that have been shown to capture many interesting associations in freely available data (Peters et al., 2018; Devlin et al., 2018).

2.4.4. Grounding of meanings

A tightly related issue to the abstraction issue is *grounding* natural language surface information to their actual meaning (Harnad, 1990), as discussed in Section 1.2. Practitioners often address this challenging by enriching their representations; for example by mapping textual information to Wikipedia entries (Mihalcea and Csomai, 2007). In Chapter 4 we use the disambiguation of semantic actions and their roles (Punyakanok et al., 2004; Dang and Palmer, 2005). Chapter 8 of this thesis provides a formalism that incorporates elements of the symbol-grounding problem and shed theoretical light on existing empirical intuitions.

2.4.5. Common sense and implied meanings

A major portion of our language understanding is only implied in language and not explicitly mention (examples in Section 1.2). This difficulty of this challenge has historically been under-estimated. Early AI, during the sixties and onward, experienced a lot of interest in modeling common sense knowledge. McCarthy, one of the founders of AI, believed in formal logic as a solution to common sense reasoning (McCarthy and Lifschitz, 1990). Minsky (1988) estimated that "... commonsense is knowing maybe 30 or 60 million things about the world and having them represented so that when something happens, you can make analogies with others". There have been decade-long efforts to create knowledge bases of common sense information, such as Cyc (Lenat, 1995) and ConceptNet (Liu and Singh,

2004), but none of these have yielded any major impact so far. A roadblock in progress towards such goals is the lack of *natural end-tasks* that can provide an objective measure of progress in the field. To facilitate research in this direction, in Chapter 6 we provide a new natural language QA dataset that performing well on it requires significant progress on multiple *temporal* common sense tasks.

2.4.6. Abstractions of the representations

Abstraction of information is one of the key issues in any effort towards an effective representation. Having coarser abstraction could result in better generalization. However, too much abstraction could result in losing potentially-useful details. In general, there is a trade-off between the expressive level of the representation and the reasoning complexity. We also deal with this issue in multiple ways: (i) we use unsupervised representations that have been shown to indirectly capture abstractions (Mahabal et al., 2018). (ii) we use systems pre-trained with annotations that abstract over raw text; for example, in Chapter 4 we use semantic roles representations of sentences, which abstract over low-level words and map the argument into their high-level thematic roles.

For a given problem instance, how does a system internally choose the right level of abstraction? The human attention structure is extremely good in abstracting concepts (Johnson and Proctor, 2004; Janzen and Vicente, 1997), although automating this is an open question. One way of dealing with such issues is to use multiple levels of abstraction and let the reasoning algorithm use the right level of abstraction when available (Rasmussen, 1985; Bisantz and Vicente, 1994). In Chapter 4, we take a similar approach by using a collection of different abstractions.

2.5. Reasoning/Decision-making Paradigms for NLU

2.5.1. Early formalisms of reasoning

The idea of automated reasoning dates back before AI itself and can be traced to ancient Greece. Aristotle's syllogisms paved the way for *deductive* reasoning formalism. It continued its way with philosophers like Al-Kindi, Al-Farabi, and Avicenna (Davidson, 1992), before culminating as the modern mathematics and logic.

Within AI research, McCarthy (1963) pioneered the use of logic for automating reasoning for language problems, which over time branched into other classes of reasoning (Holland et al., 1989; Evans et al., 1993).

A closely related reasoning to what we study here is *abduction* (Peirce, 1883; Hobbs et al., 1993), which is the process of finding *the best minimal explanation* from a set of observations (see Figure 8). Unlike in deductive reasoning, in abductive reasoning the premises do not guarantee the conclusion. Informally speaking, abduction is inferring cause from effect (reverse direction from deductive reasoning). The two reasoning systems in Chapter 3 and 4 can be interpreted as abductive systems.

We define the notation to make the exposition slightly more formal. Let $\vdash$ denote entailment and $\perp$ denote contradiction. Formally, (logical) abductive reasoning is defined as follows:

> Given background knowledge B and observations O, find a hypothesis H, such that $B \cup H \nvdash \perp$ (consistency with the given background) and $B \cup H \vdash O$ (explaining the observations).

In practical settings, this purely logical definition has many limitations: (a) There could be multiple hypotheses H that explain a particular set of observations given the background knowledge. The best hypothesis has to be selected based on some measure of goodness and the simplicity of the hypothesis (Occam's Razor). (b) Real life has many uncertain

Figure 8: Brief definitions for popular reasoning classes and their examples.

elements, i.e. there are degrees of certainties (rather than binary assignments) associated with observations and background knowledge. Hence the decision of consistency and explainability has to be done with respect to this fuzzy measure. (c) The inference problem in its general form is computationally intractable; often assumptions have to be made to have tractable inference (e.g., restricting the representation to Horn clauses).

2.5.2. Incorporating "uncertainty" in reasoning

Over the years, a wide variety of *soft* alternatives have emerged for reasoning algorithms, by incorporating uncertainty into symbolic models. This resulted in theories like fuzzy-logic (Zadeh, 1975), or probabilistic Bayesian networks (Pearl, 1988; Dechter, 2013), soft abduction (Hobbs et al., 1988; Selman and Levesque, 1990; Poole, 1990). In Bayesian networks, the (uncertain) background knowledge is encoded in a graphical structure and upon receiving observations, the probabilistic explanation is derived by maximizing a posterior probability distribution. These models are essentially based on propositional logic and cannot handle quantifiers (Kate and Mooney, 2009). Weighted abduction combines the weights of relevance/plausibility with first-order logic rules (Hobbs et al., 1988). However, unlike probability theoretic frameworks, their weighting scheme does not have any solid theoret-

ical basis and does not lend itself to a complete probabilistic analysis. Our framework in Chapter 3,4 is also a way to perform abductive reasoning under uncertainty. Our proposal is different from the previous models in a few ways: (i) Unlike Bayesian network our framework is not limited to propositional rules; in fact, there are first-order relations used in the design of TABLEILP (more details in Chapter 3). (ii) unlike many other previous works, we do not make representational assumptions to make the inference simpler (like limiting to Horn clauses, or certain independence assumptions). In fact, the inference might be NP-hard, but with the existence of industrial ILP solvers this is not an issue in practice. Our work is inspired by a prior line of work on inference on structured representations to reason on (and with) language; see Chang et al. (2008, 2010); **?**, 2012), among others.

2.5.3. Macro-reading vs micro-reading

With increased availability of information (especially through the internet) **macro-reading** systems have emerged with the aim of leveraging a large variety of resources and exploiting the redundancy of information (Mitchell et al., 2009). Even if a system does not understand one text, there might be many other texts that convey a similar meaning. Such systems derive significant leverage from relatively shallow statistical methods with surprisingly strong performance (Clark et al., 2016). Today's Internet search engines, for instance, can successfully retrieve *factoid* style answers to many natural language queries by efficiently searching the Web. Information Retrieval (IR) systems work under the assumption that answers to many questions of interest are often explicitly stated somewhere (Kwok et al., 2001), and all one needs, in principle, is access to a sufficiently large corpus. Similarly, statistical correlation based methods, such as those using Pointwise Mutual Information or PMI (Church and Hanks, 1989), work under the assumption that many questions can be answered by looking for words that tend to co-occur with the question words in a large corpus. While both of these approaches help identify correct answers, they are not suitable for questions requiring language understanding and reasoning, such as chaining together multiple facts in order to arrive at a conclusion. On the other hand, **micro-reading** aims at *understanding*

a piece of evidence given to the system, without reliance of *redundancy*. The focus of this thesis is *micro-reading* as it directly addresses NLU; that being said, whenever possible, we use macro-reading systems as our baselines.

2.5.4. Reasoning on "structured" representations

With increasing knowledge resources and diversity of the available knowledge representations, numerous QA systems are developed to operate over large-scale *explicit* knowledge representations. These approaches perform reasoning over *structured* (discrete) abstractions. For instance, Chang et al. (2010) address RTE (and other tasks) via inference on structured representations), Banarescu et al. (2013) use AMR annotators (Wang et al., 2015), Unger et al. (2012) use RDF knowledge (Yang et al., 2017), Zettlemoyer and Collins (2005); Clarke et al. (2010); Goldwasser and Roth (2014); Krishnamurthy et al. (2016) use semantic parsers to answer a given question, and Do et al. (2011, 2012) employ constrained inference for temporal/causal reasoning. The framework we study in Chapter 3 is a reasoning algorithm functioning over tabular knowledge (frames) of basic science concepts.

An important limitation of IR-based systems is their inability to connect distant pieces of information together. However, many other realistic domains (such as science questions or biology articles) have answers that are not explicitly stated in text, and instead require combining facts together. Khot et al. (2017) creates an inference system capable of combining Open IE tuples (Banko et al., 2007). Jansen et al. (2017) propose reasoning by aggregating sentential information from multiple knowledge bases. Socher et al. (2013); McCallum et al. (2017) propose frameworks for chaining relations to infer new (unseen) relations. Our work in Chapter 3 creates chaining of information over multiple tables. The reasoning framework in Chapter 4 investigates reasoning over multiple peaces of raw text. The QA dataset in Chapter 5 we propose also encourages the use of information from different segments of the story. Chapter 8 proposes a formalism to study limits of chaining long-range information.

2.5.5. Models utilizing massive annotated data

A highlight over the past two decades is the advent of statistical techniques into NLP (Hirschman et al., 1999). Since then, a wide variety of supervised-learning algorithms have shown strong performances on different datasets.

The increasingly large amount of data available for recent benchmarks make it possible to train neural models (see "Connectionism"; Section 2.4.2) (Seo et al., 2016; Parikh et al., 2016; Wang et al., 2018; Liu et al., 2018; Hu et al., 2018). Moreover, an additional technical shift was using distributional representation of words (word vectors or embeddings) extracted from large-scale text corpora (Mikolov et al., 2013; Pennington et al., 2014) (see Section 2.4.3).

Despite all the decade-long excited about supervised-learning algorithms, the main progress, especially in the past few years, has mostly been due to the re-emergence of *unsupervised* representations (Peters et al., 2018; Devlin et al., 2018).[2]

2.6. Technical background and notation

In this section, we provide the relevant mathematical background used throughout this thesis. We cover three main areas used widely across this document.

2.6.1. Complexity theory

We follow the standard notation for asymptotic comparison of functions: $O(.), o(.), \Theta(.), \Omega(.)$, and $\omega(.)$ (Cormen et al., 2009).

We use P and NP to refer to the basic complexity classes. We briefly review these classes: P consists of all problems that can be solved efficiently (in polynomial time). NP (non-deterministic polynomial time) includes all problems that given a solution, one can efficiently *verify* the solution. When a problem is called *intractable*, it refers to its complexity class

[2] *Unsupervised* in the sense that they are constructed with freely available data, as opposed to task-specific annotated data.

being at least *NP*-hard.

2.6.2. Probability Theory

$X \sim f(\theta)$ denotes a random variable X distributed according to probability distribution $f(\theta)$, paramterized by θ. The mean and variance of X are denoted as $\mathbb{E}_{X \sim f(\theta)}[X]$ and $\mathbb{V}[X]$, resp. $\text{Bern}(p)$ and $\text{Bin}(n, p)$ denote the Bernoulli and Binomial distributions, resp.

2.6.3. Graph theory

We denote an undirected graph with $G(V, E)$ where V and E are the sets of nodes and edges, resp. We use the notations V_G and E_G to refer to the nodes and edges of a graph G, respectively.

A *subgraph* of a graph G is another graph formed from a subset of the vertices and edges of G. The vertex subset must include all endpoints of the edge subset, but may also include additional vertices.

A *cut* $C = (S, T)$ in G is a partition of the nodes V into subsets S and T. The *size* of the cut C is the number of edges in E with one endpoint in S and the other in T.

2.6.4. Optimization Theory

As it is widely known an ILP can be written as the following:

$$\begin{aligned} &\text{maximize} && \mathbf{w}^{\mathrm{T}}\mathbf{x} && (2.1)\\ &\text{subject to} && A\mathbf{x} \leq \mathbf{b}, && (2.2)\\ &\text{and} && \mathbf{x} \in \mathbb{Z}^n. && (2.3) \end{aligned}$$

We first introduce the basic variables, and define the full definition of the ILP program: define the weights in the objective function ($\mathbf{w}$ in Equation 2.1), and the constraints (A and $\mathbf{b}$ in Equation 2.2).

This formulation is incredibly powerful and has been used for many problems. In the context of NLP problems, ILP based discrete optimization was introduced by Roth and Yih (2004) and has been successfully used (Chang et al., 2010; Berant et al., 2010; Srikumar and Roth, 2011; Goldwasser and Roth, 2014). In Chapter 3 and 4 also, we formalize our desired behavior as an optimization problem.

This optimization problem with integrality constraint and its general form, is an NP-hard problem. That being said, the industrial solvers (which use cutting-plane and other heuristics) are quite fast across a wide variety of problems.

Part I

Reasoning-Driven System Design

CHAPTER 3 : QA as Subgraph Optimization on Tabular Knowledge

"The techniques of artificial intelligence are to the mind what bureaucracy is to human social interaction."

— Terry Winograd, Thinking Machines: Can there be? 1991

3.1. Overview

Consider a question from the NY Regents 4th Grade Science Test:[1]

> In New York State, the longest period of daylight occurs during which month?
> (A) June (B) March (C) December (D) September

We would like a QA system that, even if the answer is not explicitly stated in a document, can *combine basic scientific and geographic facts* to answer the question, e.g., New York is in the north hemisphere; the longest day occurs during the summer solstice; and the summer solstice in the north hemisphere occurs in June (hence the answer is June). Figure 9 illustrates how our system approaches this, with the highlighted support graph representing its line of reasoning.

Further, we would like the system to be *robust under simple perturbations*, such as changing New York to New Zealand (in the southern hemisphere) or changing an incorrect answer option to an irrelevant word such as "last" that happens to have high co-occurrence

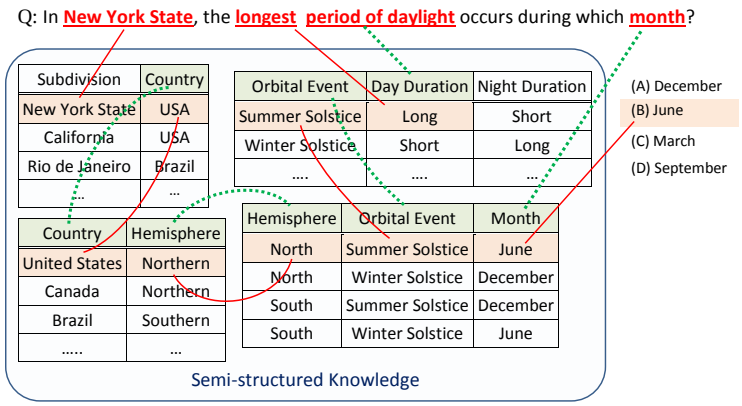


Figure 9: TABLEILP searches for the best support graph (chains of reasoning) connecting the question to an answer, in this case June. Constraints on the graph define what constitutes valid support and how to score it (Section 3.3.3).

[1]This chapter is based on the following publication: Khashabi et al. (2016).

with the question text.

To this end, we propose a structured reasoning system, called TableILP, that operates over a semi-structured knowledge base derived from text and answers questions by chaining multiple pieces of information and combining parallel evidence.[2] The knowledge base consists of *tables*, each of which is a collection of instances of an n-ary relation defined over natural language phrases. E.g., as illustrated in Figure 9, a simple table with schema *(country, hemisphere)* might contain the instance *(United States, Northern)* while a ternary table with schema *(hemisphere, orbital event, month)* might contain *(North, Summer Solstice, June)*. TableILP treats lexical constituents of the question Q, as well as cells of potentially relevant tables T, as nodes in a large graph $\mathcal{G}_{Q,T}$, and attempts to find a subgraph G of $\mathcal{G}_{Q,T}$ that "best" supports an answer option. The notion of best support is captured via a number of structural and semantic constraints and preferences, which are conveniently expressed in the Integer Linear Programming (ILP) formalism. We then use an off-the-shelf ILP optimization engine called SCIP (Achterberg, 2009) to determine the best supported answer for Q.

Following a recently proposed AI challenge (Clark, 2015), we evaluate TableILP on unseen elementary-school science questions from standardized tests. Specifically, we consider a challenge set (Clark et al., 2016) consisting of all non-diagram multiple choice questions from 6 years of NY Regents 4th grade science exams. In contrast to a state-of-the-art structured inference method (Khot et al., 2015) for this task, which used Markov Logic Networks (MLNs) (Richardson and Domingos, 2006), TableILP achieves a significantly (+14% absolute) higher test score. This suggests that a combination of a rich and fine-grained constraint language, namely ILP, even with a publicly available solver is more effective in practice than various MLN formulations of the task. Further, while the scalability of the MLN formulations was limited to very few (typically one or two) selected science

[2] A preliminary version of our ILP model was used in the ensemble solver of Clark et al. (2016). We build upon this earlier ILP formulation, providing further details and incorporating additional syntactic and semantic constraints that improve the score by 17.7%.

rules at a time, our approach easily scales to hundreds of relevant scientific facts. It also complements the kind of questions amenable to IR and PMI techniques, as is evidenced by the fact that a combination (trained using simple Logistic Regression (Clark et al., 2016)) of TableILP with IR and PMI results in a significant (+10% absolute) boost in the score compared to IR alone.

Our ablation study suggests that combining facts from multiple tables or multiple rows within a table plays an important role in TableILP's performance. We also show that TableILP benefits from the table structure, by comparing it with an IR system using the same knowledge (the table rows) but expressed as simple sentences; TableILP scores significantly (+10%) higher. Finally, we demonstrate that our approach is robust to a simple perturbation of incorrect answer options: while the simple perturbation results in a relative drop of 20% and 33% in the performance of IR and PMI methods, respectively, it affects TableILP's performance by only 12%.

3.2. Related Work

In this section, we provide additional related work, and augment our review related work provided in Section 2.1.

Clark et al. (2016) proposed an ensemble approach for the science QA task, demonstrating the effectiveness of a combination of information retrieval, statistical association, rule-based reasoning, and an ILP solver operating on semi-structured knowledge. Our ILP system extends their model with additional constraints and preferences (e.g., semantic relation matching), substantially improving QA performance.

A number of systems have been developed for answering factoid questions with short answers (e.g., "What is the capital of France?") using document collections or databases (e.g., Freebase (Bollacker et al., 2008), NELL (Carlson et al., 2010)), for example (Brill et al., 2002; Fader et al., 2014; Ferrucci et al., 2010; Ko et al., 2007; t. Yih et al., 2014; Yao and Durme, 2014; Zou et al., 2014). However, many science questions have answers that are not

explicitly stated in text, and instead require combining information together. Conversely, while there are AI systems for formal scientific reasoning (e.g., (Gunning et al., 2010; Novak, 1977)), they require questions to be posed in logic or restricted English. Our goal here is a system that operates between these two extremes, able to combine information while still operating with natural language.

There is a relatively rich literature in the databases community, on executing different commands on the tablular content (e.g., searching, joining, etc) via a user commands issued by a semi-novice user Talukdar et al. (2008, 2010). A major distinguishing perspective is that in our problem the queries are generated completely independent of the the table content. However, in a database system application, a user is at-least partially informed of the common keywords, could observe the outputs of the queries and adjust the commands accordingly.

3.3. QA as Subgraph Optimization

We begin with our knowledge representation formalism, followed by our treatment of QA as an optimal subgraph selection problem over such knowledge, and then briefly describe our ILP model for subgraph selection.

3.3.1. Semi-Structured Knowledge as Tables

We use semi-structured knowledge represented in the form of n-ary predicates over natural language text (Clark et al., 2016). Formally, a k-column table in the knowledge base is a predicate $r(x_1, x_2, \ldots, x_k)$ over strings, where each string is a (typically short) natural language phrase. The column headers capture the table schema, akin to a relational database. Each row in the table corresponds to an instance of this predicate. For example, a simple country-hemisphere table represents the binary predicate $r_{\text{ctry-hems}}(c, h)$ with instances such as (Australia, Southern) and (Canada, Northern). Since table content is specified in natural language, the same entity is often represented differently in different tables, posing an additional inference challenge.

thAlthough techniques for constructing this knowledge base are outside the scope of this paper, we briefly mention them. Tables were constructed using a mixture of manual and semi-automatic techniques. First, the table schemas were manually defined based on the syllabus, study guides, and training questions. Tables were then populated both manually and semi-automatically using IKE (Dalvi et al., 2016), a table-building tool that performs interactive, bootstrapped relation extraction over a corpus of science text. In addition, to augment these tables with the broad knowledge present in study guides that doesn't always fit the manually defined table schemas, we ran an Open IE (Banko et al., 2007) pattern-based *subject-verb-object* (SVO) extractor from Clark et al. (2014) over several science texts to populate three-column Open IE tables. Methods for further automating table construction are under development.

3.3.2. QA as a Search for Desirable Support Graphs

We treat question answering as the task of pairing the question with an answer such that this pair has the best support in the knowledge base, measured in terms of the strength of a "support graph" defined as follows.

Given a multiple choice question Q and tables T, we can define a labeled undirected graph $\mathcal{G}_{Q,T}$ over nodes $\mathcal{V}$ and edges $\mathcal{E}$ as follows. We first split Q into lexical constituents (e.g., non-stopword tokens, or chunks) $\mathbf{q} = \{q_\ell\}$ and answer options $\mathbf{a} = \{a_m\}$. For each table T_i, we consider its cells $\mathbf{t} = \{t_{ijk}\}$ as well as column headers $\mathbf{h} = \{h_{ik}\}$. The nodes of $\mathcal{G}_{Q,T}$ are then $\mathcal{V} = \mathbf{q} \cup \mathbf{a} \cup \mathbf{t} \cup \mathbf{h}$. For presentation purposes, we will equate a graph node with the lexical entity it represents (such as a table cell or a question constituent). The undirected edges of $\mathcal{G}_{Q,T}$ are $\mathcal{E} = ((\mathbf{q} \cup \mathbf{a}) \times (\mathbf{t} \cup \mathbf{h})) \cup (\mathbf{t} \times \mathbf{t}) \cup (\mathbf{h} \times \mathbf{h})$ excluding edges both whose endpoints are within a single table.

Informally, an edge denotes (soft) equality between a question or answer node and a table node, or between two table nodes. To account for lexical variability (e.g., that *tool* and *instrument* are essentially equivalent) and generalization (e.g., that a *dog* is an *animal*), we

replace string equality with a phrase-level entailment or similarity function $w : \mathcal{E} \rightarrow [0, 1]$ that labels each edge $e \in \mathcal{E}$ with an associated score $w(e)$. We use entailment scores (directional) from $\mathbf{q}$ to $\mathbf{t} \cup \mathbf{h}$ and from $\mathbf{t} \cup \mathbf{h}$ to $\mathbf{a}$, and similarity scores (symmetric) between two nodes in $\mathbf{t}$.[3] In the special case of column headers across two tables, the score is (manually) set to either 0 or 1, indicating whether this corresponds to a meaningful join.

Intuitively, we would like the support graph for an answer option to be connected, and to include nodes from the question, the answer option, and at least one table. Since each table row represents a coherent piece of information but cells within a row do not have any edges in $\mathcal{G}_{Q,T}$ (the same holds also for cells and the corresponding column headers), we use the notion of an augmented subgraph to capture the underlying table structure. Let $G = (V, E)$ be a subgraph of $\mathcal{G}_{Q,T}$. The *augmented subgraph* G^+ is formed by adding to G edges (v_1, v_2) such that v_1 and v_2 are in V and they correspond to either the same row (possibly the header row) of a table in T or to a cell and the corresponding column header.

Definition 1. A *support graph* $G = G(Q, T, a_m)$ for a question Q, tables T, and an answer option a_m is a subgraph (V, E) of $\mathcal{G}_{Q,T}$ with the following basic properties:

1. $V \cap \mathbf{a} = \{a_m\},\ V \cap \mathbf{q} \neq \phi,\ V \cap \mathbf{t} \neq \phi$;
2. $w(e) > 0$ for all $e \in E$;
3. if $e \in E \cap (\mathbf{t} \times \mathbf{t})$ then there exists a corresponding $e' \in E \cap (\mathbf{h} \times \mathbf{h})$ involving the same columns; and
4. the augmented subgraph G^+ is connected.

A support graph thus connects the question constituents to a unique answer option through table cells and (optionally) table headers corresponding to the aligned cells. A given question and tables give rise to a large number of possible support graphs, and the role of the inference

[3] In our evaluations, w for entailment is a simple WordNet-based (Miller, 1995) function that computes the best word-to-word alignment between phrases, scores these alignments using WordNet's hypernym and synonym relations normalized using relevant word-sense frequency, and returns the weighted sum of the scores. w for similarity is the maximum of the entailment score in both directions. Alternative definitions for these functions may also be used.

process will be to choose the "best" one under a notion of *desirable* support graphs developed next. We do this through a number of additional structural and semantic properties; the more properties the support graph satisfies, the more desirable it is.

3.3.3. ILP Formulation

We model the above support graph search for QA as an ILP optimization problem, i.e., as maximizing a linear objective function over a finite set of variables, subject to a set of linear inequality constraints (see Section 2.6.4 for a premier on ILP formulation). A summary of the model is given below.[4]

We note that the ILP objective and constraints aren't tied to the particular domain of evaluation; they represent general properties that capture what constitutes a well supported answer for a given question.

Table 3 summarizes the notation for various elements of the problem, such as t_{ijk} for cell (j, k) of table i. All core variables in the ILP model are binary, i.e., have domain $\{0, 1\}$. For each element, the model has a unary variable capturing whether this element is part of the support graph G, i.e., it is "active". For instance, row r_{ij} is active if at least one cell in row j of table i is in G. The model also has pairwise "alignment" variables, capturing edges of $\mathcal{G}_{Q,T}$. The alignment variable for an edge e in $\mathcal{G}_{Q,T}$ is associated with the corresponding weight $w(e)$, and captures whether e is included in G. To improve efficiency, we create a pairwise variable for e only if $w(e)$ is larger than a certain threshold. These unary and pairwise variables are then used to define various types of constraints and preferences, as discussed next.

Element	Description
T_i	table i
h_{ik}	header of the k-th column of i-th table
t_{ijk}	cell in row j and column k of i-th table
r_{ij}	row j of i-th table
ℓ_{ik}	column k of i-th table
q_ℓ	ℓ-th lexical constituent of the question Q
a_m	m-th answer option

Table 3: Notation for the ILP formulation.

[4]Details of the ILP model may be found in Appendix A.1.1.

To make the definitions clear, we introduce the variables used in our optimization, which we will use later to define constraints explicitly. We define variables over each element by overloading $x(.)$ or $y(.,.)$ notation to refer to a binary variable on a single elements or their pair, respectively. Table 4 contains the complete list of the variables, all of which are binary, i.e. they are defined on $\{0,1\}$ domain. The unary variables represent presence of a specific element in the support graph as a node. For example $x(T_i) = 1$ if and only if the table T_i is active. Similarly basic variables are defined between pairs of elements; e.g., $y(t_{ijk}, q_\ell)$ is a binary variable that takes value 1 if and only if the corresponding edge is present in the support graph, which can alternatively be referred to as an *alignment* between cell (j,k) of table i and the ℓ-th constituent of the question.

As previously mentioned, in practice we do not create all possible pairwise variables. Instead we choose the pairs which have the alignment score $w(e)$ exceeding a threshold. For example we create the pairwise variables $y\left(t_{ijk}, t_{i'j'k'}\right)$ only if the score $w(t_{ijk}, t_{i'j'k'}) \geq$ MinCellCellAlignment. [5]

The objective function is a weighted linear sum of all the variables we instantiate for a given problem. [6] There is a small set of auxiliary variables defined for linearizing complicated constraints, which will later introduce among constraints.

Constraints are a significant part of our model,

Basic Pairwise Activity Variables	
$y(t_{ijk}, t_{ij'k'})$	cell to cell
$y(t_{ijk}, q_\ell)$	cell to question constituent
$y(h_{ik}, q_\ell)$	header to question constituent
$y(t_{ijk}, a_m)$	cell to answer option
$y(h_{ik}, a_m)$	header to answer option
$y(\ell_{ik}, a_m)$	column to answer option
$y(T_i, a_m)$	table to answer option
$y(\ell_{ik}, \ell_{ik'})$	column to column relation
High-level Unary Variables	
$x(T_i)$	active table
$x(r_{ij})$	active row
$x(\ell_{ik})$	active column
$x(h_{ik})$	active column header
$x(q_\ell)$	active question constituent
$x(a_m)$	active answer option

Table 4: Variables used for defining the optimization problem for TableILP solver. All variables have domain $\{0,1\}$.

[5] An exhaustive list of the minimum alignment thresholds for creating pairwise variables is in Table 28 in the appendix.

[6] The complete list of weights for the pairwise and unary variables are included in Table 27 in the appendix.

which impose our desired behavior on the support graph. However due to lack of space we only show a representative subset here. [7]

Some constraints relate variables to each other. The unary variables are defined through constraints that relate them to the pairwise basic variables. For example, for active row variable $x(T_i)$, we ensure that it is active if and only if any cell in row j is active:

$$x(r_{ij}) \geq y(t_{ijk}, *), \forall (t_{ijk}, *) \in \mathcal{R}_{ij}, \forall i, j, k,$$

where $\mathcal{R}_{ij}$ is collection of pairwise variables with one end in row j of table i.

In what follows we outline the some of the important behaviors we expect from our model which come out with different combination of the active variables.

Basic Lookup

Consider the following question:

> Which characteristic helps a fox find food? (A) sense of smell (B) thick fur (C) long tail (D) pointed teeth

In order to answer such lookup-style questions, we generally seek a row with the highest aggregate alignment to question constituents. We achieve this by incorporating the question-table alignment variables with the alignment scores, $w(e)$, as coefficients and the active question constituents variable with a constant coefficient in the objective function. Since any additional question-table edge with a positive entailment score (even to irrelevant tables) in the support graph would result in an increase in the score, we disallow tables with alignments only to the question (or only to a choice) and add a small penalty for every table used in order to reduce noise in the support graph. We also limit the maximum number of alignments of a question constituent and table cells to prevent one constituent or cell from

[7] The complete list of the constraints is explained in Table 31 in the appendix.

having a large influence on the objective function and thereby the solution:

$$\sum_{(*,q_\ell)\in\mathcal{Q}_l} y\left(*, q_\ell\right) \leq \textsc{MaxAlignmentsPerQCons}, \forall l$$

where $\mathcal{Q}_l$ is the set of all pairwise variables with one end in question constituent ℓ.

Parallel Evidence

For certain questions, evidence needs to be combined from multiple rows of a table. For example,

> Sleet, rain, snow, and hail are forms of (A) erosion (B) evaporation (C) groundwater (D) precipitation

To answer this question, we need to combine evidence from multiple table entries from the weather terms table, *(term, type)*, namely (sleet, precipitation), (rain, precipitation), (snow, precipitation), and (hail, precipitation). To achieve this, we allow multiple active rows in the support graph. Similar to the basic constraints, we limit the maximum number of active rows per table and add a penalty for every active row to ensure only relevant rows are considered for reasoning:

$$\sum_{j} x\left(r_{ij}\right) \leq \textsc{MaxRowsPerTable}, \forall i$$

To encourage only coherent parallel evidence within a single table, we limit our support graph to always use the same columns across multiple rows within a table, i.e., every active row has the active cells corresponding to the same set of columns.

Evidence Chaining

Questions requiring chaining of evidence from multiple tables, such as the example in Figure 9, are typically the most challenging in this domain. Chaining can be viewed as per-

forming a *join* between two tables. We introduce alignments between cells across columns in pairs of tables to allow for chaining of evidence. To help minimize potential noise introduced by chaining irrelevant facts, we add a penalty for every inter-table alignment and also rely on the 0/1 weights of header-to-header edges to ensure only semantically meaningful table joins are considered.

Semantic Relation Matching

Our constraints so far have only looked at the content of the table cells, or the structure of the support graph, without explicitly considering the *semantics* of the table schema. By using alignments between the question and column headers (i.e., type information), we exploit the table schema to prefer alignments to columns relevant to the "topic" of the question. In particular, for questions of the form "which X ...", we prefer answers that directly entail X or are connected to cells that entail X. However, this is not sufficient for questions such as:

> What is one way to change water from a liquid to a solid? (A) decrease the temperature (B) increase the temperature (C) decrease the mass (D) increase the mass

Even if we select the correct table, say $r_{\text{change-init-fin}}(c, i, f)$ that describes the initial and final states for a phase change event, both choice (A) and choice (B) would have the exact same score in the presence of table rows (increase temperature, solid, liquid) and (decrease temperature, liquid, solid). The table, however, does have the initial vs. final state structure. To capture this semantic structure, we annotate pairs of columns within certain tables with the semantic relationship present between them. In this example, we would annotate the phase change table with the relations: changeFrom(c, i), changeTo(c, f), and fromTo(i, f).

Given such semantic relations for table schemas, we can now impose a preference towards question-table alignments that respect these relations. We associate each semantic relation with a set of linguistic patterns describing how it might be expressed in natural language. TableILP then uses these patterns to spot possible mentions of the relations in the question

Q. We then add the soft constraint that for every pair of active columns in a table (with an annotated semantic relation) aligned to a pair of question constituents, there should be a valid expression of that relation in Q between those constituents. In our example, we would match the relation fromTo(liquid, solid) in the table to "liquid to a solid" in the question via the pattern "X to a Y" associated with fromTo(X,Y), and thereby prefer aligning with the correct row (decrease temperature, liquid, solid).

3.4. Evaluation

We compare our approach to three existing methods, demonstrating that it outperforms the best previous structured approach (Khot et al., 2015) and produces a statistically significant improvement when used in combination with IR-based methods (Clark et al., 2016). For evaluations, we use a 2-core 2.5 GHz Amazon EC2 linux machine with 16 GB RAM.

Question Set. We use the same question set as Clark et al. (2016), which consists of all non-diagram multiple-choice questions from 12 years of the NY Regents 4th Grade Science exams.[8] The set is split into 108 development questions and 129 hidden test questions based on the year they appeared in (6 years each). All numbers reported below are for the hidden test set, except for question perturbation experiments which relied on the 108 development questions.

Test scores are reported as percentages. For each question, a solver gets a score of 1 if it chooses the correct answer and $1/k$ if it reports a k-way tie that includes the correct answer. On the 129 test questions, a score difference of 9% (or 7%) is statistically significant at the 95% (or 90%, resp.) confidence interval based on the binomial exact test (Howell, 2012).

Corpora. We work with three knowledge corpora:

1. Web Corpus: This corpus contains 5×10^{10} tokens (280 GB of plain text) extracted from Web pages. It was collected by Charles Clarke at the University of Waterloo,

[8] These are the only publicly available state-level science exams. http://www.nysedregents.org/Grade4/Science/home.html

and has been used previously by Turney (2013) and Clark et al. (2016). We use it here to compute statistical co-occurrence values for the PMI solver.

2. Sentence Corpus (Clark et al., 2016): This includes sentences from the Web corpus above, as well as around 80,000 sentences from various domain-targeted sources for elementary science: a Regents study guide, CK12 textbooks (www.ck12.org), and web sentences with similar content as the course material.

3. Table Corpus (cf. Section 3.3.1): This includes 65 tables totaling around 5,000 rows, designed based on the development set and study guides, as well as 4 Open IE-style (Banko et al., 2007) automatically generated tables totaling around 2,600 rows.[9]

3.4.1. Solvers

TableILP (our approach). Given a question Q, we select the top 7 tables from the Table Corpus using the the standard TF-IDF score of Q with tables treated as bag-of-words documents. For each selected table, we choose the 20 rows that overlap with Q the most. This filtering improves efficiency and reduces noise. We then generate an ILP and solve it using the open source SCIP engine (Achterberg, 2009), returning the active answer option a_m from the optimal solution. To check for ties, we disable a_m, re-solve the ILP, and compare the score of the second-best answer, if any, with that of a_m.

MLN Solver (structured inference baseline). We consider the current state-of-the-art structured reasoning method developed for this specific task by Khot et al. (2015). We compare against their best performing system, namely Praline, which uses Markov Logic Networks (Richardson and Domingos, 2006) to (a) align lexical elements of the question with probabilistic first-order science rules and (b) to control inference. We use the entire set of 47,000 science rules from their original work, which were also derived from same domain-targeted sources as the ones used in our Sentence Corpus.

[9]Table Corpus and the ILP model are available at allenai.org.

IR Solver (information retrieval baseline). We use the IR baseline by Clark et al. (2016), which selects the answer option that has the best matching sentence in a corpus. Specifically, for each answer option a_i, the IR solver sends $q + a_i$ as a query to a search engine (we use Lucene) on the Sentence Corpus, and returns the search engine's score for the top retrieved sentence s, where s must have at least one non-stopword overlap with q, and at least one with a_i. The option with the highest Lucene score is returned as the answer.

PMI Solver (statistical co-occurrence baseline). We use the PMI-based approach by Clark et al. (2016), which selects the answer option that most frequently co-occurs with the question words in a corpus. Specifically, it extracts unigrams, bigrams, trigrams, and skip-bigrams from the question and each answer option. For a pair (x, y) of n-grams, their pointwise mutual information (PMI) (Church and Hanks, 1989) in the corpus is defined as $\log \frac{p(x,y)}{p(x)p(y)}$ where $p(x, y)$ is the co-occurrence frequency of x and y (within some window) in the corpus. The solver returns the answer option that has the largest average PMI in the Web Corpus, calculated over all pairs of question n-grams and answer option n-grams.

3.4.2. Results

We first compare the accuracy of our approach against the previous structured (MLN-based) reasoning solver. We also compare against IR(tables), an IR solver using table rows expressed as sentences, thus embodying an unstructured approach operating on the same knowledge as TableILP.

As Table 5 shows, among the two structured inference approaches, TableILP outperforms the MLN baseline by 14%. The preliminary ILP system reported by Clark et al. (2016) achieves only a score of 43.8% on this question set. Further, given the same semi-structured knowledge (i.e., the Table Corpus), TableILP is substantially (+10%) better at exploiting the structure than the IR(tables)

Solver	Test Score (%)
MLN	47.5
IR(tables)	51.2
TableILP	**61.5**

Table 5: TableILP significantly outperforms both the prior MLN reasoner, and IR using identical knowledge as TableILP

baseline, which, as mentioned above, uses the same data expressed as sentences.

Complementary Strengths

While their overall score is similar, TableILP and IR-based methods clearly approach QA very differently. To assess whether TableILP adds any new capabilities, we considered the 50 (out of 129) questions incorrectly answered by PMI solver (ignoring tied scores). On these unseen but arguably more difficult questions, TableILP answered 27 questions correctly, achieving a score of 54% compared to the random chance of 25% for 4-way multiple-choice questions. Results with IR solver were similar: TableILP scored 24.75 on the 52 questions incorrectly answered by IR (i.e., 47.6% accuracy).

Solver	Test Score (%)
IR	58.5
PMI	60.7
TableILP	61.5
TableILP + IR	66.1
TableILP + PMI	67.6
TableILP + IR+ PMI	**69.0**

Table 6: Solver combination results

This analysis highlights the complementary strengths of these solvers. Following Clark et al. (2016), we create an ensemble of TableILP, IR, and PMI solvers, combining their answer predictions using a simple Logistic Regression model trained on the development set. This model uses 4 features derived from each solver's score for each answer option, and 11 features derived from TableILP's support graphs. [10] Table 6 shows the results, with the final combination at 69% representing a significant improvement over individual solvers.

ILP Solution Properties

Table 7 summarizes various ILP and support graph statistics for TableILP, averaged across all test questions.

[10] Details of the 11 features may be found in the Appendix B.

The optimization model has around 50 high-level constraints, which result, on average, in around 4000 inequalities over 1000 variables. Model creation, which includes computing pairwise entailment scores using WordNet, takes 1.9 seconds on average per question, and the resulting ILP is solved by the SCIP engine in 2.1 seconds (total for all four options), using around 1,300 LP iterations for each option.[11]

Thus, TableILP takes only 4 seconds to answer a question using multiple rows across multiple tables (typically 140 rows in total), as compared to 17 seconds needed by the MLN solver for reasoning with four rules (one per answer option).

Category	Quantity	Average
ILP complexity	#variables	1043.8
	#constraints	4417.8
	#LP iterations	1348.9
Knowledge use	#rows	2.3
	#tables	1.3
Timing stats	model creation	1.9 sec
	solving the ILP	2.1 sec

Table 7: TableILP statistics averaged across questions

While the final support graph on this question set relies mostly on a single table to answer the question, it generally combines information from more than two rows (2.3 on average) for reasoning. This suggests parallel evidence is more frequently used on this dataset than evidence chaining.

3.4.3. Ablation Study

To quantify the importance of various components of our system, we performed several ablation experiments, summarized in Table 8 and described next.

Solver	Test Score (%)
TableILP	61.5
No Multiple Row Inference	51.0
No Relation Matching	55.6
No Open IE Tables	52.3
No Lexical Entailment	50.5

Table 8: Ablation results for TableILP

No Multiple Row Inference: We modify the ILP constraints to limit inference to a single row (and hence a single table),

[11] Commercial ILP solvers (e.g., CPLEX, Gurobi) are much faster than the open-source SCIP solver we used for evaluations.

thereby disallowing parallel evidence and evidence chaining (Section 3.3.3). This drops the performance by 10.5%, highlighting the importance of being able to combine evidence from multiple rows (which would correspond to multiple sentences in a corpus) from one or more tables.

No Relation matching: To assess the importance of considering the semantics of the table, we remove the requirement of matching the semantic relation present between columns of a table with its lexicalization in the question (Section 3.3.3). The 6% drop indicates TABLEILP relies strongly on the table semantics to ensure creating meaningful inferential chains.

No Open IE tables: To evaluate the impact of relatively unstructured knowledge from a large corpus, we removed the tables containing Open IE extractions (Section 3.3.2). The 9% drop in the score shows that this knowledge is important and TABLEILP is able to exploit it even though it has a very simple triple structure. This opens up the possibility of extending our approach to triples extracted from larger knowledge bases.

No Lexical Entailment: Finally, we test the effect of changing the alignment metric w (Section 3.3.2) from WordNet based scores to a simple asymmetric word-overlap measured as $score(T, H) = \frac{|T \cap H|}{|H|}$. Relying on just word-matching results in an 11% drop, which is consistent with our knowledge often being defined in terms of generalities.

3.4.4. Question Perturbation

One desirable property of QA systems is robustness to simple variations of a question, *especially when a variation would make the question arguably easier for humans.*

To assess this, we consider a simple, automated way to perturb each 4-way multiple-choice question: (1) query Microsoft's Bing search engine (www.bing.com) with the question text and obtain the text snippet of the top 2,000 hits; (2) create a list of strings by chunking and tokenizing the results; (3) remove stop words and special characters, as well as any words

(or their lemma) appearing in the question; (4) sort the remaining strings based on their frequency; and (5) replace the three incorrect answer options in the question with the most frequently occurring strings, thereby generating a new question. For instance:

> In New York State, the longest period of daylight occurs during which month?
> (A) *eastern* (B) June (C) *history* (D) *years*

As in this example, the perturbations (italicized) are often not even of the correct "type", typically making them much easier for humans. They, however, still remain difficult for solvers.

Solver	Original Score (%)	% Drop with Perturbation	
		absolute	relative
IR	70.7	13.8	19.5
PMI	73.6	24.4	33.2
TableILP	85.0	**10.5**	**12.3**

Table 9: Drop in solver scores (on the development set, rather than the hidden test set) when questions are perturbed

For each of the 108 development questions, we generate 10 new perturbed questions, using the 30 most frequently occurring words in step (5) above. While this approach can introduce new answer options that should be considered correct as well, only 3% of the questions in a random sample exhibited this behavior. Table 9 shows the performance of various solvers on the resulting 1,080 perturbed questions. As one might expect, the PMI approach suffers the most at a 33% relative drop. TableILP's score drops as well (since answer type matching isn't perfect), but only by 12%, attesting to its higher resilience to simple question variation.

3.5. Summary and Discussion

This chapter proposed a reasoning system for question answering on elementary-school science exams, using a semi-structured knowledge base. We formulate QA as an Integer Linear Program (ILP), that answers natural language questions using a semi-structured knowledge base derived from text, including questions requiring multi-step inference and a combination of multiple facts. On a dataset of real, unseen science questions, our system significantly outperforms (+14%) the best previous attempt at structured reasoning for this task, which

used Markov Logic Networks (MLNs). When combined with unstructured inference methods, the ILP system significantly boosts overall performance (+10%). Finally, we show our approach is substantially more robust to a simple answer perturbation compared to statistical correlation methods.

There are a few factors that limit the ideas discussed in this chapter. In particular, the knowledge consumed by this system are in the form of curated tables; constructing such knowledge is not always easy. In addition, not everything might be representable in that form. Another limitation stems from the nature of multi-step reasoning: larger number of reasoning steps could result in more brittle decisions. We study this issue in Chapter 8.

CHAPTER 4 : QA as Subgraph Optimization over Semantic Abstractions

"It linked all the perplexed meanings
Into one perfect peace."

— Procter and Sullivan, The Lost Chord, 1877

4.1. Overview

In this chapter, we consider the multiple-choice setting where Q is a question, A is a set of answer candidates, and the knowledge required for answering Q is available in the form of raw text P.[1] A major difference here with the previous chapter is that, the knowledge given a system is *raw-text*, instead of of being represented in tabular format.

We demonstrate that we can use existing NLP modules, such as semantic role labeling (SRL) systems with respect to multiple predicate types (verbs, prepositions, nominals, etc.), to derive multiple semantic views of the text and perform reasoning over these views to answer a variety of questions.

As an example, consider the following snippet of sports news text and an associated question:

> *P: Teams are under pressure after PSG purchased Neymar this season.* ***Chelsea purchased Morata****. The Spaniard looked like he was set for a move to Old Trafford for the majority of the summer only for Manchester United to sign Romelu Lukaku instead, paving the way for Morata to finally move to Chelsea for an initial £56m.*
> *Q: Who did Chelsea purchase this season?*
> *A: {✓ Alvaro Morata, Neymar, Romelu Lukaku }*

Given the bold-faced text P' in P, simple word-matching suffices to correctly answer Q. However, P' could have stated the same information in many different ways. As paraphrases become more complex, they begin to involve more linguistic constructs such as coreference, punctuation, prepositions, and nominals. This makes understanding the text, and thus the QA task, more challenging.

[1] This chapter is based on the following publication: Khashabi et al. (2018b).

For instead, P' could instead say *Morata is the recent* ***acquisition*** *by Chelsea.* This simple looking transformation can be surprisingly confusing for highly successful systems such as BiDAF (Seo et al., 2016), which produces the partially correct phrase *"Neymar this season. Morata"*. On the other hand, one can still answer the question confidently by abstracting relevant parts of Q and P, and connecting them appropriately. Specifically, a verb SRL frame for Q would indicate that we seek the object of the verb *purchase*, a nominal SRL frame for P' would capture that the *acquisition* was of Morata and was done by Chelsea, and textual similarity would align *purchase* with *acquisition*.

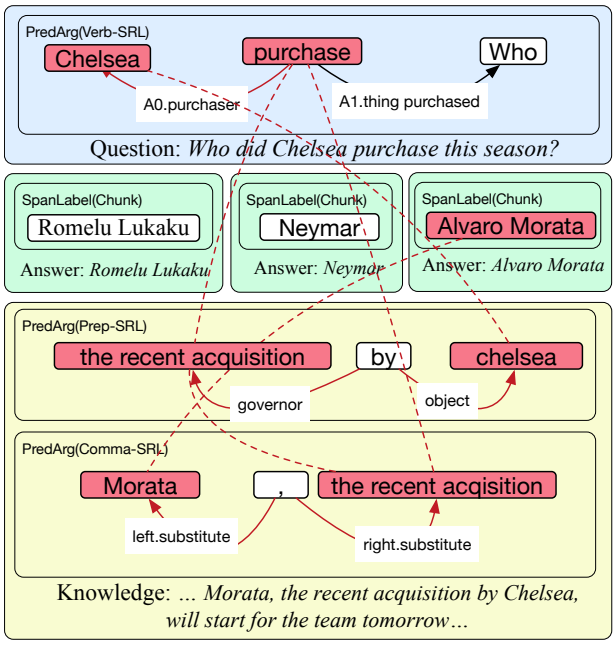


Figure 10: Depiction of SemanticILP reasoning for the example paragraph given in the text. Semantic abstractions of the question, answers, knowledge snippet are shown in different colored boxes (blue, green, and yellow, resp.). Red nodes and edges are the elements that are aligned (used) for supporting the correct answer. There are many other unaligned (unused) annotations associated with each piece of text that are omitted for clarity.

Similarly, suppose P' instead said *Morata,* ***,*** *the recent acquisition* ***by*** *Chelsea,* ***,*** *will start for the team tomorrow.* BiDAF now incorrectly chooses Neymar as the answer, presumably due to its proximity to the words *purchased* and *this season*. However, with the right abstractions, one could still arrive at the correct answer as depicted in Figure 10 for our proposed system, SemanticILP. This reasoning uses comma SRL to realize that the Morata is referring to the *acquisition*, and a preposition SRL frame to capture that the acquisition was done *by* Chelsea.

One can continue to make P' more complex. For example, P' could introduce the need for coreference resolution by phrasing the information as: ***Chelsea*** *is hoping to have a*

great start this season by actively hunting for new players in the transfer period. Morata, the recent acquisition by the team, will start for ***the team*** *tomorrow.* Nevertheless, with appropriate semantic abstractions of the text, the underlying reasoning remains relatively simple.

Given sufficiently large QA training data, one could conceivably perform end-to-end training (e.g., using a deep learning method) to address these linguistic challenges. However, existing large scale QA datasets such as SQuAD (Rajpurkar et al., 2016) often either have a limited linguistic richness or do not necessarily need reasoning to arrive at the answer (Jia and Liang, 2017). Consequently, the resulting models do not transfer easily to other domains. For instance, the above mentioned BiDAF model trained on the SQuAD dataset performs substantially worse than a simple IR approach on our datasets. On the other hand, many of the QA collections in domains that require some form of reasoning, such as the science questions we use, are small (100s to 1000s of questions). This brings into question the viability of the aforementioned paradigm that attempts to learn everything from only the QA training data.

Towards the goal of effective structured reasoning in the presence of data sparsity, we propose to use a rich set of general-purpose, pre-trained NLP tools to create various *semantic abstractions* of the raw text[2] in a domain independent fashion, as illustrated for an example in Figure 10. We represent these semantic abstractions as *families of graphs*, where the family (e.g., trees, clusters, labeled predicate-argument graphs, etc.) is chosen to match the nature of the abstraction (e.g., parse tree, coreference sets, SRL frames, etc., respectively). The collection of semantic graphs is then augmented with inter- and intra-graph edges capturing lexical similarity (e.g., word-overlap score or word2vec distance).

This semantic graph representation allows us to formulate QA as the search for an optimal *support graph*, a subgraph G of the above augmented graph connecting (the semantic graphs of) Q and A via P. The reasoning used to answer the question is captured by a variety of

[2]This applies to all three inputs of the system: Q, A, and P.

requirements or constraints that G must satisfy, as well as a number of desired properties, encapsulating the "correct" reasoning, that makes G preferable over other valid support graphs. For instance, a simple requirement is that G must be connected and it must touch both Q and A. Similarly, if G includes a verb from an SRL frame, it is preferable to also include the corresponding subject. Finally, the resulting constrained optimization problem is formulated as an Integer Linear Program (ILP), and optimized using an off-the-shelf ILP solver (see Section 2.6.4 for a review of ILP).

This formalism may be viewed as a generalization of the systems introduced in the previous chapter: instead of operating over table rows (which are akin to labeled sequence graphs or predicate-argument graphs), we operate over a much richer class of semantic graphs. It can also be viewed as a generalization of the recent TupleInf system (Khot et al., 2017), which converts P into a particular kind of semantic abstraction, namely Open IE tuples (Banko et al., 2007).

This generalization to multiple semantic abstractions poses two key technical challenges: (a) unlike clean knowledge-bases (e.g., Dong et al. (2015)) used in many QA systems, abstractions generated from NLP tools (e.g., SRL) are noisy; and (b) even if perfect, using their output for QA requires delineating what information in Q, A, and P is relevant for a given question, and what constitutes valid reasoning. The latter is especially challenging when combining information from diverse abstractions that, even though grounded in the same raw text, may not perfectly align. We address these challenges via our ILP formulation, by using our linguistic knowledge about the abstractions to design requirements and preferences for linking these abstractions.

We present a new QA system, SemanticILP,[3] based on these ideas, and evaluate it on multiple-choice questions from two domains involving rich linguistic structure and reasoning: elementary and middle-school level science exams, and early-college level biology reading comprehension. Their data sparsity, as we show, limits the performance of state-of-the-art

[3]Code available at: https://github.com/allenai/semanticilp

neural methods such as BiDAF (Seo et al., 2016). SemanticILP, on the other hand, is able to successfully capitalize on existing general-purpose NLP tools in order to outperform existing baselines by 2%-6% on the science exams, leading to a new state of the art. It also generalizes well, as demonstrated by its strong performance on biology questions in the ProcessBank dataset (Berant et al., 2014). Notably, while the best existing system for the latter relies on domain-specific structural annotation and question processing, SemanticILP needs neither.

4.1.1. Related Work

We provide a brief review of the related work, in additional to the discussion provided in Section 2.1.

Our formalism can be seen as an extension of the previous chapter. For instance, in our formalism, each table used by TableILP can be viewed as a semantic frame and represented as a predicate-argument graph. The table-chaining rules used there are equivalent to the reasoning we define when combining two annotation components. Similarly, Open IE tuples used by (Khot et al., 2017) can also be viewed as a predicate-argument structure.

One key abstraction we use is the predicate-argument structure provided by Semantic Role Labeling (SRL). Many SRL systems have been designed (Gildea and Jurafsky, 2002; Punyakanok et al., 2008) using linguistic resources such as FrameNet (Baker et al., 1998), PropBank (Kingsbury and Palmer, 2002), and NomBank (Meyers et al., 2004). These systems are meant to convey high-level information about predicates (which can be a verb, a noun, etc.) and related elements in the text. The meaning of each predicate is conveyed by a frame, the schematic representations of a situation. Phrases with similar semantics ideally map to the same frame and roles. Our system also uses other NLP modules, such as for coreference resolution (Lee et al., 2013) and dependency parsing (Chang et al., 2015).

While it appears simple to use SRL systems for QA (Palmer et al., 2005), this has found limited success (Kaisser and Webber, 2007; Pizzato and Mollá, 2008; Moreda et al., 2011).

The challenges earlier approaches faced were due to making use of VerbSRL only, while QA depends on richer information, not only verb predicates and their arguments, along with some level of brittleness of all NLP systems. Shen and Lapata (2007) have partly addressed the latter challenge with an inference framework that formulates the task as a bipartite matching problem over the assignment of semantic roles, and managed to slightly improve QA. In this work we address both these challenges and go beyond the limitations of using a single predicate SRL system; we make use of SRL abstractions that are based on verbs, nominals, prepositions, and comma predicates, as well as textual similarity. We then develop an inference framework capable of exploiting combinations of these multiple SRL (and other) views, thus operating over a more complete semantic representation of the text.

A key aspect of QA is handling textual variations, on which there has been prior work using dependency-parse transformations (Punyakanok et al., 2004). These approaches often define inference rules, which can generate new trees starting from a base tree. Bar-Haim et al. (2015) and Stern et al. (2012) search over a space of a pre-defined set of text transformations (e.g., coreference substitutions, passive to active). Our work differs in that we consider a much wider range of textual variations by combining multiple abstractions, and make use of a more expressive inference framework.

4.2. Knowledge Abstraction and Representation

We begin with our formalism for abstracting knowledge from text and representing it as a family of graphs, followed by specific instantiations of these abstractions using off-the-shelf NLP modules.

4.2.1. Semantic Abstractions

The pivotal ingredient of the abstraction is raw text. This representation is used for question Q, each answer option A_i and the knowledge snippet P, which potentially contains the answer to the question. The KB for a given raw text, consists of the text it-

self, embellished with various SemanticGraphs attached to it, as depicted in Figure 11.

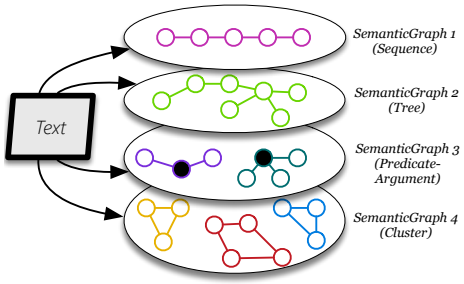


Figure 11: Knowledge Representation used in our formulation. Raw text is associated with a collection of SemanticGraphs, which convey certain information about the text. There are implicit similarity edges among the nodes of the connected components of the graphs, and from nodes to the corresponding raw-text spans.

Each SemanticGraph is representable from a family of graphs. In principle there need not be any constraints on the permitted graph families; however for ease of representation we choose the graphs to belong to one of the 5 following families: Sequence graphs represent labels for each token in the sentence. Span family represents labels for spans of the text. Tree, is a tree representation of text spans. Cluster family, contain spans of text in different groups. PredArg family represents predicates and their arguments; in this view edges represent the connections between each single predicates and its arguments. Each SemanticGraph belongs to one of the graph families and its content is determined by the semantics of the information it represents and the text itself.

We define the knowledge more formally here. For a given paragraph, T, its representation $\mathcal{K}(T)$ consists of a set of semantic graphs $\mathcal{K}(T) = \{g_1, g_2, \ldots\}$. We define $\mathbf{v}(g) = \{c_i\}$ and $\mathbf{e}(g) = \{(c_i, c_j)\}$ to be the set of nodes and edges of a given graph, respectively.

4.2.2. Semantic Graph Generators

Having introduced a graph-based abstraction for knowledge and categorized it into a family of graphs, we now delineate the instantiations we used for each family. Many of the pre-trained extraction tools we use are available in CogCompNLP.[4]

- Sequence or labels for sequence of tokens; for example Lemma and POS (Roth and Zelenko, 1998).

[4]Available at: http://github.com/CogComp/cogcomp-nlp

- Span which can contains labels for spans of text; we instantiated Shallow-Parse (Punyakanok and Roth, 2001), Quantities (Roy et al., 2015), NER (Ratinov and Roth, 2009; Redman et al., 2016)).

- Tree, a tree representation connecting spans of text as nodes; for this we used Dependency of Chang et al. (2015).

- Cluster, or spans of text clustered in groups. An example is Coreference (Lee et al., 2011).

- PredArg; for this view we used Verb-SRL and Nom-SRL(Punyakanok et al., 2008; Roth and Lapata, 2016), Prep-SRL (Srikumar and Roth, 2013), Comma-SRL (Arivazhagan et al., 2016).

Given SemanticGraph generators we have the question, answers and paragraph represented as a collection of graphs. Given the instance graphs, creating augmented graph will be done implicitly as an optimization problem in the next step.

4.3. QA as Reasoning Over Semantic Graphs

We introduce our treatment of QA as an optimal subgraph selection problem over knowledge. We treat question answering as the task of finding the best support in the knowledge snippet, for a given question and answer pair, measured in terms of the strength of a "support graph" defined as follows.

The inputs to the QA system are, a question $\mathcal{K}(Q)$, the set of answers $\{\mathcal{K}(A_i)\}$ and given a knowledge snippet $\mathcal{K}(P)$.[5] Given such representations, we will form a reasoning problem, which is formulated as an optimization problem, searching for a "support graph" that connects the question nodes to a unique answer option through nodes and edges of a snippet.

Define the instance graph $I = I(Q, \{A_i\}, P)$ as the union of knowledge graphs: $I \triangleq \mathcal{K}(Q) \cup (\mathcal{K}(A_i)) \cup \mathcal{K}(P)$. Intuitively, we would like the support graph to be connected, and to include nodes from the question, the answer option, and the knowledge. Since the SemanticGraph

[5]For simplicity, from now on, we drop "knowledge"; e.g., instead of saying "question knowledge", we say "question".

is composed of many disjoint sub-graph, we define *augmented graph* I^+ to model a bigger structure over the instance graphs I. Essentially we *augment* the instance graph and weight the new edges. Define a scoring function $f : (v_1, v_2)$ labels pair of nodes v_1 and v_2 with an score which represents their phrase-level entailment or similarity.

Definition 2. An *augmented graph* I^+, for a question Q, answers $\{A_i\}$ and knowledge P, is defined with the following properties:

1. Nodes: $\mathbf{v}(I^+) = \mathbf{v}(I(Q, \{A_i\}, P))$

2. Edges:[6]

$$\mathbf{e}(I^+) = \mathbf{e}(I) \cup \mathcal{K}(Q) \otimes \mathcal{K}(P) \cup [\cup_i \mathcal{K}(P) \otimes \mathcal{K}(A_i)]$$

3. Edge weights: for any $e \in I^+$:

   - If $e \notin I$, the edge connects two nodes in different connected components:

$$\forall e = (v_1, v_2) \notin I : w(e) = f(v_1, v_2)$$

   - If $e \in I$, the edge belongs to a connected component, and the edge weight information about the reliability of the SemanticGraph and semantics of the two nodes.

$$\forall g \in I, \forall e \in g : w(e) = f'(e, g)$$

Next, we have to define *support graphs*, the set of graphs that support the reasoning of a question. For this we will apply some structured constraints on the augmented graph.

Definition 3. A *support graph* $G = G(Q, \{A_i\}, P)$ for a question Q, answer-options $\{A_i\}$ and paragraph P, is a subgraph (V, E) of I^+ with the following properties:

[6] Define $\mathcal{K}(T_1) \otimes \mathcal{K}(T_2) \triangleq \bigcup_{\substack{(g_1,g_2) \in \\ \mathcal{K}(T_1) \times \mathcal{K}(T_2)}} \mathbf{v}(g_1) \times \mathbf{v}(g_2)$, where $\mathbf{v}(g_1) \times \mathbf{v}(g_2) = \{(v, w); v \in \mathbf{v}(g_1), w \in \mathbf{v}(g_2)\}$.

Sem. Graph	Property
PredArg	Use at least (a) a predicate and its argument, or (b) two arguments
Cluster	Use at least two nodes
Tree	Use two nodes with distance less than k
SpanLabelView	Use at least k nodes

Table 10: Minimum requirements for using each family of graphs. Each graph connected component (e.g. a PredArg frame, or a Coreference chain) cannot be used unless the above-mentioned conditioned is satisfied.

1. G is connected.

2. G has intersection with the question, the knowledge, and exactly one answer candidate:[7]

$$G \cap \mathcal{K}(Q) \neq \emptyset, \quad G \cap \mathcal{K}(P) \neq \emptyset, \quad \exists!\, i : G \cap \mathcal{K}(A_i) \neq \emptyset$$

3. G satisfies structural properties per each connected component, as summarized in Table 10.

Definition 3 characterizes what we call a potential solution to a question. A given question and paragraph give rise to a large number of possible support graphs. We define the space of feasible support graphs as $\mathcal{G}$ (i.e., all the graphs that satisfy Definition 3, for a given $(Q, \{A_i\}, P)$). To rank various feasible support graphs in such a large space, we define a scoring function score(G) as:

$$\sum_{v \in \mathbf{v}(G)} w(v) + \sum_{e \in \mathbf{e}(G)} w(e) - \sum_{c \in \mathcal{C}} w_c \, \mathbf{1}\{c \text{ is violated}\} \tag{4.1}$$

for some set of preferences (or soft-constraints) $\mathcal{C}$. When c is violated, denoted by the indicator function $\mathbf{1}\{c \text{ is violated}\}$ in Eq. (4.1), we penalize the objective value by some fixed amount w_c. The second term is supposed to bring more sparsity to the desired solutions, just like how regularization terms act in machine learning models (Natarajan, 1995). The

[7] $\exists!$ here denotes the uniqueness quantifier, meaning "there exists one and only one".

first term is the sum of weights we defined when constructing the augmented-graph, and is supposed to give more weight to the models that have better and more reliable alignments between its nodes. The role of the inference process will be to choose the "best" one under our notion of *desirable* support graphs:

$$G^* = \arg\max_{G \in \mathcal{G}} \tag{4.2}$$

4.3.1. ILP Formulation

Our QA system, SemanticILP, models the above support graph search of Eq. (4.2) as an ILP optimization problem, i.e., as maximizing a linear objective function over a finite set of variables, subject to a set of linear inequality constraints. A summary of the model is given below.

The augmented graph is not explicitly created; instead, it is implicitly created. The nodes and edges of the augmented graph are encoded as a set of binary variables. The value of the binary variables reflects whether a node or an edge is used in the optimal graph G^*. The properties listed in Table 10 are implemented as weighted linear constraints using the variables defined for the nodes and edges.

As mentioned, edge weights in the augmented graph come from a function, f, which captures (soft) phrasal entailment between question and paragraph nodes, or paragraph and answer nodes, to account for lexical variability. In our evaluations, we use two types of f. (a) Similar to Khashabi et al. (2016), we use a WordNet-based (Miller, 1995) function to score word-to-word alignments, and use this as a building block to compute a phrase-level alignment score as the weighted sum of word-level alignment scores. Word-level scores are computed using WordNet's hypernym and synonym relations, and weighted using relevant word-sense frequency. f for similarity (as opposed to entailment) is taken to be the average of the entailment scores in both directions. (b) For longer phrasal alignments (e.g., when aligning phrasal verbs) we use the Paragram system of Wieting et al. (2015).

- Number of sentences used is more than k - Active edges connected to each chunk of the answer option, more than k - More than k chunks in the active answer-option - More than k edges to each question constituent - Number of active question-terms - If using PredArgof $\mathcal{K}(Q)$, at least an argument should be used - If using PredArg(Verb-SRL) of $\mathcal{K}(Q)$, at least one predicate should be used.

Table 11: The set of preferences functions in the objective.

The final optimization is done on Eq. (4.1). The first part of the objective is the sum of the weights of the sub-graph, which is what an ILP does, since the nodes and edges are modeled as variables in the ILP. The second part of Eq. (4.1) contains a set of preferences $\mathcal{C}$, summarized in Table 11, meant to apply *soft* structural properties that partly dependant on the knowledge instantiation. These preferences are soft in the sense that they are applied with a weight to the overall scoring function (as compare to a hard constraint). For each preference function c there is an associated binary or integer variable with weight w_c, and we create appropriate constraints to simulate the corresponding behavior.

We note that the ILP objective and constraints aren't tied to the particular domain of evaluation; they represent general properties that capture what constitutes a well supported answer for a given question.

4.4. Empirical Evaluation

We evaluate on two domains that differ in the nature of the supporting text (concatenated individual sentences vs. a coherent paragraph), the underlying reasoning, and the way questions are framed. We show that SemanticILP outperforms a variety of baselines, including retrieval-based methods, neural-networks, structured systems, and the current best system for each domain. These datasets and systems are described next, followed by results.

4.4.1. Question Sets

For the first domain, we have a collection of question sets containing elementary-level science questions from standardized tests (Clark et al., 2016; Khot et al., 2017). Specifically, REGENTS 4TH contains all non-diagram multiple choice questions from 6 years of NY Regents 4th grade science exams (127 train questions, 129 test). REGENTS 8TH similarly contains 8th grade questions (147 train, 144 test). The corresponding expanded datasets are AI2PUBLIC 4TH (432 train, 339 test) and AI2PUBLIC 8TH (293 train, 282 test).[8]

For the second domain, we use the PROCESSBANK[9] dataset for the reading comprehension task proposed by Berant et al. (2014). It contains paragraphs about biological processes and two-way multiple choice questions about them. We used a broad subset of this dataset that asks about events or about an argument that depends on another event or argument.[10]. The resulting dataset has 293 train and 109 test questions, based on 147 biology paragraphs.

Test scores are reported as percentages. For each question, a system gets a score of 1 if it chooses the correct answer, $1/k$ if it reports a k-way tie that includes the correct answer, and 0 otherwise.

4.4.2. Question Answering Systems

We consider a variety of baselines, including the best system for each domain.

IR (information retrieval baseline). We use the IR solver from Clark et al. (2016), which selects the answer option that has the best matching sentence in a corpus. The sentence is forced to have a non-stopword overlap with both q and a.

SemanticILP (our approach). Given the input instance (question, answer options, and a paragraph), we invoke various NLP modules to extract semantic graphs. We then generate

[8] AI2 Science Questions V1 at http://data.allenai.org/ai2-science-questions

[9] https://nlp.stanford.edu/software/bioprocess

[10] These are referred to as "dependency questions" by Berant et al. (2014), and cover around 70% of all questions.

Combination	Representation
Comb-1	$\mathcal{K}(Q) = \{\textsf{Shallow-Parse}, \textsf{Tokens}\}$ $\mathcal{K}(P) = \{\textsf{Shallow-Parse}, \textsf{Tokens}, \textsf{Dependency}\}$
Comb-2	$\mathcal{K}(Q) = \{\textsf{Verb-SRL}, \textsf{Shallow-Parse}\}$ $\mathcal{K}(P) = \{\textsf{Verb-SRL}\}$
Comb-3	$\mathcal{K}(Q) = \{\textsf{Verb-SRL}, \textsf{Shallow-Parse}\}$ $\mathcal{K}(P) = \{\textsf{Verb-SRL}, \textsf{Coreference}\}$
Comb-4	$\mathcal{K}(Q) = \{\textsf{Verb-SRL}, \textsf{Shallow-Parse}\}$ $\mathcal{K}(P) = \{\textsf{Comma-SRL}\}$
Comb-5	$\mathcal{K}(Q) = \{\textsf{Verb-SRL}, \textsf{Shallow-Parse}\}$ $\mathcal{K}(P) = \{\textsf{Prep-SRL}\}$

Table 12: The semantic annotator combinations used in our implementation of SemanticILP.

an ILP and solve it using the open source SCIP engine (Achterberg, 2009), returning the active answer option a_m from the optimal solution found. To check for ties, we disable a_m, re-solve the ILP, and compare the score of the second-best answer, if any, with that of the best score.

For the science question sets, where we don't have any paragraphs attached to each question, we create a passage by using the above IR solver to retrieve scored sentences for each answer option and then combining the top 8 unique sentences (across all answer options) to form a paragraph.

While the sub-graph optimization can be done over the entire augmented graph in one shot, our current implementation uses multiple simplified solvers, each performing reasoning over augmented graphs for a commonly occurring annotator combination, as listed in Table 12. For all of these annotator combinations, we let the representation of the answers be $\mathcal{K}(A) = \{\textsf{Shallow-Parse}, \textsf{Tokens}\}$. Importantly, our choice of working with a few annotator combinations is mainly for simplicity of implementation and suffices to demonstrate that reasoning over even just two annotators at a time can be surprisingly powerful. There is no fundamental limitation in implementing SemanticILP using one single optimization problem as stated in Eq. (4.2).

Each simplified solver associated with an annotator combination in Table 12 produces a

confidence score for each answer option. We create an *ensemble* of these solvers as a linear combination of these scores, with weights trained using the union of training data from all questions sets.

BiDAF (neural network baseline). We use the recent deep learning reading comprehension model of Seo et al. (2016), which is one of the top performing systems on the SQuAD dataset and has been shown to generalize to another domain as well (Min et al., 2017). Since BIDAF was designed for fill-in-the-blank style questions, we follow the variation used by Kembhavi et al. (2017) to apply it to our multiple-choice setting. Specifically, we compare the predicted answer span to each answer candidate and report the one with the highest similarity.

We use two variants: the original system, BIDAF, pre-trained on 100,000+ SQuAD questions, as well as an extended version, BIDAF', obtained by performing continuous training to fine-tune the SQuAD-trained parameters using our (smaller) training sets. For the latter, we convert multiple-choice questions into reading comprehension questions by generating all possible text-spans within sentences, with token-length at most *correct answer length + 2*, and choose the ones with the highest similarity score with the correct answer. We use the ALLENNLP re-implementation of BIDAF[11], train it on SQuAD, followed by training it on our dataset. We tried different variations (epochs and learning rates) and selected the model which gives the best average score across all the datasets. As we will see, the variant that was further trained on our data often gives better results.

TupleInf (semi-structured inference baseline). Recently proposed by Khot et al. (2017), this is a state-of-the-art system designed for science questions. It uses Open IE (Banko et al., 2007) tuples derived from the text as the knowledge representation, and performs reasoning over it via an ILP. It has access to a large knowledge base of Open IE tuples, and exploits redundancy to overcome challenges introduced by noise and linguistic variability.

Proread and **SyntProx**. PROREADis a specialized and best performing system on the

[11]Available at: https://github.com/allenai/allennlp

ProcessBank question set. Berant et al. (2014) annotated the training data with events and event relations, and trained a system to extract the process structure. Given a question, Proread converts it into a query (using regular expression patterns and keywords) and executes it on the process structure as the knowledge base. Its reliance on a question-dependent query generator and on a process structure extractor makes it difficult to apply to other domains.

SyntProx is another solver suggested by (Berant et al., 2014). It aligns content word lemmas in both the question and the answer against the paragraph, and select the answer tokens that are closer to the aligned tokens of the questions. The distance is measured using dependency tree edges. To support multiple sentences they connect roots of adjacent sentences with bidirectional edges.

4.4.3. Experimental Results

We evaluate various QA systems on datasets from the two domains. The results are summarized below, followed by some some insights into SemanticILP's behavior and an error analysis.

Science Exams. The results of experimenting on different grades' science exams are summarized in Table 13, which shows the exam scores as a percentage. The table demonstrates that SemanticILP consistently outperforms the best baselines in each case by 2%-6%.

Further, there is no absolute winner among the baselines; while IR is good on the 8th grade questions, TupleInf and BiDAF' are better on 4th grade questions. This highlights the differing nature of questions for different grades.

Biology Exam. The results on the ProcessBank dataset are summarized in Table 14. While SemanticILP's performance is substantially better than most baselines and close to that of Proread, it is important to note that this latter baseline enjoys additional supervision of domain-specific event annotations. This, unlike our other relatively general

Dataset	BiDAF	BiDAF'	IR	TupleInf	SemanticILP
Regents 4th	56.3	53.1	59.3	*61.4*	**67.6**
AI2Public 4th	50.7	*57.4*	54.9	56.1	**59.7**
Regents 8th	53.5	62.8	*64.2*	61.3	**66.0**
AI2Public 8th	47.7	51.9	*52.8*	51.6	**55.9**

Table 13: Science test scores as a percentage. On elementary level science exams, SemanticILP consistently outperforms baselines. In each row, the best score is in **bold** and the best baseline is *italicized*.

baselines, makes it limited to this dataset, which is also why we don't include it in Table 13.

We evaluate IR on this reading comprehension dataset by creating an ElasticSearch index, containing the sentences of the knowledge paragraphs.

ProRead	SyntProx	IR	BiDAF	BiDAF'	SemanticILP
68.1	61.9	63.8	58.7	61.3	**67.9**

Table 14: Biology test scores as a percentage. SemanticILP outperforms various baselines on the ProcessBank dataset and roughly matches the specialized best method.

4.4.4. Error and Timing Analysis

For some insight into the results, we include a brief analysis of our system's output compared to that of other systems.

We identify a few main reasons for SemanticILP's errors. Not surprisingly, some mistakes (see Appendix figure for an example) can be traced back to failures in generating proper annotation (SemanticGraph). Improvement in SRL modules or redundancy can help address this. Some mistakes are from the current ILP model not supporting the ideal reasoning, i.e., the requisite knowledge exists in the annotations, but the reasoning fails to exploit it. Another group of mistakes is due to the complexity of the sentences, and the system lacking a way to represent the underlying phenomena with our current annotators.

A weakness (that doesn't seem to be particular to our solver) is reliance on explicit mentions. If there is a meaning indirectly implied by the context and our annotators are not able to capture it, our solver will miss such questions. There will be more room for improvement

on such questions with the development of discourse analysis systems.

When solving the questions that don't have an attached paragraph, relevant sentences need to be fetched from a corpus. A subset of mistakes on this dataset occurs because the extracted knowledge does not contain the correct answer.

ILP Solution Properties.

Our system is implemented using many constraints, requires using many linear inequalities which get instantiated on each input instanced, hence there are a different number of variables and inequalities for each input instance. There is an overhead time for pre-processing an input instance, and convert it into an instance graph. Here in the timing analysis we provide we ignore the annotation time, as it is done by black-boxes outside our solver.

Table 15 summarizes various ILP and support graph statistics for SemanticILP, averaged across ProcessBank questions. Next to SemanticILP we have included numbers from TableILP which has similar implementation machinery, but on a very different representation. While the size of the model is a function of the input instance, on average, SemanticILP tends to have a bigger model (number of constraints and variables). The model creation time is significantly time-consuming in SemanticILP as involves many graph traversal operations and jumps between nodes and edges. We also providing times statistics for TupleInf which takes roughly half the time of TableILP, which means that it is faster than SemanticILP.

Category	Quantity	Avg. (SemanticILP)	Avg. (TableILP)	Avg. (TupleInf)
ILP complexity	#variables	2254.9	1043.8	1691.0
	#constraints	4518.7	4417.8	4440.0
Timing stats	model creation	5.3 sec	1.9 sec	1.7 sec
	solving the ILP	1.8 sec	2.1 sec	0.3 sec

Table 15: SemanticILP statistics averaged across questions, as compared to TableILP and TupleInf statistics.

4.4.5. Ablation Study

In order to better understand the results, we ablate the contribution of different annotation combinations, where we drop different combination from the ensemble model. We retrain the ensemble, after dropping each combination.

The results are summarized in Table 16. While Comb-1 seems to be important for science tests, it has limited contribution to the biology tests. On 8th grade exams, the Verb-SRL and Comma-SRL-based alignments provide high value. Structured combinations (e.g., Verb-SRL-based alignments) are generally more important for the biology domain.

	AI2Public 8th	ProcessBank
Full SemanticILP	55.9	67.9
no Comb-1	-3.1	-1.8
no Comb-2	-2.0	-4.6
no Comb-3	-0.6	-1.8
no Comb-4	-3.1	-1.8
no Comb-5	-0.1	-5.1

Table 16: Ablation study of SemanticILP components on various datasets. The first row shows the overall test score of the full system, while other rows report the change in the score as a result of dropping an individual combination. The combinations are listed in Table 12.

Complementarity to IR. Given that in the science domain the input snippets fed to SemanticILP are retrieved through a process similar to the IR solver, one might naturally expect some similarity in the predictions. The pie-chart in Figure 12 shows the overlap between mistakes and correct predictions of SemanticILP and IR on 50 randomly chosen training questions from AI2Public 4th. While there is substantial overlap in questions that both answer cor-

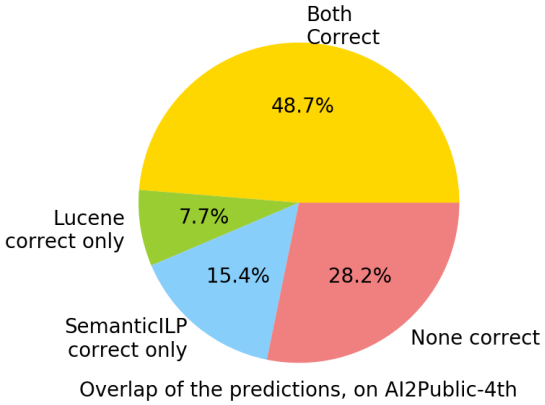


Figure 12: Overlap of the predictions of SemanticILP and IR on 50 randomly-chosen questions from AI2Public 4th.

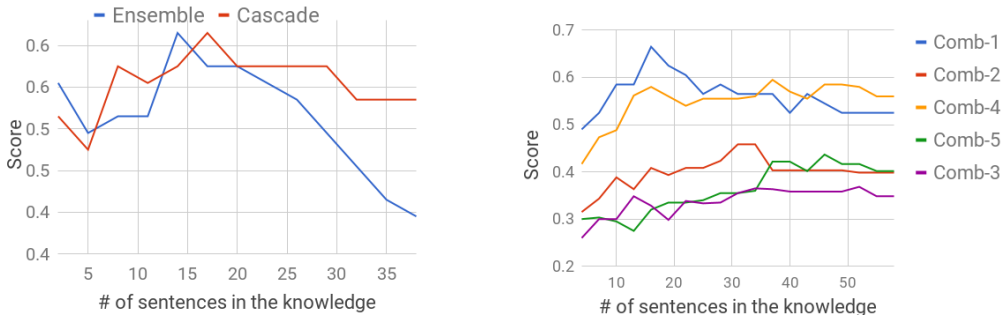


Figure 13: Performance change for varying knowledge length.

rectly (the yellow slice) and both miss (the
red slice), there is also a significant number of questions solved by SemanticILP but not IR (the blue slice), almost twice as much as the questions solved by IRbut not SemanticILP (the green slice).

Cascade Solvers.

In Tables 13 and 14, we presented one *single* instance of SemanticILP with state-of-art results on multiple datasets, where the solver was an ensemble of semantic combinations (presented in Table 12). Here we show a simpler approach that achieves stronger results on individual datasets, at the cost of losing a little generalization across domains. Specifically, we create two "cascades" (i.e., decision lists) of combinations, where the ordering of combinations in the cascade is determined by the training set precision of the simplified solver representing an annotator combination (combinations with higher precision appear earlier). One cascade solver targets science exams and the other the biology dataset.

The results are reported in Table 17. On the 8th grade data, the cascade solver created for science test achieves higher scores than the generic ensemble solver. Similarly, the cascade solver on the biology domain outperforms the ensemble solver on the ProcessBank dataset.

Effect of Varying Knowledge Length. We analyze the performance of the system as a function of the length of the paragraph fed into SemanticILP, for 50 randomly selected training questions from the Regents 4th set. Figure 13 (left) shows the overall system, for two combinations introduced earlier, as a function of knowledge length, counted as the number of sentences in the paragraph.

	Dataset	Ensemble	Cascade (Science)	Cascade (Biology)
Science	Regents 4th	**67.6**	64.7	63.1
	AI2Public 4th	**59.7**	56.7	55.7
	Regents 8th	66.0	**69.4**	60.3
	AI2Public 8th	55.9	**56.5**	54.3
	ProcessBank	67.9	59.6	**68.8**

Table 17: Comparison of test scores of SemanticILP using a generic ensemble vs. domain-targeted cascades of annotation combinations.

As expected, the solver improves with more sentences, until around 12-15 sentences, after which it starts to worsen with the addition of more irrelevant knowledge. While the cascade combinations did not show much generalization across domains, they have the advantage of a smaller drop when adding irrelevant knowledge compared to the ensemble solver. This can be explained by the simplicity of cascading and minimal training compared to the ensemble of annotation combinations.

Figure 13 (right) shows the performance of individual combinations as a function of knowledge length. It is worth highlighting that while Comb-1 (blue) often achieves higher coverage and good scores in simple paragraphs (e.g., science exams), it is highly sensitive to knowledge length. On the other hand, highly-constrained combinations have a more consistent performance with increasing knowledge length, at the cost of lower coverage.

4.5. Summary and Discussion

This chapter extends our abductive reasoning system from Chapter 3 to consume raw text as input knowledge. This is the first system to successfully use a wide range of semantic abstractions to perform a high-level NLP task like Question Answering. The approach is especially suitable for domains that require reasoning over a diverse set of linguistic

constructs but have limited training data. To address these challenges, we present the first system, to the best of our knowledge, that reasons over a wide range of semantic abstractions of the text, which are derived using off-the-shelf, general-purpose, pre-trained natural language modules such as semantic role labelers. Representing multiple abstractions as a family of graphs, we translate question answering (QA) into a search for an optimal subgraph that satisfies certain global and local properties. This formulation generalizes several prior structured QA systems. Our system, SemanticILP, demonstrates strong performance on two domains simultaneously. In particular, on a collection of challenging science QA datasets, it outperforms various state-of-the-art approaches, including neural models, broad coverage information retrieval, and specialized techniques using structured knowledge bases, by 2%-6%.

A key limitation of the system here is that its abstractions are mostly extracted from explicit mentions of in a given text. However, a major portion of our understanding come is only implied from text (not directly mention). We propose a challenge dataset for such questions (limited to the *temporal* domain) in Chapter 7. Additionally, the two systems discussed in Chapter 3 and here, lack explicit explicit attention mechanism to the content of the questions. We study this topic in Chapter 5.

CHAPTER 5 : Learning Essential Terms in Questions

> "The trouble with Artificial Intelligence is that computers don't give a damn-or so I will argue by considering the special case of understanding natural language."
>
> — John Haugeland, 1979

5.1. Overview

Many of today's QA systems often struggle with seemingly simple questions because they are unable to reliably identify which question words are redundant, irrelevant, or even intentionally distracting.[1] This reduces the systems' precision and results in questionable "reasoning" even when the correct answer is selected among the given alternatives. The variability of subject domain and question style makes identifying essential question words challenging. Further, essentiality is context dependent—a word like 'animals' can be critical for one question and distracting for another. Consider the following example:

One way animals usually respond to a sudden drop in temperature is by (A) sweating (B) shivering (C) blinking (D) salivating.

The system we discussed in Chapter 3, TableILP (Khashabi et al., 2016), which performs reasoning by aligning the question to semi-structured knowledge, aligns only the word 'animals' when answering this question. Not surprisingly, it chooses an incorrect answer. The issue is that it does not recognize that "drop in temperature" is an essential aspect of the question.

Towards this goal, we propose a system that can assign an essentiality score to each term in the question. For the above example, our system generates the scores shown in Figure 14, where more weight is put on "temperature"

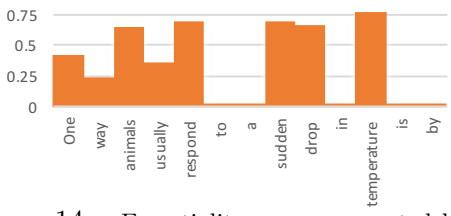


Figure 14: Essentiality scores generated by our system, which assigns high essentiality to "drop" and "temperature".

[1] This chapter is based on the following publication: Khashabi et al. (2017)

and "sudden drop". A QA system, when armed with such information, is expected to exhibit a more informed behavior.

We make the following contributions:

(A) We introduce the notion of *question term essentiality* and release a new dataset of 2,223 crowd-sourced essential term annotated questions (total 19K annotated terms) that capture this concept.[2] We illustrate the importance of this concept by demonstrating that humans become substantially worse at QA when even a few essential question terms are dropped.

(B) We design a classifier that is effective at predicting question term essentiality. The F1 (0.80) and per-sentence mean average precision (MAP, 0.90) scores of our classifier supercede the closest baselines by 3%-5%. Further, our classifier generalizes substantially better to unseen terms.

(C) We show that this classifier can be used to improve a surprisingly effective IR based QA system (Clark et al., 2016) by 4%-5% on previously used question sets and by 1.2% on a larger question set. We also incorporate the classifier in TableILP (Khashabi et al., 2016), resulting in fewer errors when sufficient knowledge is present for questions to be meaningfully answerable.

5.1.1. Related Work

Our work can be viewed as the study of an intermediate layer in QA systems. Some systems implicitly model and learn it, often via indirect signals from end-to-end training data. For instance, Neural Networks based models (Wang et al., 2016; Tymoshenko et al., 2016; Yin et al., 2016) implicitly compute some kind of *attention*. While this is intuitively meant to weigh key words in the question more heavily, this aspect hasn't been systematically evaluated, in part due to the lack of ground truth annotations.

There is related work on extracting *question type* information (Li and Roth, 2002; Li et al.,

[2] Annotated dataset and classifier available at `https://github.com/allenai/essential-terms`

2007) and applying it to the design and analysis of end-to-end QA systems (Moldovan et al., 2003). The concept of term essentiality studied in this work is different, and so is our supervised learning approach compared to the typical rule-based systems for question type identification.

Another line of relevant work is sentence compression (Clarke and Lapata, 2008), where the goal is to minimize the content while maintaining grammatical soundness. These approaches typically build an internal importance assignment component to assign significance scores to various terms, which is often done using language models, co-occurrence statistics, or their variants (Knight and Marcu, 2002; Hori and Sadaoki, 2004). We compare against unsupervised baselines inspired by such importance assignment techniques.

In a similar spirit, Park and Croft (2015) use translation models to extract key terms to prevent semantic drift in query expansion.

One key difference from general text summarization literature is that we operate on questions, which tend to have different essentiality characteristics than, say, paragraphs or news articles. As we discuss in Section 5.2.1, typical indicators of essentiality such as being a proper noun or a verb (for event extraction) are much less informative for questions. Similarly, while the opening sentence of a Wikipedia article is often a good summary, it is the last sentence (in multi-sentence questions) that contains the most pertinent words.

In parallel to our effort, Jansen et al. (2017) recently introduced a science QA system that uses the notion of *focus words*. Their rule-based system incorporates grammatical structure, answer types, etc. We take a different approach by learning a supervised model using a new annotated dataset.

5.2. Essential Question Terms

In this section, we introduce the notion of *essential question terms*, present a dataset annotated with these terms, and describe two experimental studies that illustrate the importance

of this notion—we show that when dropping terms from questions, humans' performance degrades significantly faster if the dropped terms are essential question terms.

Given a question q, we consider each non-stopword token in q as a candidate for being an essential question term. Precisely defining what is essential and what isn't is not an easy task and involves some level of inherent subjectivity. We specified *three broad criteria*: 1) altering an essential term should change the intended meaning of q, 2) dropping non-essential terms should not change the correct answer for q, and 3) grammatical correctness is not important. We found that given these relatively simple criteria, human annotators had a surprisingly high agreement when annotating elementary-level science questions. Next we discuss the specifics of the crowd-sourcing task and the resulting dataset.

5.2.1. Crowd-Sourced Essentiality Dataset

We collected 2,223 elementary school science exam questions for the annotation of essential terms. This set includes the questions used by Clark et al. (2016)[3] and additional ones obtained from other public resources such as the Internet or textbooks. For each of these questions, we asked crowd workers[4] to annotate essential question terms based on the above criteria as well as a few examples of essential and non-essential terms. Figure 15 depicts the annotation interface.

The questions were annotated by 5 crowd workers,[5] and resulted in 19,380 annotated terms. The Fleiss' kappa statistic (Fleiss, 1971) for this task was $\kappa = 0.58$, indicating a level of inter-annotator agreement very close to 'substantial'. In particular, all workers agreed on 36.5% of the terms and at least 4 agreed on 69.9% of the terms. We use the proportion of workers that marked a term as essential to be its annotated essentiality score.

On average, less than one-third (29.9%) of the terms in each question were marked as

[3]These are the only publicly available state-level science exams. http://www.nysedregents.org/Grade4/Science/

[4]We use Amazon Mechanical Turk for crowd-sourcing.

[5]A few invalid annotations resulted in about 1% of the questions receiving fewer annotations. 2,199 questions received at least 5 annotations (79 received 10 annotations due to unintended question repetition), 21 received 4 annotations, and 4 received 3 annotations.

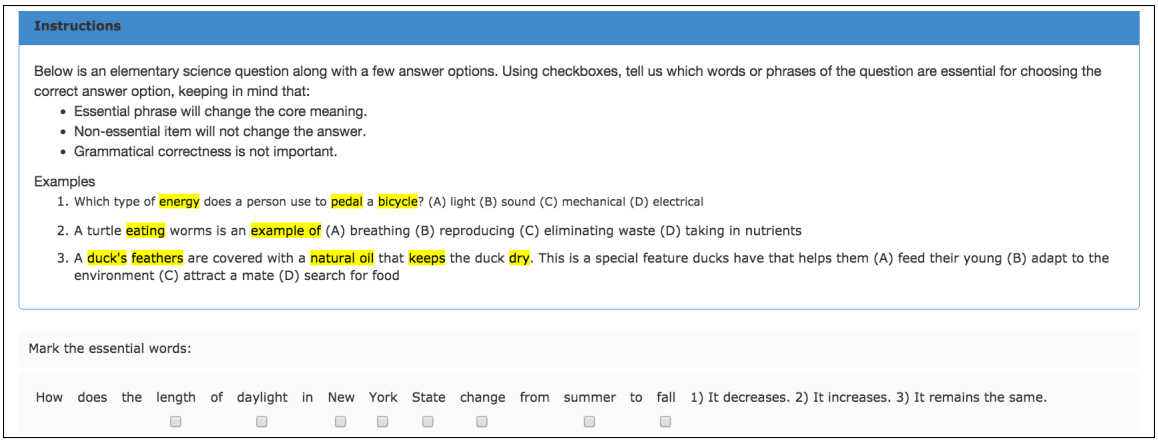


Figure 15: Crowd-sourcing interface for annotating essential terms in a question, including the criteria for essentiality and sample annotations.

essential (i.e., score > 0.5). This shows the large proportion of distractors in these science tests (as compared to traditional QA datasets), further showing the importance of this task. Next we provide some insights into these terms.

We found that part-of-speech (POS) tags are not a reliable predictor of essentiality, making it difficult to hand-author POS tag based rules. Among the proper nouns (NNP, NNPS) mentioned in the questions, fewer than half (47.0%) were marked as essential. This is in contrast with domains such as news articles where proper nouns carry perhaps the most important information. Nearly two-thirds (65.3%) of the mentioned comparative adjectives (JJR) were marked as essential, whereas only a quarter of the mentioned superlative adjectives (JJS) were deemed essential. Verbs were marked essential less than a third (32.4%) of the time. This differs from domains such as math word problems where verbs have been found to play a key role (Hosseini et al., 2014).

The best single indicator of essential terms, not surprisingly, was being a scientific term[6] (such as *precipitation* and *gravity*). 76.6% of such terms occurring in questions were marked as essential.

In summary, we have a term essentiality annotated dataset of 2,223 questions. We split this

[6]We use 9,144 science terms from Khashabi et al. (2016).

into train/development/test subsets in a 70/9/21 ratio, resulting in 483 test sentences used for per-question evaluation.

We also derive from the above an annotated dataset of 19,380 terms by pooling together all terms across all questions. Each term in this larger dataset is annotated with an essentiality score in the context of the question it appears in. This results in 4,124 test instances (derived from the above 483 test questions). We use this dataset for per-term evaluation.

5.2.2. The Importance of Essential Terms

Here we report a second crowd-sourcing experiment that validates our hypothesis that the question terms marked above as essential are, in fact, essential for understanding and answering the questions. Specifically, we ask: *Is the question still answerable by a human if a fraction of the essential question terms are eliminated?* For instance, the sample question in the introduction is unanswerable when "drop" and "temperature" are removed from the question: *One way animals usually respond to a sudden * in * is by ___?*

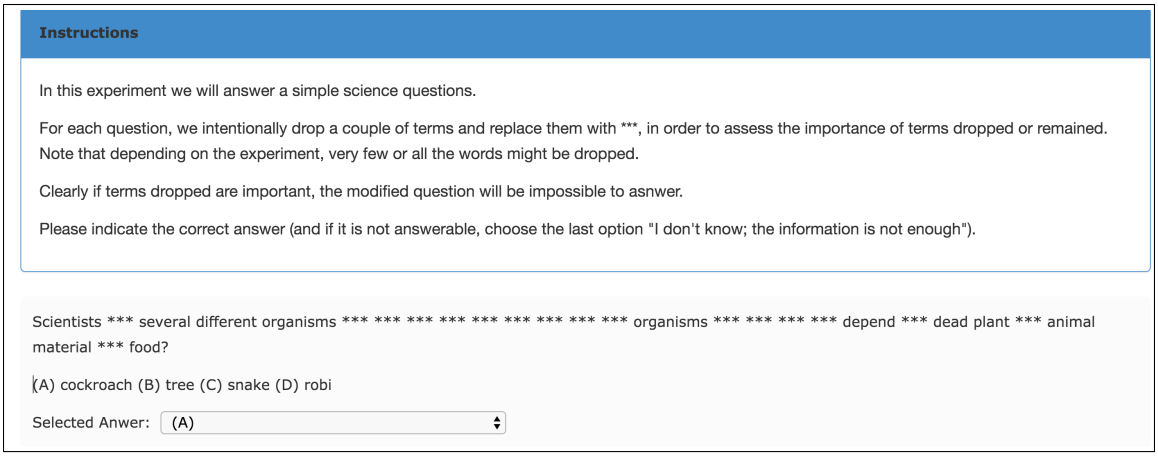


Figure 16: Crowd-sourcing interface for verifying the validity of essentiality annotations generated by the first task. Annotators are asked to answer, if possible, questions with a group of terms dropped.

To this end, we consider both the annotated essentiality scores as well as the score produced by our trained classifier (to be presented in Section 5.3). We first generate candidate sets of terms to eliminate using these essentiality scores based on a threshold $\xi \in \{0, 0.2, \ldots, 1.0\}$:

(a) **essential set**: terms with score $\geq \xi$; (b) **non-essential set**: terms with score $< \xi$. We then ask crowd workers to try to answer a question after replacing each candidate set of terms with "***". In addition to four original answer options, we now also include "I don't know. The information is not enough" (cf. Figure 16 for the user interface).[7] For each value of ξ, we obtain 5×269 annotations for 269 questions. We measure how often the workers feel there is sufficient information to attempt the question and, when they do attempt, how often do they choose the right answer.

Each value of ξ results in some fraction of terms to be dropped from a question; the exact number depends on the question and on whether we use annotated scores or our classifier's scores. In Figure 17, we plot the average fraction of terms dropped on the horizontal axis and the corresponding fraction of questions attempted on the vertical axis. Solid lines indicate annotated scores and dashed lines indicate classifier scores. Blue lines (bottom left) illustrate the effect of eliminating essential sets while red lines (top right) reflect eliminating non-essential sets.

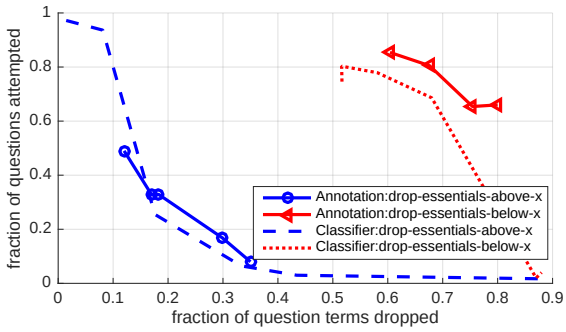


Figure 17: The relationship between the fraction of question words dropped and the fraction of the questions attempted (fraction of the questions workers felt comfortable answering). Dropping most essential terms (blue lines) results in very few questions remaining answerable, while least essential terms (red lines) allows most questions to still be answerable. Solid lines indicate human annotation scores while dashed lines indicate predicted scores.

We make two observations. First, the solid blue line (bottom-left) demonstrates that dropping even a small fraction of question terms marked as essential dramatically reduces the QA performance of humans. E.g., dropping just 12% of the terms (with high essentiality scores) makes 51% of the questions unanswerable. The solid red line (top-right), on the

[7] It is also possible to directly collect essential term groups using this task. However, collecting such sets of essential terms would be substantially more expensive, as one must iterate over exponentially many subsets rather than the linear number of terms used in our annotation scheme.

other hand, shows the opposite trend for terms marked as not-essential: even after dropping 80% of such terms, 65% of the questions remained answerable.

Second, the dashed lines reflecting the results when using scores from our ET classifier are very close to the solid lines based on human annotation. This indicates that our classifier, to be described next, closely captures human intuition.

5.3. Essential Terms Classifier

Given the dataset of questions and their terms annotated with essential scores, is it possible to learn the underlying concept? Towards this end, given a question q , answer options a, and a question term q_l, we seek a classifier that predicts whether q_l is essential for answering q. We also extend it to produce an essentiality score $et(q_l, q, a) \in [0, 1]$.[8] We use the annotated dataset from Section 2, where real-valued essentiality scores are binarized to 1 if they are at least 0.5, and to 0 otherwise.

We train a linear SVM classifier (Joachims, 1998), henceforth referred to as **ET classifier**. Given the complex nature of the task, the features of this classifier include syntactic (e.g., dependency parse based) and semantic (e.g., Brown cluster representation of words (Brown et al., 1992), a list of scientific words) properties of question words, as well as their combinations. In total, we use 120 types of features (cf. Appendix A.3.1).

Baselines. To evaluate our approach, we devise a few simple yet relatively powerful baselines.

First, for our supervised baseline, given (q_l, q, a) as before, we ignore q and compute how often is q_l annotated as essential in the entire dataset. In other words, the score for q_l is the proportion of times it was marked as essential in the annotated dataset. If the instance is never observer in training, we choose an arbitrary label as prediction. We refer to this

[8]The essentiality score may alternatively be defined as $et(q_l, q)$, independent of the answer options a. This is more suitable for non-multiple choice questions. Our system uses a only to compute PMI-based statistical association features for the classifier. In our experiments, dropping these features resulted in only a small drop in the classifier's performance.

baseline as *label proportion baseline* and create two variants of it: PROPSURF based on surface string and PROPLEM based on lemmatizing the surface string. For unseen q_l, this baseline makes a random guess with uniform distribution.

Our unsupervised baseline is inspired by work on sentence compression (Clarke and Lapata, 2008) and the PMI solver of Clark et al. (2016), which compute word importance based on co-occurrence statistics in a large corpus. In a corpus $\mathcal{C}$ of 280 GB of plain text (5×10^{10} tokens) extracted from Web pages,[9] we identify unigrams, bigrams, trigrams, and skip-bigrams from q and each answer option a_i. For a pair (x, y) of n-grams, their pointwise mutual information (PMI) (Church and Hanks, 1989) in $\mathcal{C}$ is defined as $\log \frac{p(x,y)}{p(x)p(y)}$ where $p(x, y)$ is the co-occurrence frequency of x and y (within some window) in $\mathcal{C}$. For a given word x, we find all pairs of question n-grams and answer option n-grams. MAXPMI and SUMPMI score the importance of a word x by max-ing or summing, resp., PMI scores $p(x, y)$ across all answer options y for q. A limitation of this baseline is its dependence on the existence of answer options, while our system makes essentiality predictions independent of the answer options.

We note that all of the aforementioned baselines produce real-valued confidence scores (for each term in the question), which can be turned into binary labels (essential and non-essential) by thresholding at a certain confidence value.

5.3.1. Evaluation

We consider two natural evaluation metrics for essentiality detection, first treating it as a binary prediction task at the level of individual terms and then as a task of ranking terms within each question by the degree of essentiality.

Binary Classification of Terms. We consider all question terms pooled together as described in Section 5.2.1, resulting in a dataset of 19,380 terms annotated (in the context of the corresponding question) independently as essential or not. The ET classifier is trained

[9] Collected by Charles Clarke at the University of Waterloo, and used previously by Turney (2013).

on the train subset, and the threshold is tuned using the dev subset.

For each term in the corresponding test set of 4,124 instances, we use various methods to predict whether the term is essential (for the corresponding question) or not. Table 18 summarizes the resulting performance. For the threshold-based scores, each method was tuned to maximize the F1 score based on the dev set. The ET classifier achieves an F1 score of 0.80, which is 5%-14% higher than the baselines. Its accuracy at 0.75 is statistically significantly better than all baselines based on the Binomial[10] exact test (Howell, 2012) at p-value 0.05.

	AUC	Acc	P	R	F1
MaxPMI †	0.74	0.67	0.88	0.65	0.75
SumPMI †	0.74	0.67	0.88	0.65	0.75
PropSurf	0.79	0.61	0.68	0.64	0.66
PropLem	0.80	0.63	0.76	0.64	0.69
ET Classifier	**0.79**	**0.75**	**0.91**	**0.71**	**0.80**

Table 18: Effectiveness of various methods for identifying essential question terms in the test set, including area under the PR curve (AUC), accuracy (Acc), precision (P), recall (R), and F1 score. ET classifier substantially outperforms all supervised and unsupervised (denoted with †) baselines.

As noted earlier, each of these essentiality identification methods are parameterized by a threshold for balancing precision and recall. This allows them to be tuned for end-to-end performance of the downstream task. We use this feature later when incorporating the ET classifier in QA systems. Figure 18 depicts the PR curves for various methods as the threshold is varied, highlighting that the ET classifier performs reliably at various recall points. Its precision, when tuned to optimize F1, is 0.91, which is very suitable for high-precision applications. It has a 5% higher AUC (area under the curve) and outperforms baselines by roughly 5% throughout the precision-recall spectrum.

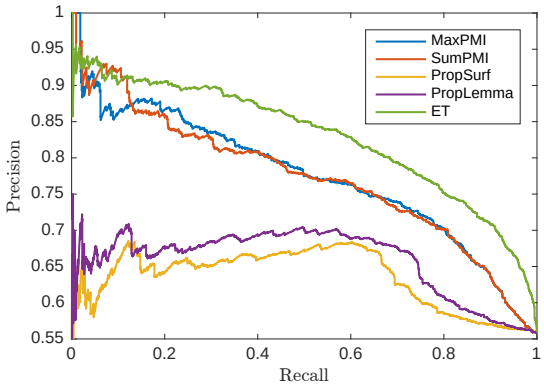


Figure 18: Precision-recall trade-off for various classifiers as the threshold is varied. ET classifier (green) is significantly better throughout.

[10]Each test term prediction is assumed to be a binomial.

As a second study, we assess how well our classifier **generalizes to unseen terms**. For this, we consider only the 559 test terms that do not appear in the train set.[11] Table 19 provides the resulting performance metrics. We see that the frequency based supervised baselines, having never seen the test terms, stay close to the default precision of 0.5. The unsupervised baselines, by nature, generalize much better but are substantially dominated by our ET classifier, which achieves an F1 score of 78%. This is only 2% below its own F1 across all seen and unseen terms, and 6% higher than the second best baseline.

	AUC	Acc	P	R	F1
MaxPMI †	0.75	0.63	0.81	0.65	0.72
SumPMI †	0.75	0.63	0.80	0.66	0.72
PropSurf	0.57	0.51	0.49	0.61	0.54
PropLem	0.58	0.49	0.50	0.59	0.54
ET Classifier	**0.78**	**0.71**	**0.88**	**0.71**	**0.78**

Table 19: Generalization to unseen terms: Effectiveness of various methods, using the same metrics as in Table 18. As expected, supervised methods perform poorly, similar to a random baseline. Unsupervised methods generalize well, but the ET classifier again substantially outperforms them.

Ranking Question Terms by Essentiality. Next, we investigate the performance of the ET classifier as a system that ranks all terms within a question in the order of essentiality. Thus, unlike the previous evaluation that pools terms together across questions, we now consider each question as a unit. For the ranked list produced by each classifier for each question, we compute the average precision (AP).[12] We then take the mean of these AP values across questions to obtain the mean average precision (MAP) score for the classifier.

System	MAP
MaxPMI †	0.87
SumPMI †	0.85
PropSurf	0.85
PropLem	0.86
ET Classifier	**0.90**

Table 20: Effectiveness of various methods for ranking the terms in a question by essentiality. † indicates unsupervised method. Mean-Average Precision (MAP) numbers reflect the mean (across all test set questions) of the average precision of the term ranking for each question. ET classifier again substantially outperforms all baselines.

[11] In all our other experiments, test and train questions are always distinct but may have some terms in common.

[12] We rank all terms within a question based on their essentiality scores. For any true positive instance at rank k, the precision at k is defined to be the number of positive instances with rank no more than k, divided by k. The average of all these precision values for the ranked list for the question is the *average precision*.

The results for the test set (483 questions) are shown in Table 20. Our ET classifier achieves a MAP of 90.2%, which is 3%-5% higher than the baselines, and demonstrates that one can learn to reliably identify essential question terms.

5.4. Using ET Classifier in QA Solvers

In order to assess the utility of our ET classifier, we investigate its impact on two end-to-end QA systems. We start with a brief description of the question sets.

Question Sets. We use three question sets of 4-way multiple choice questions.[13] REGENTS and AI2PUBLIC are two publicly available elementary school science question set. REGENTS comes with 127 training and 129 test questions; AI2PUBLIC contains 432 training and 339 test questions that subsume the smaller question sets used previously (Clark et al., 2016; Khashabi et al., 2016). REGTSPERTD set, introduced by Khashabi et al. (2016), has 1,080 questions obtained by automatically perturbing incorrect answer choices for 108 New York Regents 4th grade science questions. We split this into 700 train and 380 test questions.

For each question, a solver gets a score of 1 if it chooses the correct answer and $1/k$ if it reports a k-way tie that includes the correct answer.

QA Systems. We investigate the impact of adding the ET classifier to two state-of-the-art QA systems for elementary level science questions. Let q be a multiple choice question with answer options $\{a_i\}$. The *IR Solver* from Clark et al. (2016) searches, for each a_i, a large corpus for a sentence that best matches the (q, a_i) pair. It then selects the answer option for which the match score is the highest. The inference based TABLEILP *Solver* from Khashabi et al. (2016), on the other hand, performs QA by treating it as an optimization problem over a semi-structured knowledge base derived from text. It is designed to answer questions requiring multi-step inference and a combination of multiple facts.

[13] Available at `http://allenai.org/data.html`

For each multiple-choice question (q, a), we use the ET classifier to obtain essential term scores s_l for each token q_l in q; $s_l = et(q_l, q, a)$. We will be interested in the subset ω of all terms T_q in q with essentiality score above a threshold ξ: $\omega(\xi; q) = \{l \in T_q \mid s_l > \xi\}$. Let $\overline{\omega}(\xi; q) = T_q \setminus \omega(\xi; q)$. For brevity, we will write $\omega(\xi)$ when q is implicit.

5.4.1. IR solver + ET

To incorporate the ET classifier, we create a parameterized IR system called IR + ET(ξ) where, instead of querying a (q, a_i) pair, we query $(\omega(\xi; q), a_i)$.

While IR solvers are generally easy to implement and are used in popular QA systems with surprisingly good performance, they are often also sensitive to the nature of the questions they receive. Khashabi et al. (2016) demonstrated that a minor perturbation of the questions, as embodied in the RegtsPertd question set, dramatically reduces the performance of IR solvers. Since the perturbation involved the introduction of distracting incorrect answer options, we hypothesize that a system with better knowledge of what's important in the question will demonstrate increased robustness to such perturbation.

Table 21 validates this hypothesis, showing the result of incorporating ET in IR, as IR + ET($\xi = 0.36$), where ξ was selected by optimizing end-to-end performance on the training set. We observe a 5% boost in the score on RegtsPertd, showing that incorporating the notion of essentiality makes the system more robust to perturbations.

Dataset	Basic IR	IR + ET
Regents	59.11	**60.85**
AI2Public	57.90	**59.10**
RegtsPertd	61.84	**66.84**

Table 21: Performance of the IR solver without (Basic IR) and with (IR + ET) essential terms. The numbers are solver scores (%) on the test sets of the three datasets.

Adding ET to IR also improves its performance on standard test sets. On the larger AI2Public question set, we see an improvement of 1.2%. On the smaller Regents set, introducing ET improves IRsolvers score by 1.74%, bringing it close to the state-of-the-art solver, TableILP, which achieves a score of 61.5%. This demonstrates that the notion of

essential terms can be fruitfully exploited to improve QA systems.

5.4.2. TableILP *solver* + ET

Our essentiality guided query filtering helped the IR solver find sentences that are more relevant to the question. However, for TableILP an added focus on essential terms is expected to help only when the requisite knowledge is present in its relatively small knowledge base. To remove confounding factors, we focus on questions that are, in fact, answerable.

To this end, we consider three (implicit) requirements for TableILP to demonstrate reliable behavior: (1) the existence of relevant knowledge, (2) correct alignment between the question and the knowledge, and (3) a valid reasoning chain connecting the facts together. Judging this for a question, however, requires a significant manual effort and can only be done at a small scale.

Question Set. We consider questions for which the TableILP solver does have access to the requisite knowledge and, as judged by a human, a reasoning chain to arrive at the correct answer. To reduce manual effort, we collect such questions by starting with the correct reasoning chains ('support graphs') provided by TableILP. A human annotator is then asked to paraphrase the corresponding questions or add distracting terms, while maintaining the general meaning of the question. Note that this is done independent of essentiality scores. For instance, the modified question below changes two words in the question without affecting its core intent:

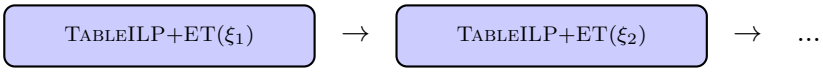
Original question: A fox grows thicker fur as a season changes. This adaptation helps the fox to (A) find food(B) keep warmer(C) grow stronger(D) escape from predators
Generated question: An animal grows thicker hair as a season changes. This adaptation helps to (A) find food(B) keep warmer(C) grow stronger(D) escape from predators

While these generated questions should arguably remain correctly answerable by TableILP, we found that this is often not the case. To investigate this, we curate a small dataset Q_R with 12 questions (see the Appendix) on each of which, despite having the required knowl-

edge and a plausible reasoning chain, TableILP fails.

Modified Solver. To incorporate question term essentiality in the TableILP solver while maintaining high recall, we employ a *cascade* system that starts with a strong essentiality requirement and progressively weakens it.

Following the notation of Chapter 3, let $x(q_l)$ be a binary variable that denotes whether or not the l-th term of the question is used in the final reasoning graph. We enforce that terms with essentiality score above a threshold ξ must be used: $x(q_l) = 1,\ \forall l \in \omega(\xi)$. Let TableILP+ET($\xi$) denote the resulting system which can now be used in a cascading architecture:



where $\xi_1 < \xi_2 < \ldots < \xi_k$ is a sequence of thresholds. Questions unanswered by the first system are delegated to the second, and so on. The cascade has the same recall as TableILP, as long as the last system is the vanilla TableILP. We refer to this configuration as Cascades($\xi_1, \xi_2, \ldots, \xi_k$).

This can be implemented via repeated calls to TableILP+ET(ξ_j) with j increasing from 1 to k, stopping if a solution is found. Alternatively, one can simulate the cascade via a single extended ILP using k new binary variables z_j with constraints: $|\omega(\xi_j)| * z_j \leq \sum_{l \in \omega(\xi_j)} x(q_l)$ for $j \in \{1, \ldots, k\}$, and adding $M * \sum_{j=1}^{k} z_j$ to the objective function, for a sufficiently large constant M.

We evaluate Cascades(0.4, 0.6, 0.8, 1.0) on our question set, Q_R. By employing essentiality information provided by the ET classifier, Cascades corrects 41.7% of the mistakes made by vanilla TableILP. This error-reduction illustrates that the extra attention mechanism added to TableILP via the concept of essential question terms helps it cope with distracting terms.

5.5. Summary

This chapter introduces and studies the notion of *essential question terms* with the goal of improving such QA solvers. We illustrate the importance of essential question terms by showing that humans' ability to answer questions drops significantly when essential terms are eliminated from questions. We then develop a classifier that reliably (90% mean average precision) identifies and ranks essential terms in questions. Finally, we use the classifier to demonstrate that the notion of question term essentiality allows state-of-the-art QA solvers for elementary-level science questions to make better and more informed decisions, improving performance by up to 5%.

Part II

Moving the Peaks Higher: Designing More Challenging Datasets

CHAPTER 6 : A Challenge Set for Reasoning on Multiple Sentences

> "Human beings, viewed as behaving systems, are quite simple. The apparent complexity of our behavior over time is largely a reflection of the complexity of the environment in which we find ourselves."
>
> — Herbert A. Simon, The Sciences of the Artificial, 1968

6.1. Overview

In this chapter we develop a reading comprehension challenge in which answering each of the questions requires reasoning over multiple sentences.[1]

There is evidence that answering 'single-sentence questions', i.e. questions that can be answered from a single sentence of the given paragraph, is easier than answering multi-sentence questions', which require multiple sentences to answer a given question. For example, (Richardson et al., 2013) released a reading comprehension dataset that contained both single-sentence and multi-sentence questions; models proposed for this task yielded considerably better performance on the single-sentence questions than on the multi-sentence questions (according to (Narasimhan and Barzilay, 2015) accuracy of about 83% and 60% on these two types of questions, respectively).

There could be multiple reasons for this. First, multi-sentence reasoning seems to be inherently a difficult task. Research has shown that while complete-sentence construction emerges as early as first grade for many children, their ability to integrate sentences emerges only in fourth grade (Berninger et al., 2011). Answering multi-sentence questions might be more challenging for an automated system because it involves more than just processing individual sentences but rather combining linguistic, semantic and background knowledge across sentences—a computational challenges in itself. Despite these challenges, multi-sentence questions can be answered by humans and hence present an interesting yet reasonable goal for AI systems (Davis, 2014).

[1] This chapter is based on the following publication: Khashabi et al. (2018a).

In this work, we propose a multi-sentence QA challenge in which questions can be answered only using information from multiple sentences. Specifically, we present MULTIRC (Multi-Sentence Reading Comprehension)[2]—a dataset of short paragraphs and multi-sentence questions that can be answered from the content of the paragraph. Each question is associated with several choices for answer-options, out of which *one or more* correctly answer the question. Figure 19 shows two examples from our dataset. Each instance consists of a multi-sentence paragraph, a question, and answer-options. All instances were constructed such that it is not possible to answer a question correctly without gathering information from multiple sentences. Due to space constraints, the figure shows only the relevant sentences from the original paragraph. The entire corpus consists of 871 paragraphs and about $\sim$ 6k multi-sentence questions.

S3: Hearing noises in the garage, Mary Murdock finds a bleeding man, mangled and impaled on her jeep's bumper. S5: Panicked, she hits him with a golf club. S10: Later the news reveals the missing man is kindergarten teacher, Timothy Emser. S12: It transpires that Rick, her boyfriend, gets involved in the cover up and goes to retrieve incriminatory evidence off the corpse, but is killed, replaced in Emser's grave. S13: It becomes clear Emser survived. S15: He stalks Mary many ways.
Who is stalking Mary? A)* Timothy D) Rick B) Timothy's girlfriend E) Murdock C)* The man she hit F) Her Boyfriend
S1: Most young mammals, including humans, play. S2: Play is how they learn the skills that they will need as adults. S6: Big cats also play. S8: At the same time, they also practice their hunting skills. S11: Human children learn by playing as well. S12: For example, playing games and sports can help them learn to follow rules. S13: They also learn to work together.
What do human children learn by playing games and sports? A)* They learn to follow rules and work together B) hunting skills C)* skills that they will need as adult

Figure 19: Examples from our MULTIRCcorpus. Each example shows relevant excerpts from a paragraph; multi-sentence question that can be answered by combining information from multiple sentences of the paragraph; and corresponding answer-options. The correct answer(s) is indicated by a *. Note that there can be multiple correct answers per question.

The goal of this dataset is to encourage the research community to explore approaches that can do more than sophisticated lexical-level matching. To accomplish this, we designed the dataset with three key challenges in mind. (i) The number of correct answer-options for each question is not pre-specified. This removes the over-reliance of current approaches on answer-options and forces them to decide on the correctness of each candidate answer

[2] http://cogcomp.org/multirc/

independently of others. In other words, unlike previous work, the task here is not to simply identify the best answer-option, but to evaluate the correctness of each answer-option individually. For example, the first question in Figure 19 can be answered by combining information from sentences 3, 5, 10, 13 and 15. It requires not only understanding that the stalker's name is Timothy but also that he is the man who Mary had hit. (ii) The correct answer(s) is not required to be a span in the text. For example, the correct answer, A, of the second question in Figure 19 is not present in the paragraph verbatim. It is instead a combination of two spans from 2 sentences: 12 and 13. Such answer-options force models to process and understand not only the paragraph and the question but also the answer-options. (iii) The paragraphs in our dataset have diverse provenance by being extracted from 7 different domains such as news, fiction, historical text etc., and hence are expected to be more diverse in their contents as compared to single-domain datasets. We also expect this to lead to diversity in the types of questions that can be constructed from the passage.

Overall, we introduce a reading comprehension dataset that significantly differs from most other datasets available today in the following ways:

- $\sim 6k$ high-quality multiple-choice RC questions that are generated (and manually verified via crowdsourcing) to require integrating information from multiple sentences.
- The questions are not constrained to have a *single* correct answer, generalizing existing paradigms for representing answer-options.
- Our dataset is constructed using 7 different sources, allowing more diversity in content, style, and possible question types.
- We show a significant performance gap between current solvers and human performance, indicating an opportunity for developing sophistical reasoning systems.

6.2. Relevant Work

Some recent datasets proposed for machine comprehension pay attention to type of questions and reasoning required. For example, RACE (Lai et al., 2017) attempts to incorporate different types of reasoning phenomena, and MCTest (Richardson et al., 2013) attempted to contain at least 50% multi-sentence reasoning questions. However, since the crowdsourced workers who created the dataset were only encouraged, and not required, to write such questions, it is not clear how many of these questions actually require multi-sentence reasoning (see Sec. 6.3.5). Similarly, only about 25% of question in the RACE dataset require multi-sentence reasoning as reported in their paper. Remedia (Hirschman et al., 1999) also contains 5 different types of questions (based on question words) but is a much smaller dataset. Other datasets which do not deliberately attempt to include multi-sentence reasoning, like SQuAD (Rajpurkar et al., 2016) and the CNN/Daily Mail dataset (Hermann et al., 2015), suffer from even lower percentage of such questions (12% and 2% respectively (Lai et al., 2017)). There are several other corpora which do not guarantee specific reasoning types, including MS MARCO (Nguyen et al., 2016), WikiQA (Yang et al., 2015), and TriviaQA (Joshi et al., 2017).

The complexity of reasoning required for a reading comprehension dataset would depend on several factors such as the source of questions or paragraphs; the way they are generated; and the order in which they are generated (i.e. questions from paragraphs, or the reverse). Specifically, paragraphs' source could influence the complexity and diversity of the language of the paragraphs and questions, and hence the required level of reasoning capabilities. Unlike most current datasets which rely on only one or two sources for their paragraphs (e.g. CNN/Daily Mail and SQuAD rely only on news and Wikipedia articles respectively) our dataset uses 7 different domains.

Another factor that distinguishes our dataset from previously proposed corpora is the way answers are represented. Several datasets represent answers as multiple-choices with a single correct answer. While multiple-choice questions are easy to grade, coming up with non-

trivial correct and incorrect answers can be challenging. Also, assuming exactly one correct answer (e.g., as in MCTest and RACE) inadvertently changes the task from choosing the correct answer to choosing the most likely answer. Other datasets (e.g MS-MARCO and SQuAD) represent answers as a contiguous substring within the passage. This assumption of the answer being a span of the paragraph, limits the questions to those whose answer is contained verbatim in the paragraph. Unfortunately, it rules out more complicated questions whose answers are only implied by the text and hence require a deeper understanding. Because of these limitations, we designed our dataset to use multiple-choice representations, but without specifying the number of correct answers for each question.

6.3. Construction of MultiRC

In this section we describe our principles and methodology of dataset collection. This includes automatically collecting paragraphs, composing questions and answer-options through crowd-sourcing platform, and manually curating the collected data. We also summarize a pilot study that helped us design this process, and end with a summary of statistics of the collected corpus.

6.3.1. Principles of design

Questions and answers in our dataset are designed based on the following key principles:

Multi-sentenceness. Questions in our challenge require models to use information from multiple sentences of a paragraph. This is ensured through explicit validation. We exclude any question that can be answered based on a single sentence from a paragraph.

Open-endedness. Our dataset is not restricted to questions whose answer can be found verbatim in a paragraph. Instead, we provide a set of hand-crafted answer-options for each question. Notably, they can represent information that is not explicitly stated in the text but is only inferable from it (e.g. implied counts, sentiments, and relationships).

Answers to be judged independently. The total number of answer options per question is variable in our data and we explicitly allow multiple correct and incorrect answer options (e.g. 2 correct and 1 incorrect options). As a consequence, correct answers cannot be guessed solely by a process of elimination or by simply choosing the best candidates out of the given options.

Through these principles, we encourage users to explicitly model the semantics of text beyond individual words and sentences, to incorporate extra-linguistic reasoning mechanisms, and to handle answer options independently of one another.

Variability. We encourage variability on different levels. Our dataset is based on paragraphs from multiple domains, leading to linguistically diverse questions and answers. Also, we do not impose any restrictions on the questions, to encourage different forms of reasoning.

6.3.2. Sources of documents

The paragraphs used in our dataset are extracted from various sources. Here is the complete list of the text types and sources used in our dataset, and the number of paragraphs extracted from each category (indicated in square brackets on the right):

1. News: [121]
    - CNN (Hermann et al., 2015)
    - WSJ (Ide et al., 2008)
    - NYT (Ide et al., 2008)
2. Wikipedia articles [92]
3. Articles on society, law and justice (Ide and Suderman, 2006) [91]
4. Articles on history and anthropology (Ide et al., 2008) [65]
5. Elementary school science textbooks [3] [153]
6. 9/11 reports (Ide and Suderman, 2006) [72]

[3] https://www.ck12.org

7. Fiction: [277]

   - Stories from the Gutenberg project
   - Children stories from MCTest (Richardson et al., 2013)
   - Movie plots from CMU Movie Summary corpus (Bamman et al., 2013)

From each of the above-mentioned sources we extracted paragraphs that had enough content. To ensure this we followed a 3-step process. In the first step we selected top few sentences from paragraphs such that they contained 1k-1.5k characters. To ensure coherence, all sentences were contiguous and extracted from the same paragraph. In this process we also discarded paragraphs that seemed to deviate too much from third person narrative style. For example, while processing Gutenberg corpus we considered files that had at least 5k lines because we found that most of them were short poetic texts. In the second step, we annotated (Khashabi et al., 2018c) the paragraphs and automatically filtered texts using conditions such as the average number of words per sentence; number of named entities; number of discourse connectives in the paragraph. These were designed by the authors of this paper after reviewing a small sample of paragraphs. A complete set of conditions is listed in Table 22. Finally in the last step, we manually verified each paragraph and filtered out the ones that had formatting issues or other concerns that seemed to compromise their usability.

Condition	bound
Number of sentences	≥ 6 & ≤ 18
Number of NER(CoNLL) mentions	≥ 2
Avg. number of NER(CoNLL) mentions	≥ 0.2
Number of NER(Ontonotes) mentions	≥ 4
Avg. number of NER(Ontonotes) mentions	≥ 0.25
Avg. number of words per sentence	≥ 5
Number of coreference mentions	≥ 3
Avg. number of coreference mentions	≥ 0.1
Number of coreference relations	≥ 3
Avg. number of coreference relations	≥ 0.08
Number of coreference chains	≥ 2
Avg. number of coreference chains	≥ 0.1
Number of discourse markers	≥ 2

Table 22: Bounds used to select paragraphs for dataset creation.

6.3.3. Pipeline of question extraction

In this section, we delineate details of the process for collecting questions and answers. Figure 20 gives a high-level idea of the process. The first two steps deal with creating multi-sentence questions, followed by two steps for construction of candidate answers. Interested readers can find more details on set-ups of each step in Appendix I.

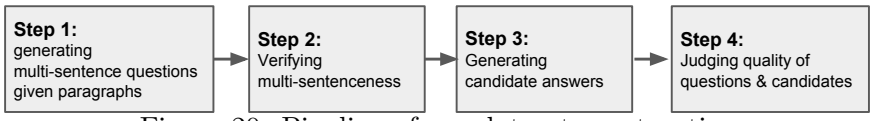


Figure 20: Pipeline of our dataset construction.

Step 1: Generating questions. The goal of the first step of our pipeline is to collect multi-sentence questions. We show each paragraph to 5 turkers and ask them to write 3-5 questions such that: (1) the question is answerable from the passage, and (2) only those questions are allowed whose answer cannot be determined from a single sentence. We clarify this point by providing example paragraphs and questions. In order to encourage turkers to write meaningful questions that fit our criteria, we additionally ask them for a correct answer and for the sentence indices required to answer the question. To ensure the grammatical quality of the questions collected in this step, we limit the turkers to the countries with English as their major language. After the acquisition of questions in this step, we filter out questions which required less than 2 or more than 4 sentences to be answered; we also run them through an automatic spell-checker[4] and manually correct questions regarding typos and unusual wordings.

Step 2: Verifying multi-sentenceness of questions. In a second step, we verify that each question can only be answered using more than one sentence. For each question collected in the previous step, we create question-sentence pairs by pairing it with each of the sentences necessary for answering it as indicated in the previous step. For a given question-sentence pair, we then ask turkers to annotate if they could answer the question from the sentence it is paired with (binary annotation). The underlying idea of this step

[4]Grammarly: www.grammarly.com

is that a multi-sentence question would not be answerable from a single sentence, hence turkers should not be able to give a correct answer for any of the question-sentence pair. Accordingly, we determine a question as requiring multiple sentences only if the correct answer cannot be guessed from any single question-sentence pair. We collected at least 3 annotations per pair, and to avoid sharing of information across sentences, no two pairs shown to a turker came from the same paragraph. We aggregate the above annotations for each question-answer pair and retain only those questions for which no pair was judged as answerable by a majority of turkers.

Step 3: Generating answer-options. In this step, we collect answer-options that will be shown with each question. Specifically, for each verified question from the previous steps, we ask 3 turkers to write as many correct and incorrect answer options as they can think of. In order to not curb creativity, we do not place a restriction on the number of options they have to write. We explicitly ask turkers to design difficult and non-trivial incorrect answer-options (e.g. if the question is about a person, a non-trivial incorrect answer-option would be other people mentioned in the paragraph).

After this step, we perform a light clean up of the candidate answers by manually correcting minor errors (such as typos), completing incomplete sentences and rephrasing any ambiguous sentences. We further make sure there is not much repetition in the answer-options, to prevent potential exploitation of correlation between some candidate answers in order to find the correct answer. For example, we drop obviously duplicate answer-options (i.e. identical options after lower-casing, lemmatization, and removing stop-words).

Step 4: Verifying quality of the dataset. This step serves as the final quality check for both questions and the answer-options generated in the previous steps. We show each paragraph, its questions, and the corresponding answer-options to 3 turkers, and ask them to indicate if they find any errors (grammatical or otherwise), in the questions and/or answer-options. We then manually review, and correct if needed, all erroneous questions

and answer-options. This ensures that we have meaningful questions and answer-options. In this step, we also want to verify that the correct (or incorrect) options obtained from Step 3 were indeed correct (or incorrect). For this, we additionally ask the annotators to select all correct answer-options for the question. If their annotations did not agree with the ones we had after Step 3 (e.g. if they unanimously selected an 'incorrect' option as the answer), we manually reviewed and corrected (if needed) the annotation.

6.3.4. Pilot experiments

The 4-step process described above was a result of detailed analysis and substantial refinement after two small pilot studies.

In the first pilot study, we ran a set of 10 paragraphs extracted from the CMU Movie Summary Corpus through our pipeline. Our then pipeline looked considerably different from the one described above. We found the steps that required turkers to write questions and answer-options to often have grammatical errors, possibly because a large majority of turkers were non-native speakers of English. This probslem was more prominent in questions than in answer-options. Because of this, we decided to limit the task to native speakers. Also, based on the results of this pilot, we overhauled the instructions of these steps by including examples of grammatically correct—but undesirable (not multi-sentence)—questions and answer-options, in addition to several minor changes.

Thereafter, we decided to perform a manual validation of the verification steps (current Steps 2 and 4). For this, we (the authors of this paper) performed additional annotations ourselves on the data shown to turkers, and compared our results with those provided by the turkers. We found that in the verification of answer-options, our annotations were in high agreement (98%) with those obtained from mechanical turk. However, that was not the case for the verification of multi-sentence questions. We made several further changes to the first two steps. Among other things, we clarified in the instructions that turkers should not use their background knowledge when writing and verifying questions, and also

included negative examples of such questions. Additionally, when turkers judged a question to be answerable using a single sentence, we decided to encourage (but not require) them to guess the answer to the question. This improved our results considerably, possibly because it forced annotators to think more carefully about what the answer might be, and whether they *actually* knew the answer or they just *thought* that they knew it (possibly because of background knowledge or because the sentence contained a lot of information relevant to the question). Guessed answers in this step were only used to verify the validity of multi-sentence questions. They were not used in the dataset or subsequent steps.

After revision, we ran a second pilot study in which we processed a set of 50 paragraphs through our updated pipeline. This second pilot confirmed that our revisions were helpful, but thanks to its larger size, also allowed us to identify a couple of borderline cases for which additional clarifications were required. Based on the results of the second pilot, we made some additional minor changes and then decided to apply the pipeline for creating the final dataset.

6.3.5. Verifying multi-sentenceness

While collecting our dataset, we found that, even though Step 1 instructed turkers to write multi-sentence questions, not all generated questions indeed required multi-sentence reasoning. This happened even after clarifications and revisions to the corresponding instructions, and we attribute it to honest mistakes. Therefore, we designed the subsequent verification step (Step 2).

There are other datasets which aim to include multi-sentence reasoning questions, especially MCTest. Using our verification step, we systematically verify their multi-sentenceness. For this, we conducted a small pilot study on about 60 multi-sentence questions from MCTest. As for our own verification, we created question-sentence pairs for each question and asked annotators to judge whether they can answer a question from the single sentence shown. Because we did not know which sentences contain information relevant to a question, we

created question-sentence pairs using all sentences from a paragraph. After aggregation of turker annotations, we found that about half of the questions annotated as multi-sentence could be answered from a single sentence of the paragraph. This study, though performed on a subset of the data, underscores the necessity of rigorous verification step for multi-sentence reasoning when studying this phenomenon.

6.3.6. Statistics on the dataset

We now provide a brief summary of MULTIRC. Overall, it contains roughly $\sim 6k$ multi-sentence questions collected for about +800 paragraphs.[5] The median number of correct and total answer options for each question is 2 and 5, respectively. Additional statistics are given in Table 23.

In Step 1, we also asked annotators to identify sentences required to answer a given question. We found that answering each question required 2.4 sentences on average. Also, required sentences are often not contiguous, and the average distance between sentences is 2.4. Next, we analyze the types of questions in our dataset. Figure 22 shows the count of first word(s) for our questions. We can see that while the popular question words (*What*, *Who*, etc.) are very common, there is a wide variety in the first word(s) indicating a diversity in question types. About 28% of our questions require binary decisions (true/false or yes/no).

Parameter	Value
# of paragraphs	871
# of questions	9,872
# of multi-sentence questions	5,825
avg # of candidates (per question)	5.44
avg # of correct answers (per question)	2.58
avg paragraph length (in sentences)	14.3 (4.1)
avg paragraph length (in tokens)	263.1 (92.4)
avg question length (in tokens)	10.9 (4.8)
avg answer length (in tokens)	4.7 (5.5)
% of yes/no/true/false questions	27.57%
avg # of sent. used for questions	2.37 (0.63)
avg distance between the sent.'s used	2.4 (2.58)
% of correct answers verbatim in paragraph	34.96%
% of incorrect answers verbatim in paragraph	25.84%

Table 23: Various statistics of our dataset. Figures in parentheses represent standard deviation.

We randomly selected 60 multi-sentence questions from our corpus and asked two indepen-

[5] We will also release the $3.7k$ questions that did not pass Step 2. Though not multi-sentence questions, they could be a valuable resource on their own.

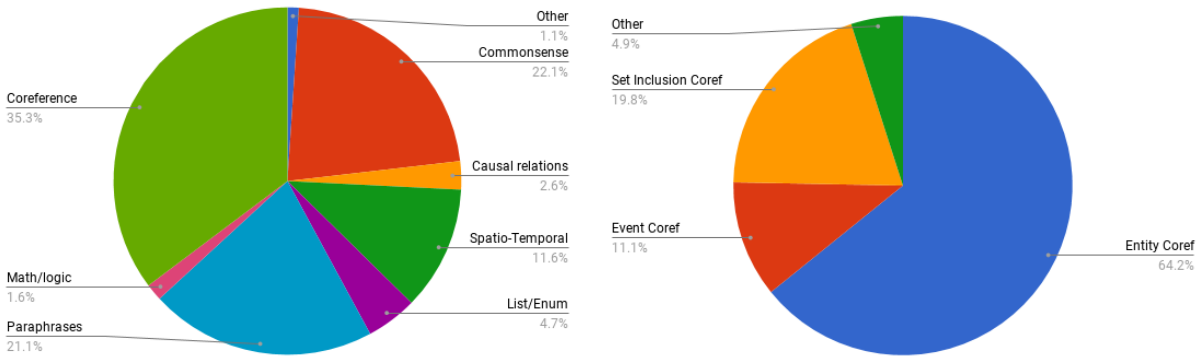


Figure 21: Distribution of (left) general phenomena; (right) variations of the "coreference" phenomena.

dent annotators to label them with the type of reasoning phenomenon required to answer them.[6] During this process, the annotators were shown a list of common reasoning phenomena (shown below), and they had to identify one or more of the phenomena relevant to a given question. The list of phenomena shown to the annotators included the following categories: mathematical and logical reasoning, spatio-temporal reasoning, list/enumeration, coreference resolution (including implicit references, abstract pronouns, event coreference, etc.), causal relations, paraphrases and contrasts (including lexical relations such as synonyms, antonyms), commonsense knowledge, and 'other'. The categories were selected after a manual inspection of a subset of questions by two of the authors. The annotation process revealed that answering questions in our corpus requires a broad variety of reasoning phenomena. The left plot in Figure 21 provides detailed results.

The figure shows that a large fraction of questions require coreference resolution, and a more careful inspection revealed that there were different types of coreference phenomena at play here. To investigate these further, we conducted a follow-up experiment in which manually annotated all questions that required coreference resolution into finer categories. Specifically, each question was shown to two annotators who were asked to select one or more of the following categories: entity coreference (between two entities), event coreference (between two events), set inclusion coreference (one item is part of or included in the

[6]The annotations were adjudicated by two authors of this paper.

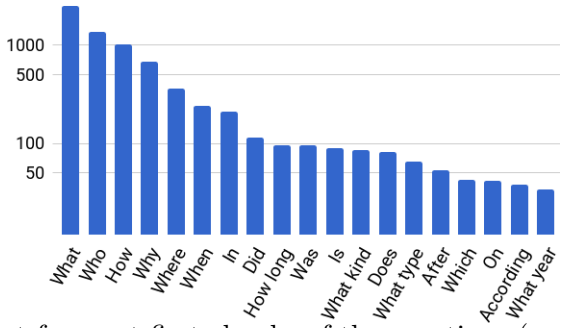


Figure 22: Most frequent first chunks of the questions (counts in log scale).

other) and 'other'. Figure 21 (right) shows the results of this experiment. We can see that, as expected, entity coreference is the most common type of coreference resolution needed in our corpus. However, a significant number of questions also require other types of coreference resolution. We provide some examples of questions along with the required reasoning phenomena in Appendix II.

6.4. Analysis

In this section, we provide a quantitative analysis of several baselines for our challenge.

Evaluation Metrics. We define precision and recall for a question q as: $\text{Pre}(q) = \frac{|A(q) \cap \hat{A}(q)|}{|\hat{A}(q)|}$ and $\text{Rec}(q) = \frac{|A(q) \cap \hat{A}(q)|}{|A(q)|}$, where $A(q)$ and $\hat{A}(q)$ are the sets of correct and selected answer-options. We define (macro-average) F1_m as the harmonic mean of average-precision $avg_{q \in Q}(\text{Pre}(q))$ and average-recall $avg_{q \in Q}(\text{Rec}(q))$ with Q as the set of all questions.

Since by design, each answer-option can be judged independently, we consider another metric, F1_a, evaluating binary decisions on all the answer-options in the dataset. We define F1_a to be the harmonic mean of $\text{Pre}(Q)$ and $\text{Rec}(Q)$, with $\text{Pre}(Q) = \frac{|A(Q) \cap \hat{A}(Q)|}{|\hat{A}(Q)|}$; $A(Q) = \bigcup_{q \in Q} A(q)$; and similar definitions for $\hat{A}(Q)$ and $\text{Rec}(Q)$.

6.4.1. Baselines

Human. Human performance provides us with an estimate of the best achievable results on datasets. Using mechanical turk, we ask 4 people (limited to native speakers) to solve

our data. We evaluate score of each label by averaging the decision of the individuals.

Random. To get an estimate on the lower-bound we consider a random baseline, where each answer option is selected as correct with a probability of 50% (an unbiased coin toss). The numbers reported for this baseline represent the expected outcome (statistical expectation).

IR (information retrieval baseline). This baseline selects answer-options that best match sentences in a text corpus (Clark et al., 2016). Specifically, for each question q and answer option a_i, the IR solver sends $q + a_i$ as a query to a search engine (we use Lucene) on a corpus, and returns the search engine's score for the top retrieved sentence s, where s must have at least one non-stopword overlap with q, and at least one with a_i.

We create two versions of this system. In the first variation IR(paragraphs) we create a corpus of sentences extracted from all the paragraphs in the dataset. In the second variation, IR(web) in addition to the knowledge of the paragraphs, we use extensive external knowledge extracted from the web (Wikipedia, science textbooks and study guidelines, and other webpages), with 5×10^{10} tokens (280GB of plain text).

SurfaceLR (logistic regression baseline). As a simple baseline that makes use of our small training set, we reimplemented and trained a logistic regression model using word-based overlap features. As described in (Merkhofer et al., 2018), this baseline takes into account the lengths of a text, question and each answer candidate, as well as indicator features regarding the (co-)occurrences of any words in them.

SemanticILP (semi-structured baseline). This state-of-the-art solver, originally proposed for science questions and biology tests, uses a semi-structured representation to formalize the scoring problem as a subgraph optimization problem over multiple layers of semantic abstractions (Khashabi et al., 2018b). Since the solver is designed for multiple-choice with single-correct answer, we adapt it to our setting by running it for each answer-option.

Specifically for each answer-option, we create a single-candidate question, and retrieve a real-valued score from the solver.

BiDAF (neural network baseline). As a neural baseline, we apply this solver by Seo et al. (2016), which was originally proposed for SQuAD but has been shown to generalize well to another domain (Min et al., 2017). Since BiDAF was designed for cloze style questions, we apply it to our multiple-choice setting following the procedure by Kembhavi et al. (2017): Specifically, we score each answer-option by computing the similarity value of it's output span with each of the candidate answers, computed by phrasal similarity tool of Wieting et al. (2015).

	Dev		Test	
	$F1_m$	$F1_a$	$F1_m$	$F1_a$
Random	44.3	43.8	47.1	47.6
IR(paragraphs)	64.3	60.0	54.8	53.9
SurfaceLR	66.1	63.7	66.7	63.5
Human	86.4	83.8	84.3	81.8

Table 24: Performance comparison for different baselines tested on a subset of our dataset (in percentage). There is a significant gap between the human performance and current statistical methods.

6.4.2. Results

To get a sense of our dataset's hardness, we evaluate both human performance and multiple computational baselines. Each baseline scores an answer-option with a real-valued score, which we threshold to decide whether an answer option is selected or not, where the threshold is tuned on the development set. Table 24 shows performance results for different baselines. The significantly high human performance shows that humans do not have much difficulties in answering the questions. Similar observations

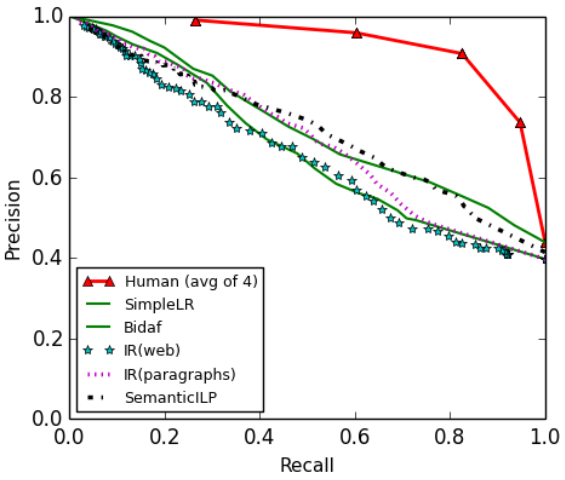


Figure 23: PR curve for each of the baselines. There is a considerable gap with the baselines and human.

can be made in Figure 23 where we plot
$avg_{q \in Q}(\text{Pre}(q))$ vs. $avg_{q \in Q}(\text{Rec}(q))$, for different threshold values.

6.5. Summary

To motivate the community to work on more challenging forms of natural language comprehension, in this chapter we discussed a dataset that requires reasoning over multiple sentences. We solicit and verify questions and answers for this challenge through a 4-step crowdsourcing experiment. Our challenge dataset contains $\sim 6k$ questions for +800 paragraphs across 7 different domains (elementary school science, news, travel guides, fiction stories, etc) bringing in linguistic diversity to the texts and to the questions wordings. On a subset of our dataset, we found human solvers to achieve an F1-score of 86.4%. We analyze a range of baselines, including a recent state-of-art reading comprehension system, and demonstrate the difficulty of this challenge, The dataset is the first to study multi-sentence inference at scale, with an open-ended set of question types that requires reasoning skills.

An additional important aspect of this work is that we challenged the community to change the harmful "fixed" test set methodology, and committed to updating the test set every few months.

CHAPTER 7 : A Question Answering Benchmark for Temporal Common-sense

"Everything changes and nothing stands still."

— Heraclitus, 535 BC - 475 BC

7.1. Overview

Automating natural language understanding requires models that are informed by commonsense knowledge and the ability to reason with it in both common and unexpected situations. The NLP community has started in the last few year to investigate how to acquire such knowledge Forbes and Choi (2017); Zhang et al. (2017); Yang et al. (2018); Rashkin et al. (2018); Bauer et al. (2018); Tandon et al. (2018); Zellers et al. (2018).

This work studies a specific type of commonsense, *temporal commonsense*.[1] For instance, given two events *"going on a vacation"* and *"going for a walk,"* most humans would know that a vacation is typically longer and occurs less often than a walk, but our programs currently do not know that.

Temporal commonsense has received limited attention so far. **Our first contribution** is that, to the best of our knowledge, we are the first to systematically study and quantify performance on a range of temporal commonsense phenomena. Specifically, we consider five temporal properties: *duration* (how long an event takes), *temporal ordering* (typical order of events), *typical time* (when an event happens), *frequency* (how often an event occurs), and *stationarity* (whether a state holds for a very long time). Previous works have investigated some of them, either explicitly or implicitly (e.g., duration DivyeKhilnani and Jurafsky (2011); Williams (2012) and ordering Chklovski and Pantel (2004); Ning et al. (2018a)), but none of them have defined or studied all aspects of temporal commonsense in a unified framework. Kozareva and Hovy (2011) came close, when they defined a few temporal aspects to be investigated, but failed short of distinguishing in text and quantifying

[1] This chapter is based on the following publication: Zhou et al. (2019).

S1: *Growing up on a farm near St. Paul, L. Mark Bailey didn't dream of becoming a judge.*
Q1: *Is Mark still on the farm now?*
[x] no [ ] yes
Reasoning type: *stationarity*

S2: *The massive ice sheet, called a glacier, caused the features on the land you see today.*
Q2: *When did the glacier start to impact the land's features?*
[x] centuries ago [ ] hours ago
[ ] 10 years ago [x] tens of millions of years ago
Reasoning type: *event typical time*

S3: *Carl Laemmle, head of Universal Studios, gave Einstein a tour of his studio and introduced him to Chaplin.*
Q3: *How long did the tour last?*
[ ] 9 hours [ ] 15 days
[x] 45 minutes [ ] 5 seconds
Reasoning type: *event duration*

S4: *Mr. Barco has refused U.S. troops or advisers but has accepted U.S. military aid.*
Q4: *What happened after Mr. Barco accepted the military aid?*
[ ] the aid was denied [x] things started to progress
[x] he received the aid
Reasoning type: *event ordering*

S5: *The Minangkabau custom of freely electing their leaders provided the model for rulership elections in modern federal Malaysia.*
Q5: *How often are the elections held?*
[ ] every day [ ] every month
[x] every 4 years [ ] every 100 years
Reasoning type: *event frequency*

Figure 24: Five types of temporal commonsense in TacoQA. Note that a question may have multiple answers.

performances on these.

Given the lack of an evaluation standards and datasets for temporal commonsense, **our second contribution** is the collection of a new dataset dedicated for it, TacoQA (short for **t**empor**a**l **co**mmon-sense **q**uestion **a**nswering).[2] TacoQA is constructed via crowdsourcing with three meticulously-designed stages to guarantee its quality. An entry in TacoQA contains a *sentence* providing context information, a *question* requiring temporal commonsense, and *candidate answers* with or without correct ones (see Fig. 24). More details about TacoQA are in Sec. 7.3.

Our third contribution is that we propose multiple systems, including *ESIM*, BERT and their variants, for this task. TacoQA allows us to investigate how state-of-the-art NLP techniques do on temporal commonsense tasks. Results in Sec. 7.4 show that, despite a significant improvement over random-guess baselines, BERT is still far behind human performance on temporal commonsense reasoning, indicating that existing NLP techniques still have limited capability of capturing high-level semantics like time.

7.2. Related Work

commonsense has been a very popular topic in recent years and existing NLP works have mainly investigated the acquisition and evaluation of commonsense in the physical world, including but not limited to, size, weight, and strength Forbes and Choi (2017), roundness and deliciousness Yang et al. (2018), and intensity Cocos et al. (2018). In terms of commonsense on "events", Rashkin et al. (2018) investigated the intent and reaction of participants of an event, and Zellers et al. (2018) tried to select the most likely subsequent event. As far as we know, no existing work has focused on temporal commonsense yet.

There have also been many works trying to understand time in natural language but not necessarily with respect to commonsense, such as the extraction and normalization of temporal expressions Lee et al. (2014), temporal relation extraction Ning et al. (2018b), and

[2]The dataset and code will be released upon publication.

Measure		Value
# of unique questions		1893
# of unique question-answer pairs		13,225
avg. sentence length		17.8
avg. question length		8.2
avg. answer length		3.3
Category	# questions	avg # of candidate
event frequency	433	8.5
event duration	440	9.4
event stationarity	279	3.1
event ordering	370	5.4
event typical time	371	6.8

Table 25: Statistics of TacoQA.

timeline construction Leeuwenberg and Moens (2018). Among these, some works are implicitly on temporal commonsense, such as event durations Williams (2012); Vempala et al. (2018), typical temporal ordering Chklovski and Pantel (2004); Ning et al. (2018a), and script learning (i.e., what happens next after certain events) Granroth-Wilding and Clark (2016); Li et al. (2018). However, either in terms of datasets or approaches, existing works did not study all five types of temporal commonsense in a unified framework as we do here.

Instead of working on each individual aspect of temporal commonsense, we formulate the problem as a machine reading comprehension task in the format of question-answering (QA). The past few years have also seen significant progress on QA Clark et al. (2018); Ostermann et al. (2018); Merkhofer et al. (2018), but mainly on *general* natural language comprehension tasks without tailoring it to test *specific* reasoning capabilities such as temporal commonsense. Therefore, a new dataset like TacoQA is strongly desired.

7.3. Construction of TacoQA

We describe our crowdsourcing scheme for TacoQA that is designed after extensive pilot studies. The multi-step scheme asks annotators to generate questions, validate questions, and then label candidate answers. We use Amazon Mechanical Turk and restrict our tasks to English-speakers only. Before working on our task, annotators need to read through our

guidelines and pass a qualification test designed to ensure their understandings.[3]

Step 1: Question generation. In the first step, we ask crowdsourcers to generate questions given a sentence. We randomly select 630 sentences from MultiRC Khashabi et al. (2018a) (70 from each of the 9 domains) as input sentences. To make sure that the questions indeed require temporal commonsense knowledge, we instruct annotators to follow two requirements when generating questions: (a) "temporal" questions, from one of our five categories (see Fig. 24); (b) not having direct answers mentioned in the given sentence. We also ask annotators to provide a correct answer for each of their questions to make sure that the questions are answerable at least by themselves.

Step 2: Question verification. To improve the quality of the questions generated in Step 1, we further ask two different annotators to check (a) whether the two requirements above are satisfied and (b) whether there exist grammatical or logical errors. We keep a question if and only if both annotators agree on its quality; since the annotator who provided the question in Step 1 also agrees on it, this leads to a [3/3] agreement for each question. For the questions that we keep, we continue to ask annotators to give one correct answer and one incorrect answer, which serve as a seed set for automatic answer expansion in the next step.

Step 3: Candidate answer expansion. In the previous steps, we have collected 3 positive and 2 negative answers for each question.[4] Step 3 aims to automatically expand this set of candidate answers by three approaches. First, we use a set of rules to extract temporal terms (e.g. "a.m.", "1990", "afternoon", "day"), or numbers and quantities ("2", "once"), which are replaced by terms randomly selected from a list of temporal units ("second"), adjectives ("early"), points ("a.m.") and adverbs ("always"); see the appendix for more details. Examples are "2 a.m." → "3 p.m.", "1 day" → "10 days", "once a week"→

[3]Our dataset and some related details (such as, our annotation interfaces, guidelines and qualification tests) are available at the following link: `https://bit.ly/2tZ1mkd`

[4]One positive answer from Step 1; one positive and one negative answer from each of the two annotators in Step 2.

"twice a month". Second, we mask each individual token in a candidate answer and use BERT Devlin et al. (2018) to predict them; we rank those predictions by the confidence level of BERT and keep the top ones. Third, for those candidates representing events, typically there are no temporal terms in them. We then create a pool of $60k$ event phrases using PropBank Kingsbury and Palmer (2002), and retrieve the most similar ones to a given candidate answer using an information retrieval (*IR*) system.[5] We use the three approaches sequentially to expand the candidate answer set to 20 candidates per question.

Step 4: Answer labeling. In this step, we ask annotators to label each answer with three options: "likely", "unlikely", or "garbage" (incomplete or meaningless phrases). We keep a candidate answer if and only if all 4 annotators agree on "likely" or "unlikely", and "garbage" is not marked by any annotator. We also discard any questions that end up with no valid candidate answers. Finally, the statistics of TacoQA is in Table 25.

7.4. Experiments

We assess the quality of our dataset using a couple of baseline systems. We create a uniform split of 30%/70% of the data to dev/test. The rationale behind this split is that, a successful system has to bring in a huge amount of world knowledge and derive commonsense understandings *prior* to the current task evaluation. We therefore believe that it make no sense to expect a system to *train* solely on this data, and we think of the development data as only providing a *definition* of the task. Indeed, the gains from our development data are marginal after a certain number of observations. This intuition has been studied and verified in Appendix A.5.2.

Evaluation metrics. Two question-level metrics are adopted in this work: exact match (*EM*) and *F1*. EM measures in how many questions a system is able to correctly label all candidate answers, while *F1* measures the average overlap between one's predictions and the ground truth (see Appendix A.5.3 for full definition).

[5] `www.elastic.co`

Human performance. An expert annotator also worked on TacoQA to gain a better understanding of the human performance on it. The expert specifically answered 100 questions randomly sampled from the test set, and could only see a single answer at a time, with its corresponding question and sentence.

Systems. We propose to use two state-of-the-art systems in machine reading comprehension that are suitable for our task. *ESIM* Chen et al. (2017) is a neural model effective on natural language inference. We initialize the word embeddings in *ESIM* via either *GloVe* Pennington et al. (2014) or *ELMo* Peters et al. (2018) to demonstrate the effect of pre-training in this task. BERT is a recent state-of-the-art contextualized representation used in a broad range of high-level tasks Devlin et al. (2018). We also add unit normalization to BERT, which extracts and converts temporal expressions in candidate answers to their most proper units. For example, "30 months" will be converted to "2.5 years".

Experimental setting. In both *ESIM* baselines, we model the process as a sentence-pair labeling task, following the *SNLI* setting provided in AllenNLP.[6] In both versions of *BERT*, we use the same sequence pair classification model and the same parameters as in *BERT*'s *GLUE* experiment.[7] A system receives two phrases at a time: (a) the concatenation of the sentence and question, and (b) the answer. The system makes a binary prediction on each instance, positive or negative.

Results and discussion. Table 26 provides a summary of the results on TacoQA, where we compare the *ESIM* and BERT baselines, along with a few naive baselines (always-positive, always-negative, uniformly random), to the human performance. The significant improvement brought by contextualized pre-training such as BERT and *ELMo* indicates that a significant portion of commonsense knowledge is actually acquired via pre-training. We can also see that human annotators achieved a very high performance under both metrics, indicating the high agreement level humans are on for this task. Our baselines,

[6] https://github.com/allenai/allennlp
[7] github.com/huggingface/pytorch-pretrained-BERT

System	*F1*	*EM*
Random	36.2	8.1
Always Positive	49.8	12.1
Always Negative	17.4	17.4
ESIM + GloVe	50.3	20.9
ESIM + ELMo	54.9	26.4
BERT	66.1	39.6
BERT + unit normalization	**69.9**	**42.7**
Single Human	87.1	75.8

Table 26: Summary of the performances for different baselines. All numbers are in percentages.

including *BERT*, still fall behind the human performance with a significantly margin.

Further analysis shows that *BERT*, as a language model, is good at associating surface-forms (e.g. associating "sunrise" and "morning" since they often co-occur), which is highly sensitive to *units* (*days, years, etc*). To address the high sensitivity, we added unit normalization on top of BERT, but even with normalization, *BERT+unit normalization* is still far behind the human performance. This implies that the information acquired by BERT is still not sufficient to solve this task. Moreover, the low EM scores show that the current systems do not truly understand time in those questions.

Figure 25 reveals that the performance of *BERT* is not uniform across different categories, which could stem from the nature of those different types of temporal commonsense, quality of the candidate answers, etc. For example, the number of candidates for stationarity questions are much smaller than those for other questions, leading to a relatively easy task, but the performance gain from a random baseline to BERT+normalization is not large, indicating that further improvement on stationarity is still difficult.

7.5. Summary

This chapter has focused on the challenge of temporal commonsense. Specifically, we framed it as a QA task, defined five categories of questions that capture such ability, and developed a novel crowdsourcing scheme to generate a high-quality dataset for temporal commonsense.

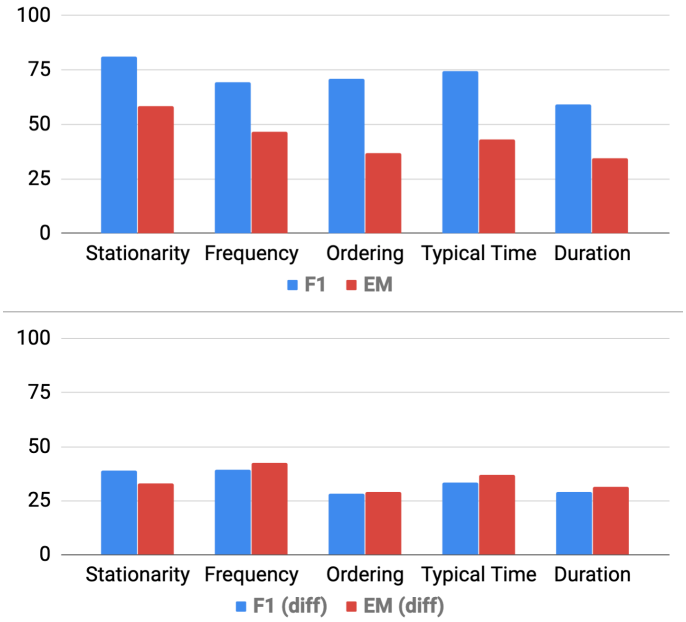


Figure 25: *BERT + unit normalization* performance per temporal reasoning category (top), performance gain over random baseline per category (bottom)

We then showed that systems equipped with state-of-the-art language models such as *ELMo* and BERT are still far behind humans, thus motivating future research in this area. Our analysis sheds light on the capabilities as well as limitations of current models. We hope that this study will inspire further research on temporal commonsense.

Part III

Formal Study of Reasoning in Natural Language

"Language is froth on the surface of thought."

— John McCarthy

8.1. Introduction

Reasoning can be defined as the process of combining facts and beliefs, in order to make decisions (Johnson-Laird, 1980). In particular, in natural language processing (NLP), it has been studied under various settings, such as question-answering (QA) (Hirschman et al., 1999).

While there is a rich literature on reasoning, there is little understanding of the nature of the problem and its limitations, especially in the context of natural language. In particular, there remains a sizable gap between empirical understanding of reasoning algorithms for language and the theoretical guarantees for their quality, often due to the complexity of the reality they operate on. An important challenge in many language understanding problems is the *symbol grounding problem* (Harnad, 1990), the problem of accurately mapping symbols into its underlying meaning representation. Practitioners often address this challenging by enriching their representations; for example by mapping textual information to Wikipedia entries (Mihalcea and Csomai, 2007; Ratinov et al., 2011), or grounding text to executable rules via semantic parsing (Reddy et al., 2017). Building upon such rep-

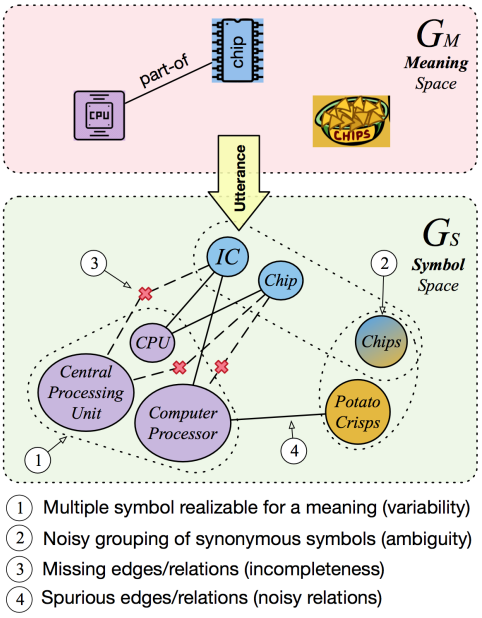


Figure 26: The interface between meanings and symbols: each meaning (top) can be *uttered* in many ways into symbolic forms (bottom).

resentations, has produced various reasoning systems that essentially work by combining local information.

This work introduces a formalism that incorporates elements of the symbol-grounding problem, via the two spaces illustrated in Figure 26, and sheds theoretical light on existing intuitions.[1] The formalism consists of (A) an abstract model of linguistic knowledge, and (B) a reasoning model.

(A) Linguistically-inspired abstract model: We propose a theoretical framework to model and study the capabilities/limitations of reasoning, especially when taking into account key difficulties that arise when formalizing linguistic reasoning. Our model uses two spaces; cf. Figure 26. We refer to the internal conceptualization in the human mind as the *meaning space*. We assume the information in this space is free of noise and uncertainty. In contrast to human thinking in this space, *human expression* of thought via the utterance of language introduces many imperfections. The information in this linguistic space—which we refer to as the *symbol space*—has many language-specific properties. The symbolic space is often redundant (e.g., multiple symbols "CPU" and "computer processor" express the same meaning), ambiguous (e.g., a symbol like "chips" could refer to multiple meanings), incomplete (relations between some symbolic nodes might be missing), and inaccurate (there might be incorrect edges). Importantly, this noisy symbol space is also what a machine reasoning algorithm operates in.

(B) Reasoning model: We define reasoning as the ability to infer the existence of properties of interest in the *meaning* space, by observing only its representation in the *symbol* space. The target property in the meaning graph is what characterizes the nature of the reasoning algorithm, e.g., are two nodes connected. While there are many flavors of reasoning (including *multi-hop reasoning*), in this first study, we explore a common primitive shared among various reasoning formalisms; namely, the *connectivity problem* between a pair of nodes in an undirected graph in the meaning space, while observing its noisy ver-

[1] This chapter is based on the following publication: Khashabi et al. (2019).

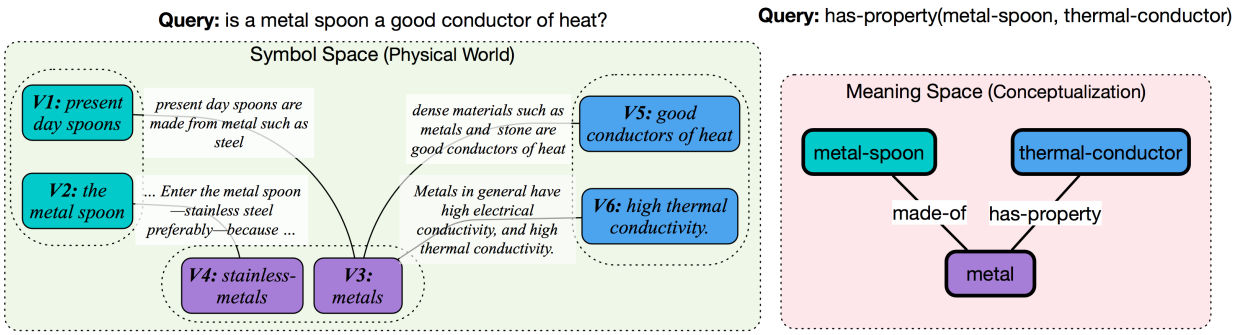


Figure 27: The meaning space contains [clean and unique] symbolic representation and the facts, while the symbol space contains [noisy, incomplete and variable] representation of the facts. We show sample meaning and symbol space nodes to answer the question: *Is a metal spoon a good conductor of heat?*.

sion in the symbol space. This simplification clarifies the exposition and the analysis,and we expect similar results to hold for a broader class of reasoning algorithms that rely on connectivity.

Figure 27 illustrates a reasoning setting where the semantics of the edges is included. Most humans understand that ***V1:****"present day spoons"* and ***V2:****"the metal spoons"* are equivalent nodes (have the same meaning). However, a machine has to infer this understanding. The semantics of the connection between nodes are expressed through natural language sentences. For example, connectivity could express the semantic relation between two nodes: `has-property(metal,thermal-conductor)`. However a machine may find it difficult to infer this fact from, say, reading text over the Internet as it may be expressed in many different ways, e.g., can be found in a sentence like *"dense materials such as [**V3:**]metals and stones are [**V5:**]good conductors of heat"*.

To ground this in existing efforts, consider *multi-hop* reasoning for QA systems (Khashabi et al., 2016; Jansen et al., 2018). Here the reasoning task is to connect local information, via multiple local "hops", in order to arrive at a conclusion. In the meaning graph, one can trace a path of locally connected nodes to verify the correctness of a query; for example the query `has-property(metal-spoon, thermal-conductor)` can be verified by tracing a sequence of nodes, as shown in Figure 27. In other words, answering queries can be cast as inferring the

existence of a path connecting two nodes m and m'. [2] While doing so on the meaning graph is straightforward, doing so on the noisy symbol graph is not. Intuitively, each local "hop" introduces more noise, allowing reliable inference to be performed only when it does not require too many steps in the underlying meaning space. To study this issue, one must quantify the effect of noise accumulation for long-range reasoning.

Contributions. We believe that this is the first work to provide a mathematical study of the challenges and limitations of reasoning algorithms in the presence of the symbol-meaning mapping challenge. We make three main contributions.

First, we establish a novel, linguistically motivated formal framework for analyzing the problem of reasoning about the ground truth (the meaning space) while operating over a noisy and incomplete linguistic representation (the symbol space). This framework allows one to derive rigorous intuitions about what various classes of reasoning algorithms *can* and *cannot* achieve.

Second, we study in detail the *connectivity reasoning* problem, in particular the interplay between the noise level in the symbol space (due to ambiguity, variability, and missing information) and the distance (in terms of inference steps, or hops) between two elements in the meaning space. We prove that under low noise levels, it is indeed possible to perform reliable connectivity reasoning up to a few hops (Theorem 1). On the flip side, even a moderate increase in the noise level makes it difficult to assess the connectivity of elements if they are logarithmic distance apart in the meaning space (Theorems 2 and 3). This finding is aligned with empirical observations of "semantic drift", i.e., substantial drop in performance beyond a few (usually 2-3) hops (Fried et al., 2015; Jansen, 2016).

Third, we apply the framework to a subset of a real-world knowledge-base, FB15k237, treated as the meaning graph, illustrating how key noise parameters influence the possibility (or not) of accurately solving the connectivity problem.

[2] This particular grounding is meant to help relate our graph-based formalism to existing applications, and is not the only way of realizing reasoning on graphs.

Classical views on reasoning. Philosophers, all the way from Aristotle and Avicenna, were the first ones to notice reasoning and rationalism (Kirk et al., 1983; Davidson, 1992). In modern philosophy, the earlier notions were mixed with mathematical logic, resulting in formal theories of reasoning, such as deductive, inductive, and abductive reasoning (Peirce, 1883). Our treatment of reasoning applies to all these, that can be modeled and executed using graphical representations.

Reasoning in AI literature. The AI literature has seen a variety of formalisms for automated reasoning. These include, reasoning with logical representations (McCarthy, 1963), semantic networks (Quillan, 1966), frame-semantic based systems (Fillmore, 1977), Bayesian networks (Pearl, 1988), among others.

It is widely believed that a key obstacle to progress has been the *symbol grounding problem* (Harnad, 1990; Taddeo and Floridi, 2005). Our formalism is directly relevant to this issue. We assume that symbols available to reasoning systems are results of communication meaning n natural language. This results in ambiguity since a given symbol could be mapped to multiple actual meanings but also in variablity (redundancy).

Reasoning for natural language comprehension. In the context of natural language applications (such as QA) flavors of linguistic theories are blended with the foundation provided by AI. A major roadblock has been the problem of *symbol grounding*, or grounding free-form texts to a higher-level meaning. Example proposals to deal with this issue are, extracting semantic parses (Kaplan et al., 1982; Steedman and Baldridge, 2011; Banarescu et al., 2013), linking to the knowledge bases (Mihalcea and Csomai, 2007), mapping to semantic frames (Punyakanok et al., 2004), etc. These methods can be thought of as approximate solutions for grounding symbolic information to some *meaning*. (Roth and Yih, 2004) suggested a general abductive framework that addresses it by connecting reasoning to models learned from data; it has been used in multiple NLP reasoning problems (Khashabi

et al., 2018b).

On the execution of reasoning with the disambiguated inputs there are varieties of proposals, e.g., using executable formulas (Reddy et al., 2017; Angeli and Manning, 2014), chaining relations to infer new relations (Socher et al., 2013; McCallum et al., 2017; Khot et al., 2017), and possible combinations of the aforementioned paradigms (Gardner et al., 2015; Clark et al., 2016). Our analysis covers any algorithm for inferring patterns that can be formulated in graph-based knowledge, e.g., chaining local information, often referred to as *multi-hop* reasoning (Jansen et al., 2016, 2018; Lin et al., 2018). For example, Jansen et al. (2017) propose a structured multi-hop reasoning by aggregating sentential information from multiple knowledge bases. The work shows that while this strategy improves over baselines with no reasoning (showing the effectiveness of reasoning), with aggregation of more than 2-3 sentences the quality declines (showing a limitation for reasoning). Similar observations were also made in (Khashabi et al., 2016). These empirical observations support the theoretical intuition proven in this work.

8.3. Background and Notation

We start with basic definitions and notation.

Graph Theory. We denote an undirected graph with $G(V, E)$ where V and E are the sets of nodes and edges, resp. We use the notations V_G and E_G to refer to the nodes and edges of a graph G, respectively. Let $\text{dist}(v_i, v_j)$ be the distance between nodes v_i and v_j in G. A *simple path* (henceforth referred to as just a *path*) is a sequence of adjacent nodes that does not have repeating nodes. Let $v_i \overset{d}{\rightsquigarrow} v_j$ denote the existence of a path of length d between v_i and v_j. Similarly, $v_i \not\rightsquigarrow v_j$ denotes that there is no path between v_i and v_j. We define the notion of *d-neighborhood* in order to analyze local properties of the graphs:

Definition 4. For a graph $G = (V, E)$, $s \in V$, and $d \in \mathbb{N}$, the *d-neighbourhood* of s is $\{v \mid \text{dist}(s, v) \leq d\}$, i.e., the 'ball' of radius d around s. $\mathcal{B}(s, d)$ denotes the number of nodes in this d-neighborhood, and $\mathcal{B}(d) = \max_{s \in V} \mathcal{B}(s, d)$.

Finally, a *cut* $C = (S, T)$ in G is a partition of the nodes V into subsets S and T. The *size* of the cut C is the number of edges in E with one endpoint in S and the other in T.

Probability Theory. $X \sim f(\theta)$ denotes a random variable X distributed according to probability distribution $f(\theta)$, paramterized by θ.

Given random variables $X \sim \text{Bern}(p)$ and $Y \sim \text{Bern}(q)$, their disjunction $X \vee Y$ is another Bernoulli $\text{Bern}(p \oplus q)$, where $p \oplus q \triangleq 1 - (1-p)(1-q) = p + q - pq$. We will make extensive use of this notation throughout this work.

8.4. The Meaning-Symbol Interface

We introduce two notions of knowledge spaces:

- The *meaning space*, M, is a conceptual hidden space where all the facts are accurate and complete. We assume the knowledge in this space can be represented as an undirected graph, denoted $G_M(V_M, E_M)$. This knowledge is hidden, and representative of the information that exists within human minds.

- The *symbol space*, S, is the space of written sentences, curated knowledge-based, etc., in which knowledge is represented for human and machine consumption. We assume access to a knowledge graph $G_S(V_S, E_S)$ in this space that is an incomplete, noisy, redundant, and ambiguous approximation of G_M.

There are interactions between the two spaces: when we read a sentence, we are reading from the *symbol space* and interpreting it in the *meaning space*. When writing out our thoughts, we symbolize our thought process, by moving them from *meaning space* to the *symbol space*. Figure 26 provides a high-level view of the framework. A reasoning system is not aware of the exact structure and information encoded in the meaning graph.

The only information given is the ball-assumption, i.e., we assume that each node m is connected to at most $\mathcal{B}(m, d)$ many nodes, within distance at most d. If this bound holds

Algorithm 1: Generative construction of knowledge graphs; sampling a symbol knowledge graph G_S given a meaning graph G_M.

Input: Meaning graph $G_M(V_M, E_M)$, discrete distribution $r(\lambda)$, edge retention probability p_+, edge creation probability p_-
Output: Symbol graph $G_S(V_S, E_S)$
foreach $v \in V_M$ **do**
  sample $k \sim r(\lambda)$
  construct a collection of new nodes U s.t. $|U| = k$
  $V_S \leftarrow V_S \cup U$
  $\mathcal{O}(v) \leftarrow U$
foreach $(m_1, m_2) \in (V_M \times V_M), m_1 \neq m_2$ **do**
  $S_1 \leftarrow \mathcal{O}(m_1)$, $S_2 \leftarrow \mathcal{O}(m_2)$
  foreach $e \in S_1 \times S_2$ **do**
    if $(m_1, m_2) \in E_M$ **then**
      with probability p_+: $E_S \leftarrow E_S \cup \{e\}$
    else
      with probability p_-: $E_S \leftarrow E_S \cup \{e\}$

for all the nodes in a graph, we'd simply write it as $\mathcal{B}(d)$. The ball assumption is a simple understanding of the maximum-connectivity in the meaning-graph, without knowing the details of the connections.

Meaning-Symbol mapping. We define an oracle function $\mathcal{O} : M \to 2^S$ that map nodes in the meaning space to those in the symbol space. When $s \in \mathcal{O}(m)$, with some abuse of notation, we write $\mathcal{O}^{-1}(s) = m$.

Generative Modeling of Symbol Graphs. We now explain a generative process for constructing symbol graphs. Starting with G_M, we sample a symbol graph $G_S \leftarrow \mathsf{ALG}(G_M)$ using a stochastic process, detailed in Algorithm 1. Informally, the algorithm simulates the process of transforming conceptual information into linguistic utterances (web-pages, conversations, knowledge-bases).

Our stochastic process has three main parameters: (a) the distribution $r(\lambda)$ of the number of replicated symbols to be created for each node in the meaning space; (b) the edge retention probability p_+; and (c) the noisy edge creation probability p_-. We will discuss later the regimes under which Algorithm 1 generates interesting symbol graphs.

This construction models a few key properties of linguistic representation of meaning. Each node in the meaning space is potentially mapped to multiple nodes in the symbol space, which models redundancy. Incompleteness of knowledge is modeled by the fact that not all meaning space edges appear in the symbol space (controlled by parameter p_+ in Algorithm 1). There are also edges in the symbol space that do not correspond to any edges in the meaning space and account for the noise (controlled by parameter p_- in Algorithm 1).

Next, we introduce a linguistic similarity based connection to model ambiguity, i.e., a single node in the symbol graph mapping to multiple nodes in the meaning graph. The ambiguity phenomena is modelled indirectly via the linguistic similarity based connections (discussed next). We view ambiguity as treating (or confusing) two symbol nodes as the same even when they originate from different nodes in the meaning space.

Noisy Similarity Metric. Similarity metrics are typically used to judge the equivalence of symbolic assertions. Let $\rho : V_S \times V_S \to \{0, 1\}$ be such a metric, where $\rho(s, s') = 1$ denotes the equivalence of two nodes in the symbol graph. Specifically, we define the similarity to be a noisy version of the true node similarity between node pairs:

$$\rho(s, s') \triangleq \begin{cases} 1 - \text{Bern}(\varepsilon_+) & \text{if } \mathcal{O}^{-1}(s) = \mathcal{O}^{-1}(s') \\ \text{Bern}(\varepsilon_-) & \text{otherwise} \end{cases},$$

where $\varepsilon_+, \varepsilon_- \in (0, 1)$ are the noise parameters of the similarity function, both typically close to zero. Intuitively, the similarity function is a *perturbed* version of ground-truth similarities, with small random noise (parameterized with ε_+ and ε_-). Specifically with a high probability $1-\varepsilon_{+/-}$, it returns the correct similarity decision (i.e., whether two symbols have the same meaning); and with a low probability $\varepsilon_{+/-}$ it returns an incorrect similarity decision. In particular, $\varepsilon_+ = \varepsilon_- = 0$ models the perfect similarity metric. In practice, even the best entailment/similarity systems have some noise (modeled as $\varepsilon_{+/-} > 0$).

We assume algorithms have access to the symbol graph G_S and the similarity function ρ,

and that they use the following procedure to verify the existence of a connection between two nodes:

function NODEPAIRCONNECTIVITY(s, s')
    return $(s, s') \in E_S$ or $\rho(s, s') = 1$
end function

There are many corner cases that result in uninteresting meaning or symbol graphs. Below we define the regime of realistic instances:

Definition 5 (Nontrivial Graph Instances)**.** A pair (G_M, G_S) of a meaning graph and a symbol graph sampled from it is *non-trivial* if it satisfies:

1. non-zero noise, i.e., $p_-, \varepsilon_-, \varepsilon_+ > 0$;
2. incomplete information, i.e., $p_+ < 1$;
3. noise content does not dominate the actual information, i.e., $p_- \ll p_+, \varepsilon_+ < 0.5$ and $p_+ > 0.5$;
4. G_M is not overly-connected, i.e., $\mathcal{B}(d) \in o(n)$, where n is the number of nodes in G_M;
5. G_M is not overly-sparse, i.e., $|E_{G_M}| \in \omega(1)$.

Henceforth, we will only consider sampling parameters satisfying the above conditions.

Reasoning About Meaning, through Symbols. While the reasoning engine only sees the symbol graph G_S, it must make inferences about the potential latent meaning graph. Given a pair of nodes $\mathcal{V}_S := \{s, s'\} \subset V_S$ in the symbol graph, the reasoning algorithm must then predict properties about the corresponding nodes $\mathcal{V}_M = \{m, m'\} = \{\mathcal{O}^{-1}(s), \mathcal{O}^{-1}(s')\}$ in the meaning graph.

We use a hypothesis testing setup to assess the likelihood of two disjoint hypotheses defined over these meaning nodes: $H_M^{①}(\mathcal{V}_M)$ and $H_M^{②}(\mathcal{V}_M)$. Given observations about the symbol nodes, defined as $X_S(\mathcal{V}_S)$, the goal of a reasoning algorithm is to identify which of

the two hypotheses about the meaning graph has a higher likelihood of resulting in these observations under the sampling process of Algorithm 1. Formally, we are interested in:

$$\underset{h \in \{H_M^{①}(\mathcal{V}_M), H_M^{②}(\mathcal{V}_M)\}}{\operatorname{argmax}} \mathbb{P}^{(h)}\left[X_S(\mathcal{V}_S)\right] \tag{8.1}$$

where $\mathbb{P}^{(h)}[x]$ denotes the probability of an event x in the sample space induced by Algorithm 1 on the latent meaning graph G_M when it satisfies hypothesis h.

Since we start with two disjoint hypotheses on G_M, the resulting probability spaces are generally different, making it plausible to identify the correct hypothesis with high confidence. At the same time, with sufficient noise in the sampling process, it can also become difficult for an algorithm to distinguish the two resulting probability spaces (corresponding to the two hypotheses) especially depending on the observations $X_S(\mathcal{V}_S)$ used by the algorithm. For example, the distance between the symbolic nodes can often be an insufficient indicator for distinguishing these hypotheses. We will explore these two contrasting behaviors in the next section.

Definition 6 (Reasoning Problem)**.** The input for an instance $\mathcal{P}$ of the *reasoning problem* is a collection of parameters that characterize how a symbol graph G_S is generated from a (latent) meaning graph G_M, two hypotheses $H_M^{①}(\mathcal{V}_M), H_M^{②}(\mathcal{V}_M)$ about G_M, and available observations $X_S(\mathcal{V}_S)$ in G_S. The reasoning problem, $\mathcal{P}(p_+, p_-, \varepsilon_+, \varepsilon_-, \mathcal{B}(d), \mathrm{n}, \lambda, H_M^{①}(\mathcal{V}_M), H_M^{②}(\mathcal{V}_M), X_S(\mathcal{V}_S))$, is to map the input to the hypothesis h as per Eq. (8.1).

We use the following notion to measure the effectiveness of the observation X_S in distinguishing between the two hypotheses as in Eq. (8.1):

Definition 7 (γ-Separation)**.** For $\gamma \in [0, 1]$ and a problem instance $\mathcal{P}$ with two hypotheses $h_1 = H_M^{①}(\mathcal{V}_M)$ and $h_2 = H_M^{②}(\mathcal{V}_M)$, we say an observation $X_S(\mathcal{V}_S)$ in the symbol space γ-separates h_1 from h_2 if:

$$\mathbb{P}^{(h_1)}\left[X_S(\mathcal{V}_S)\right] - \mathbb{P}^{(h_2)}\left[X_S(\mathcal{V}_S)\right] \geq \gamma.$$

We can view γ as the *gap* between the likelihoods of the observation $X_S(\mathcal{V}_S)$ having originated from a meaning graph satisfying hypothesis h_1 vs. one satisfying hypothesis h_2. When $\gamma = 1$, $X_S(\mathcal{V}_S)$ is a perfect discriminator for distinguishing h_1 and h_2. In general, any positive γ bounded away from 1 yields a valuable observation.[3]

Given an observation X_S that γ-separates h_1 and h_2, there is a simple algorithm that distinguishes h_1 from h_2:

function SEPARATOR$_{X_S}(G_S, \mathcal{V}_S = \{s, s'\})$
  if $X_S(\mathcal{V}_S) = 1$ **then** return h_1 **else** return h_2
end function

Importantly, this algorithm does *not* compute the probabilities in Definition 7. Rather, it works with a particular instantiation G_S of the symbol graph. We refer to such an algorithm $\mathcal{A}$ as γ-**accurate** for h_1 and h_2 if, under the sampling choices of Algorithm 1, it outputs the 'correct' hypothesis with probability at least γ; that is, for both $i \in \{1, 2\}$: $\mathbb{P}^{(h_i)}[\mathcal{A} \text{ outputs } h_i] \geq \gamma$.

Proposition 1. If observation X_S γ-separates h_1 and h_2, then algorithm SEPARATOR$_{X_S}$ is γ-accurate for h_1 and h_2.

Proof. Let $\mathcal{A}$ denote SEPARATOR$_{X_S}$ for brevity. Combining γ-separation of X_S with how $\mathcal{A}$ operates, we obtain:

$$\mathbb{P}^{(h_1)}[\mathcal{A} \text{ outputs } h_1] - \mathbb{P}^{(h_2)}[\mathcal{A} \text{ outputs } h_1] \geq \gamma$$
$$\Rightarrow \mathbb{P}^{(h_1)}[\mathcal{A} \text{ outputs } h_1] + \mathbb{P}^{(h_2)}[\mathcal{A} \text{ outputs } h_2] \geq 1 + \gamma$$

Since each term on the left is bounded above by 1, each of them must also be at least γ. □

In the rest of work, we will analyze when one can obtain a γ-accurate algorithm, using γ-separation of the underlying observation as a tool for the analysis.

[3] If the above probability gap is negative, one can instead use the complement of $X_S(\mathcal{V}_S)$ for γ-separation.

We will assume that the replication factor (i.e., the number of symbol nodes corresponding to each meaning node) is a constant, i.e., r is such that $\mathbb{P}[|U| = \lambda] = 1$.

8.5. Connectivity Reasoning Algorithm

One simple but often effective approach for reasoning is to focus on connectivity (as described in Figure 27). Specifically, we consider reasoning chains as valid if they correspond to a short path in the meaning space, and invalid if they correspond to disconnected nodes. Given nodes $m, m' \in G_M$, this corresponds to two possible hypotheses:

$$h_1 = m \overset{d}{\nleftrightsquigarrow} m', \text{ and } h_2 = m \not\nleftrightsquigarrow m'$$

We refer to distinguishing between these two worlds as the **d-connectivity reasoning problem**. While we consider two extreme hypotheses for our analysis, we find that with a small amount of noise, even these extreme hypotheses can be difficult to distinguish.

For the reasoning algorithm, one natural observation that can be used is the connectivity of the symbol nodes in G_S. Existing models of multi-hop reasoning (Khot et al., 2017) use similar features to identify valid reasoning chains. Specifically, we consider the observation that there is a path of length at most $\tilde{d}$ between s and s':

$$X_S^{\tilde{d}}(s, s') = \; s \overset{\leq \tilde{d}}{\nleftrightsquigarrow} s'$$

The corresponding **connectivity algorithm** is $\textsc{Separator}_{X_S^{\tilde{d}}}$, which we would like to be γ-accurate for the two hypotheses under consideration. Next, we derive bounds on γ for these specific hypotheses and observation. Note that while the space of possible hypotheses and observations is large, the above natural and simple choices still allow us to derive valuable intuitions for the limits of reasoning.

8.5.1. Possibility of accurate connectivity

We begin by defining the following accuracy threshold, γ^*, as a function of the parameters for sampling a symbol graph:

Definition 8. Given $n, d \in \mathbb{N}$ and symbol graph sampling parameters $p_+, \varepsilon_+, \lambda$, define $\gamma^*(n, d, p_+, \varepsilon_+, \varepsilon_-, \lambda)$ as

$$\left(1 - (1 - (p_+ \oplus \varepsilon_-))^{\lambda^2}\right)^d \cdot \left(1 - 2e^3 \varepsilon_+^{\lambda/2}\right)^{d+1} - 2en(\lambda \mathcal{B}(d))^2 p_-.$$

This expression is somewhat difficult to follow. Nevertheless, as one might expect, the accuracy threshold γ^* increases (higher accuracy) as p_+ increases (higher edge retention) or ε_+ decreases (fewer dropped connections between replicas). As λ increases (higher replication), the impact of the noise on edges between node cluster or d decreases (shorter paths), the accuracy threshold will also increase.

The following theorem (see Appendix for a proof) establishes the possibility of a γ-accurate algorithm for the connectivity problem:

Theorem 1. Let $p_+, p_-, \varepsilon_+, \varepsilon_-, \lambda$ be parameters of the sampling process in Algorithm 1 on a meaning graph with n nodes. Let $d \in \mathbb{N}$ and $\tilde{d} = d(1 + \lambda)$. If p_- and d satisfy

$$(p_- \oplus \varepsilon_-) \cdot \mathcal{B}^2(d) < \frac{1}{2e\lambda^2 n},$$

and $\gamma = \max\{0, \gamma^*(n, d, p_+, \varepsilon_+, \varepsilon_-, \lambda)\}$, then the connectivity algorithm $\textsc{Separator}_{X_S^{\tilde{d}}}$ is γ-accurate for the d-connectivity problem.

Proof idea. The proof consists of two steps: first show that for the assumed choice of parameters, connectivity in the meaning space is recoverable in the symbol space, with high-probability. Then show that spurious connectivity in the symbol space (with no meaning space counterparts) has low probability. □

Corollary 1. (Informal) If p_-, ε_-, d, and γ are small enough, then the connectivity algo-

rithm $\textsc{Separator}_{X_S^{\tilde{d}}}$ with $\tilde{d} = d(1+\lambda)$ is γ-accurate for the d-connectivity problem.

8.5.2. Limits of connectivity algorithm

We show that as d, the distance between two nodes in the meaning space, increases, it is unlikely that we will be able to make any inference about their connectivity by assessing connectivity of the corresponding symbol-graph nodes. More specifically, if d is at least logarithmic in the number of nodes in the graph, then, even for relatively small amounts of noise, the algorithm will see all node-pairs as connected within distance d; hence any informative inference will be unlikely.

Theorem 2. Let $c > 1$ be a constant and $p_-, \varepsilon_-, \lambda$ be parameters of the sampling process in Algorithm 1 on a meaning graph G_M with n nodes. Let $d \in \mathbb{N}$ and $\tilde{d} = \lambda d$. If

$$p_- \oplus \varepsilon_- \geq \frac{c}{\lambda n} \quad \text{and} \quad d \in \Omega(\log n),$$

then the connectivity algorithm $\textsc{Separator}_{X_S^{\tilde{d}}}$ almost-surely infers any node-pair in G_M as connected, and is thus not γ-accurate for any $\gamma > 0$ for the d-connectivity problem.

Proof idea. One can show that, for the given choice of parameters, noisy edges would dominate over informative ones and the symbol-graph would be a densely connected graph (i.e., one cannot distinguish actual connectivities from the spurious ones). □

This result exposes an inherent limitation to multi-hop reasoning: even for small values of noise, the diameter of the symbol graph becomes very small, namely, logarithmic in n. This has a resemblance to similar observations in various contexts, commonly known as the *small-world phenomenon*. This principle states that in many real-world graphs, nodes are all linked by short chains of acquaintances, such as "six degrees of separation" (Milgram, 1967; Watts and Strogatz, 1998). Our result affirms that if NLP reasoning algorithms are not designed carefully, such macro behaviors will necessarily become bottlenecks.

We note that the preconditions of Theorems 1 and 2 are disjoint, that is, both results do not

apply simultaneously. Since $\mathcal{B}(.) \geq 1$ and $\lambda \geq 1$, Theorem 1 requires $p_- \oplus \varepsilon_- \leq \frac{1}{2e\lambda^2 n} < \frac{1}{\lambda^2 n}$, whereas Theorem 2 applies when $p_- \oplus \varepsilon_- \geq \frac{c}{\lambda n} > \frac{1}{\lambda^2 n}$.

8.6. Limits of General Algorithms

While in the previous section we showed limitations of multi-hop reasoning in inferring long-range relations, here we extend the argument to prove the difficulty for *any* reasoning algorithm.

Our exposition is algorithm independent; in other words, we do not make any assumption on the choice of $E_S(s, s')$ in Equation 8.1. In our analysis we use the spectral properties of the graph to quantify local information within graphs.

Consider a meaning graph G_M in which two nodes m and m' are connected. We drop edges in a min-cut C to make the two nodes disconnected and get G'_M (Figure 28).

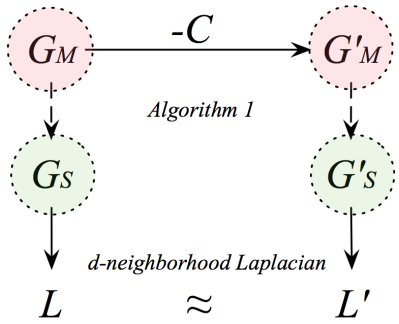


Figure 28: The construction considered in Definition 9. The node-pair m-m' is connected with distance d in G_M, and disconnected in G'_M, after dropping the edges of a cut C. For each symbol graph, we consider it "local" Laplacian.

Definition 9. Define a pair of meaning-graphs G and G', both with size n and satisfying the ball assumption $\mathcal{B}(d)$, with the following properties: (1) $m \overset{d}{\leftrightsquigarrow} m'$ in G, (2) $m \not\leftrightsquigarrow m'$ in G', (3) $E_{G'} \subset E_G$, (4) $C = E_G \setminus E_{G'}$, an (m, m') min-cut of G.

We define a uniform distribution over all the instances that satisfy the construction explained in Definition 9:

Definition 10. We define a distribution $\mathcal{G}$ over pairs of possible meaning graphs G, G' and pairs of nodes m, m' which satisfies the requirements of Definition 9. Formally, $\mathcal{G}$ is a uniform distribution over the following set:

$$\{(G, G', m, m') \mid G, G', m, m' \text{satisfy Definition 9}\}.$$

For the meaning graphs, we sample a symbol graph G_S and G'_S, as denoted in Figure 28. In the sampling of G_S and G'_S, all the edges share the randomization, except for the ones that correspond to C (i.e., the difference between the G_M and G'_M). Let $\mathcal{U}$ be the union of the nodes involved in $\tilde{d}$-neighborhood of s, s', in G_S and G'_S. Define L, L' to be the Laplacian matrices corresponding to the nodes of $\mathcal{U}$. As n grows, the two Laplacians become less distinguishable whenever $p_- \oplus \varepsilon_-$ and d are large enough:

Lemma 1. Let $c > 0$ be a constant and p_-, λ be parameters of the sampling process in Algorithm 1 on a pair of meaning graphs G and G' on n nodes constructed according to Definition 9. Let $d \in \mathbb{N}, \tilde{d} \geq \lambda d$, and L, L' be the Laplacian matrices for the $\tilde{d}$-neighborhoods of the corresponding nodes in the sampled symbol graphs G_S and $G_S{}'$. If $p_- \oplus \varepsilon_- \geq \frac{c \log n}{n}$ and $d > \log n$, then, with a high probability, the two Laplacians are close:

$$\|\tilde{L} - \tilde{L}'\| \leq \frac{\sqrt{2\lambda}\,\mathcal{B}(1)}{\sqrt{n \log(n\lambda)}}$$

This can be used to show that, for such large enough p_- and d, the two symbol graphs, G_S and G'_S sampled as above, are indistinguishable by any function operating over a λd-neighborhood of s, s' in G_S, with a high probability.

A reasoning function can be thought of a mapping defined on normalized Laplacians, since they encode all the information in a graph. For a reasoning function f with limited precision, the input space can be partitioned into regions where the function is constant; and for large enough values of n both $\tilde{L}, \tilde{L}'$ (with a high probability) fall into regions where f is constant.

Note that a reasoning algorithm is oblivious to the the details of C, i.e. it does not know where C is, or where it has to look for the changes. Therefore a realistic algorithm ought to use the neighborhood information collectively.

In the next lemma, we define a function f to characterize the reasoning function, which uses Laplacian information and maps it to binary decisions. We then prove that for any such functions, there are regimes that the function won't be able to distinguish $\tilde{L}$ and $\tilde{L}'$:

Lemma 2. Let meaning and symbol graphs be constructed under the conditions of Lemma 1. Let $\beta > 0$ and $f : \mathbb{R}^{|\mathcal{U}|\times|\mathcal{U}|} \to \{0, 1\}$ be the indicator function of an open set. Then there exists $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$:

$$\mathbb{P}_{\substack{(G,G',m,m')\sim\mathcal{G} \\ G_S\leftarrow\mathsf{ALG}(G), \\ G'_S\leftarrow\mathsf{ALG}(G')}} \left[f(\tilde{L}) = f(\tilde{L}') \right] \geq 1 - \beta.$$

This yields the following result:

Theorem 3. Let $c > 0$ be a constant and $p_-, \varepsilon_-, \lambda$ be parameters of the sampling process in Algorithm 1 on a meaning graph G_M with n nodes. Let $d \in \mathbb{N}$. If

$$p_- \oplus \varepsilon_- > \frac{c \log n}{\lambda n} \quad \text{and} \quad d > \log n,$$

then there exists $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, *any* algorithm cannot distinguish, with a high probability, between two nodes in G_M having a d-path vs. being disconnected, and is thus not γ-accurate for any $\gamma > 0$ for the d-connectivity problem.

Proof idea. The proof uses Lemma 2 to show that for the given choice of parameters, the informative paths are indistinguishable from the spurious ones, with high probability. □

This reveals a fundamental limitation: under noisy conditions, our ability to infer interesting phenomena in the meaning space is limited to a small, logarithmic neighborhood.

8.7. Empirical Analysis

Our formal analysis thus far provides worst-case bounds for two regions in the rather large spectrum of noisy sampling parameters for the symbol space, namely, when $p_- \oplus \varepsilon_-$ and d are either both small (Theorem 1), or both large (Theorem 2).

This section complements the theoretical findings in two ways: (a) by grounding the formalism empirically into a real-world knowledge graph, and (b) by quantifying the impact of

noisy sampling parameters on the success of the connectivity algorithm. We use $\varepsilon_- = 0$ for this experiments, but the effect turns out to be identical as long as $p_- \oplus \varepsilon_-$ stays unchanged (see Remark 1 in Appendix).

Specifically, we consider FB15k237 (Toutanova and Chen, 2015) containing a set of ⟨head, relation, target⟩ triples from a curated knowledge base, FreeBase (Bollacker et al., 2008). For scalability, we use a subset that relates to the movies domain,[4] resulting in 2855 distinct entity nodes and 4682 relation edges. We treat this as the meaning graph and sample a symbol graph as per Algorithm 1 to simulate the observed graph derived from text.

We sample symbol graphs for various values of p_- and plot the resulting symbol and meaning graph distances in Figure 29. For every value of p_- (y-axis), we sample points in the meaning graph separated by distance d (x-axis). For these points, we compute the average distance between the corresponding symbol nodes, and indicate that in the heat map using color shades.

We make two observations from this simulation. First, for lower values of p_-, disconnected nodes in the meaning graph (rightmost column) are clearly distinguishable from meaning nodes with short paths (small d) as predicted by Theorem 1, but harder to distinguish from nodes at large distances (large d). Second, and in contrast, for higher values of p_-, almost every pair of symbol nodes is connected with a very short path (dark color), making it impossible for a distance-based reasoning algorithm to confidently assess d-connectivity in the meaning graph. This simulation also confirms our finding in Theorem 2: any graph with $p_- \geq 1/\lambda n$, which is ~ 0.0001 in this case, cannot distinguish disconnected meaning nodes from nodes with paths of short (logarithmic) length (top rows).

8.8. Summary, Discussion and Practical Lessons

Our work is inspired by empirical observations of "semantic drift" of reasoning algorithms, as the number of hops is increased. There are series of works sharing this empirical ob-

[4] Specifically, relations beginning with `/film/`.

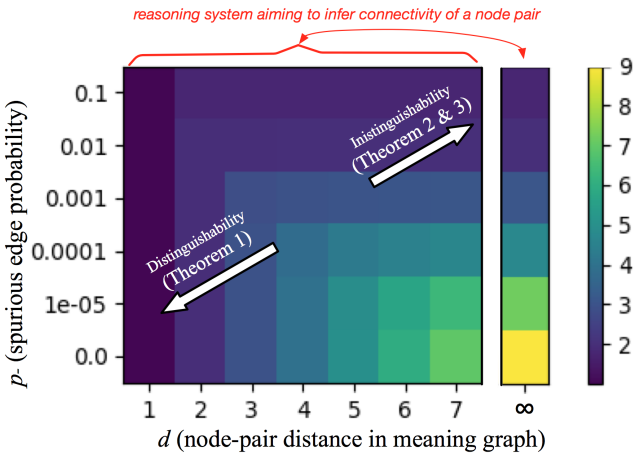


Figure 29: Various colors in the figure depict the average distance between node-pairs in the symbol graph, for each true meaning-graph distance d (x-axis), as the noise parameter p_- (y-axis) is varied. The goal is to distinguish squares in the column for a particular d with the corresponding squares in the right-most column, which corresponds to node-pairs being disconnected. This is easy in the bottom-left regime and becomes progressively harder as we move upward (more noise) or rightward (higher meaning-graph distance). ($\varepsilon_+ = 0.7, \lambda = 3$)

servation; for example, Fried et al. (2015) show modest benefits up to 2-3 hops, and then decreasing performance; Jansen (2016); Jansen et al. (2018) made similar observations in graphs built out of larger structures such as sentences, where the performance drops off around 2 hops. This pattern has interestingly been observed in a number of results with a variety of representations, including word-level representations, graphs, and traversal methods. The question we are after in this work is whether the field might be hitting a fundamental limit on multi-hop information aggregation using existing methods and noisy knowledge sources.

Our "impossibility" results are reaffirmations of the empirical intuition in the field. This means that multi-hop inference (and any algorithm that can be cast in that form), as we've been approaching it, is exceptionally unlikely to breach the few-hop barrier predicted in our analysis.

There are at least two practical lessons:

1. There are several efforts in the field pursuing "very long" multi-hop reasoning. Our results suggest that such efforts, especially without a careful understanding of the limi-

tations, are unlikely to succeed, unless some fundamental building blocks are altered.

2. A corollary of this observation suggests that, due to the limited number of hops, practitioners must focus on richer representations that allow reasoning with only a "few" hops. This, in part, requires higher-quality abstraction and grounding mechanisms. It also points to alternatives, such as offline KB completion/expansion, which indirectly reduce the number of steps needed at inference time. It basically suggests that ambiguity and variability must be handled well to reduce the number of hops needed.

Finally, we note that our proposed framework applies to any machine comprehension task over natural text that requires multi-step decision making, such as multi-hop QA or textual entailment.

CHAPTER 9 : Summary and Future Work

There was a Door to which I found no Key
There was a Veil past which I could not see:
Some little Talk awhile of ME and THEE
There seemed--and then no more of THEE and ME.

اسرار ازل را نه تو دانی و نه من
وین حرف معما نه تو خوانی و نه من
هست از پس پرده گفتگوی من و تو
چون پرده در افتد نه تو مانی و نه من

— Omar Khayyam, Rubaiyat, 1120 CE

This thesis aims at progressing towards natural language understanding, by means of the task of question answering. This chapter, gives a summary of our contributions across this document and provides a few angles along which we would like to extend this work.

9.1. Summary of Contributions

We start the discussion in Chapter 2 by providing a thorough review of the past literature concerning NLU, highlighting the ones that are related to the works in this thesis.

Chapter 3 studies reasoning systems for question answering on elementary-school science exams, using a semi-structured knowledge base. We treat QA as a subgraph selection problem and then formulate this as an ILP optimization. Most importantly, this formulation allows multiple, semi-formally expressed facts to be combined to answer questions, a capability outside the scope of IR-based QA systems. In our experiments, this approach significantly outperforms both the previous best attempt at structured reasoning for this task, and an IR engine provided with the same knowledge. Our effort has had great impacts since publication. Our work has inspired others to to build systems based on our design and to improve the state of the art in other domains; for instance, Khot et al. (2017) uses similar ideas to reasoning with OpenIE tuples (Etzioni et al., 2008). In addition, the system has been incorporated into Allen Institute's reading-comprehension project[1] and is shown to give a significant boost to their performance (Clark et al., 2016). Even after a couple of years, the system has been shown to be among the best systems on a recently-proposed reading comprehension task (Clark et al., 2018).

[1] `https://allenai.org/aristo/`

Chapter 4 extends our abductive reasoning system (from Chapter 3) to consume raw text as input knowledge. This is the first system to successfully use a wide range of semantic abstractions to perform a high-level NLP task like Question Answering. Departing from the currently popular paradigm of generating a very large dataset and learning "everything" from it in an end-to-end fashion, we demonstrate that one can successfully leverage pre-trained NLP modules to extract a sufficiently complete linguistic abstraction of the text that allows answering interesting questions about it. This approach is particularly valuable in settings where there is a small amount of data. Instead of exploiting peculiarities of a large but homogeneous dataset, as many state-of-the-art QA systems end up doing, we focus on confidently performing certain kinds of reasoning, as captured by our semantic graphs and the ILP formulation of support graph search over them.

Chapter 5 introduces the concept of essential question terms and demonstrates its importance for question answering. We introduce a dataset for this task and show that our classifier trained on this dataset substantially outperforms several baselines in identifying and ranking question terms by the degree of essentiality. Since the publication, this work has been picked up by others to significantly improve their systems (Ni et al., 2018).

Chapter 6 presents a reading comprehension dataset in which questions require *reasoning over multiple sentences*. This dataset contains $\sim 6k$ questions from different domains and wide variety of complexities. We have shown a significant performance different between human and state-of-the-art systems and we hope that this performance gap will encourage the community to work towards more sophisticated reasoning systems. It is encouraging to see that the work is already been used in a couple of works (Sun et al., 2019; Trivedi et al., 2019; Wang et al., 2019).

Chapter 7 offers a question answering dataset dedicated to *temporal common sense understanding*. We show that systems equipped with the state-of-the-art techniques are still far behind human performance. We hope that the dataset will bring more attention to the study of common sense (especially in the context of understanding of *time*).

In Chapter 8, we develop a theoretical formalism to investigate fundamental limitations pertaining to multi-step reasoning in the context of natural language problems. We present the first analysis of reasoning in the context of properties like ambiguity, variability, incompleteness, and inaccuracy. We show that a multi-hop inference (and any algorithm that can be cast in that form), as we've been approaching it, is exceptionally unlikely to breach the few-hop barrier predicted in our analysis. Our results suggest that such efforts, especially without a careful understanding of the limitations, are unlikely to succeed, unless some fundamental building blocks are altered. A corollary of this observation suggests that, practitioners must focus on richer representations that allow reasoning with only a "few" hops. This, in part, requires higher-quality abstraction and grounding mechanisms. In other words, ambiguity and variability must be handled well to reduce the number of hops needed.

9.2. Discussion and Future Directions

This thesis has taken a noticeably distinct approach towards a few important problems in the field and has shown progress on multiple ends. For example, the formalism of Chapter 3 and 4 are novel and provide general ways to formalize and implement reasoning algorithms. The datasets of Chapter 6 and 7 are distinct from the many QA datasets in the field. The theoretical analysis of Chapter 8 takes a uniquely distinct formal analysis of reasoning in the context of natural language.

All these said, there are many issues that are not addressed as extensively as we could have (or should have), or there are aspects that turned out slightly differently from what we initially expected.

Looking back at the reasoning formalism of Chapter 4, we underestimated the hardness of extracting the underlying semantic representations. Even though the field has made significant progress in low-level NLP tasks (like SRL or Coreference), such tasks still suffer from brittleness and lack of transfer across domains. And brittleness in the extraction

of such annotations, result in exponentially bigger errors when reasoning with them (as also justified by the theoretical observations of Chapter 8); in practice, it worked well only for short-ranged chains (1, 2, and sometimes 3 hops). With more recent progress in unsupervised representations and improvement of semantic extraction systems, my hope is to redo these ideas in the coming years and revisit the remaining challenges.

A vision that I would like to pursue (influenced by discussions with my advisor) is reasoning with *minimal* data. We (humans) are able to perform the same reasoning on many high-level concepts and are able to transfer them in all sorts of domains: for instance, an average human uses the same inductive reasoning to conclude *the sky is blue* and inferring that *there is another number after every number*. Effective (unsupervised) representation could potentially need a huge amount of data (and many parameters), but successful reasoning systems will likely need very minimal data (and very simple, but general definitions).

Over the past years, the field has witnessed a wave of activity on unsupervised language models (Peters et al., 2018; Devlin et al., 2018). There are many questions with respect to the success of such models on several datasets: for instance, what kinds of reasoning are they capable of? what is it that they are missing? And how we can address them by possibly creating hybrid systems. What is clear is that these systems will offer increasingly richer representations of meaning; we need better ways to effectively understand what these systems are capable of and what are the scenarios they are used to represent. And in conjunction to understanding their capabilities and limitations, we have to build reasoning algorithms on top of them. It's unlikely that these tools will ever be enough to solve all of our challenges; one has to equip these representations with the ability to reason, especially when they face an unusual/unseen scenario.

In Chapter 5 (essential terms) an initial motivation was to model *knowing what we don't know* (Rajpurkar et al., 2018); basically, systems should be able to infer whether they have enough confidence about the answer to a given query before acting. In hindsight, I think our supervised system ended up using too many shallow features, which didn't end

up generalizing to tricky instances. Additionally, it would have been better if the decision of essentiality was more involved within reasoning systems (rather than an independently supervised classifier, which limited its domain transfer).

The datasets of Chapter 6 and 7 are critical parts of this thesis which, I suspect, are likely to be remembered longer than the rest of the chapters. In general, the construction of datasets (including the ones we described) is a menial task. It's unfortunate that many small empirical details are usually left out. It is not clear to me whether using static datasets is the best way for the road ahead. In the future, I hope that the field discovers more effective ways of measuring the progress towards NLU.

A key issue contributing to the complexity of NLU (and Question Answering) is the set of implied information (common sense). We touch upon a class of such understanding in Chapter 7, where we introduce a dataset for such problems. A natural next step is addressing such questions and exploring the many ways we can incorporate such understanding in the models.

The analysis of Chapter 8 is uniquely distinct within the field. That said, there are many issues that make me feel unsatisfied about our current attempt. In particular, there are many assumptions that may or may not stand the test of time (e.g., the generative construction of symbol graph from the meaning graph or the connectivity reasoning as a proxy for the actual reasoning in language). And there are some important reasoning phenomena missing from this formalism: conditional reasoning, transitivity and directionality, inductive reasoning, just to name a few. In general, our (the field's) understanding of "reasoning" (and its formalisms) is very limited. And the existing formalisms are not easily applicable, since those who formalized reasoning were not intimately aware of the complexity of NLU; they were philosophers and mathematicians. In practice, it's really hard to make the existing theories of reasoning work in the existence of many of the properties of language. In the coming years, I would like to see more efforts on reconciling the issues in the interface of "language" and "reasoning".

APPENDIX

A.1. Supplementary Details for Chapter 3

A.1.1. The ILP Model for TableILP

Variables: We start with a brief overview of the basic variables and how they are combined into high level variables.

Table 30 summarizes our notation to refer to various elements of the problem, such as t_{ijk} for cell (j, k) of table i, as defined in Section 3. We define variables over each element by overloading $x(.)$ or $y(.,.)$ notation which refer to a binary variable on elements or their pair, respectively. Table 4 contains the complete list of basic variables in the model, all of which are binary. The pairwise variables are defined between pairs of elements; e.g., $y(t_{ijk}, q_\ell)$ takes value 1 if and only if the corresponding edge is present in the support graph. Similarly, if a node corresponding to an element of the problem is present in the support graph, we will refer to that element as being *active*.

Reference	Description
i	index over tables
j	index over table rows
k	index over table columns
l	index over lexical constituents of question
m	index over answer options
$x(.)$	a unary variable
$y(.,.)$	a pairwise variable

Figure 30: Notation for the ILP formulation.

In practice we do not create pairwise variables for all possible pairs of elements; instead we create pairwise variables for edges that have an entailment score exceeding a threshold. For example we create the pairwise variables $y\left(t_{ijk}, t_{i'j'k'}\right)$ only if $w(t_{ijk}, t_{i'j'k'}) \geq$ MinCellCellAlignment. An exhaustive list of the minimum alignment thresholds for creating pairwise variables is in Table 28.

Table 4 also includes some high level unary variables, which help conveniently impose struc-

Pairwise Variables	$y\,(t_{ijk}, t_{i'j'k'})$	1	$y\,(t_{ijk}, t_{ij'k'})$	$w(t_{ijk}, t_{ij'k'}) - 0.1$	$y\,(t_{ijk}, q_\ell)$	$w(q_\ell, t_{ijk})$	$y\,(h_{ik}, q_\ell)$	$w(q_\ell, h_{ik})$
	$y\,(t_{ijk}, a_m)$	$w(t_{ijk}, a_m)$	$y\,(h_{ik}, a_m)$	$w(h_{ik}, a_m)$				
Unary Variables	$x\,(T_i)$	1.0	$x\,(r_{ij})$	-1.0	$x\,(\ell_{ik})$	1.0	$x\,(h_{ik})$	0.3
	$x\,(t_{ijk})$	0.0	$x\,(q_\ell)$	0.3				

Table 27: The weights of the variables in our objective function. In each column, the weight of the variable is mentioned on its right side. The variables that are not mentioned here are set to have zero weight.

tural constraints on the support graph G we seek. An example is the *active row* variable $x\,(T_i)$ which should take value 1 if and only if at least a cell in row j of table i.

Objective function: Any of the binary variables defined in our problem are included in the final weighted linear objective function. The weights of the variables in the objective function (i.e. the vector $\mathbf{w}$ in Equation 2.1) are set according to Table 27. In addition to the current set of variables, we introduce auxiliary variables for certain constraints. Defining auxiliary variables is a common trick for linearizing more intricate constraints at the cost of having more variables.

Constraints: Constraints are significant part of our model in imposing the desirable behaviors for the *support graph* (cf. Section 3.1).

The complete list of the constraints is explained in Table 31. While groups of constraints are defined for different purposes, it is hard to partition them into disjoint sets of constraints. Here we give examples of some important constraint groups.

Active variable constraints: An important group of constraints relate variables to each other. The unary variables are defined through constraints that relate them to the basic pairwise variables. For example, active row variable $x\,(T_i)$ should be active if and only if any cell in row j is active. (constraint A.12, Table 31).

Correctness Constraints: A simple, but important set of constraints force the basic correctness principles on the final answer. For example G should contain exactly one answer option which is expressed by constraint A.24, Table 31. Another example is that, G should contain at least a certain number of constituents in the question, which is modeled by

constraint A.27, Table 31.

Sparsity Constraints: Another group of constraint induce simplicity (sparsity) in the output. For example G should use at most a certain number of knowledge base tables (constraint A.25, Table 31), since letting the inference use any table could lead to unreasonably long, and likely error-prone, answer chains.

A.1.2. Features used in Solver Combination

To combine the predictions from all the solvers, we learn a Logistic Regression model (Clark et al., 2016) that returns a probability for an answer option, a_i, being correct based on the following features.

Solver-independent features: Given the solver scores s_j for all the answer options j, we generate the following set of features for the answer option a_i, for each of the solvers:

1. Score = s_i
2. Normalized score = $\frac{s_i}{\sum_j s_j}$
3. Softmax score = $\frac{\exp(s_i)}{\sum_j \exp(s_j)}$
4. Best Option, set to 1 if this is the top-scoring option = $\mathbb{I}(s_i = \max s_j)$

TableILP-specific features: Given the proof graph returned for an option, we generate the following 11 features apart from the solver-independent features:

1. Average alignment score for question constituents
2. Minimum alignment score for question constituents
3. Number of active question constituents
4. Fraction of active question constituents

MinCellCellAlignment	0.6	MinCellQConsAlignment	0.1	MinTitleQConsAlignment	0.1
MinTitleTitleAlignment	0.0	MinCellQChoiceAlignment	0.2	MinTitleQChoiceAlignment	0.2
MinCellQChoiceConsAlignment	0.4	MinCellQChoiceConsAlignment	0.4	MinTitleQChoiceConsAlignment	0.4
MinActiveCellAggrAlignment	0.1	MinActiveTitleAggrAlignment	0.1		

Table 28: Minimum thresholds used in creating pairwise variables.

MaxTablesToChain	4	qConsCoalignMaxDist	4	WhichTermSpan	2
WhichTermMulBoost	1	MinAlignmentWhichTerm	0.6	TableUsagePenalty	3
RowUsagePenalty	1	InterTableAlignmentPenalty	0.1	MaxAlignmentsPerQCons	2
MaxAlignmentsPerCell	2	RelationMatchCoeff	0.2	RelationMatchCoeff	0.2
EmptyRelationMatchCoeff	0.0	NoRelationMatchCoeff	-5	MaxRowsPerTable	4
MinActiveQCons	1	MaxActiveColumnChoiceAlignments	1	MaxActiveChoiceColumnVars	2
MinActiveCellsPerRow	2				

Table 29: Some of the important constants and their values in our model.

5. Average alignment scores for question choice

6. Sum of alignment scores for question choice

7. Number of active table cells

8. Average alignment scores across all the edges

9. Minimum alignment scores across all the edges

10. Log of number of variables in the ILP

11. Log of number of constraints in the ILP

Collection of basic variables connected to header column k of table i:	$\mathcal{H}_{ik} = \{(h_{ik}, q_\ell); \forall l\} \cup \{(h_{ik}, a_m); \forall m\}$	(A.1)
Collection of basic variables connected to cell j, k of table i:	$\mathcal{E}_{ijk} = \{(t_{ijk}, t_{ij'k'}); \forall i', j', k'\} \cup \{(t_{ijk}, a_m); \forall m\} \cup \{(t_{ijk}, q_\ell); \forall l\}$	(A.2)
Collection of basic variables connected to column k of table i	$\mathcal{C}_{ik} = \mathcal{H}_{ik} \cup \left(\bigcup_j \mathcal{E}_{ijk} \right)$	(A.3)
Collection of basic variables connected to row j of table i:	$\mathcal{R}_{ij} = \bigcup_k \mathcal{E}_{ijk}$	(A.4)
Collection of non-choice basic variables connected to row j of table i:	$\mathcal{L}_{ij} = \{(t_{ijk}, t_{ij'k'}); \forall k, i', j', k'\} \cup \{(t_{ijk}, q_\ell); \forall k, l\}$	(A.5)
Collection of non-question basic variables connected to row j of table i:	$\mathcal{K}_{ij} = \{(t_{ijk}, t_{ij'k'}); \forall k, i', j', k'\} \cup \{(t_{ijk}, a_m); \forall k, m\}$	(A.6)
Collection of basic variables connected to table i:	$\mathcal{T}_i = \bigcup_k \mathcal{C}_{ik}$	(A.7)
Collection of non-choice basic variables connected to table i:	$\mathcal{N}_i = \{(h_{ik}, q_\ell); \forall l\} \cup \{(t_{ijk}, t_{ij'k'}); \forall j, k, i', j', k'\} \cup \{(t_{ijk}, q_\ell); \forall j, k, l\}$	(A.8)
Collection of basic variables connected to question constituent q_ℓ:	$\mathcal{Q}_l = \{(t_{ijk}, q_\ell); \forall i, j, k\} \cup \{(h_{ik}, q_\ell); \forall i, k\}$	(A.9)
Collection of basic variables connected to option m	$\mathcal{O}_m = \{(t_{ijk}, a_m); \forall i, j, k\} \cup \{(h_{ik}, a_m); \forall i, k\}$	(A.10)
Collection of basic variables in column k of table i connected to option m:	$\mathcal{M}_{i,k,m} = \{(t_{ijk}, a_m); \forall j\} \cup \{(h_{ik}, a_m)\}$	(A.11)

Table 30: All the sets useful in definitions of the constraints in Table 31.

If any cell in row j of table i is active, the row should be active.	$x(r_{ij}) \geq y(t_{ijk}, e), \forall (t_{ijk}, e) \in \mathcal{R}_{ij}, \forall i, j, k$	(A.12)
If the row j of table i is active, at least one cell in that row must be active as well.	$\sum_{(t_{ijk}, e) \in \mathcal{R}_{ij}} y(t_{ijk}, e) \geq x(r_{ij}), \forall i, j$	(A.13)
Column j header should be active if any of the basic variables with one end in this column header are active.	$x(h_{ik}) \geq y(h_{ik}, e), \forall (h_{ik}, e) \in \mathcal{H}_{ik}, \forall i, k$	(A.14)
If the header of column j variable is active, at least one basic variable with one end in the end in the header	$\sum_{(h_{ik}, e) \in \mathcal{H}_{ik}} y(h_{ik}, e) \geq x(h_{ik}), \forall i$	(A.15)
Column k is active if at least one of the basic variables with one end in this column are active.	$x(\ell_{ik}) \geq y(t_{ijk}, e), \forall (t_{ijk}, e) \in \mathcal{C}_{ik}, \forall i, k$	(A.16)

A.2. Supplementary Details for Chapter 4

Here we provide some details of our reasoning formulation and its implementation as an ILP.

The support graph search for QA is modeled as an ILP optimization problem, i.e., as maximizing a linear objective function over a finite set of variables, subject to a set of linear inequality constraints. We note that the ILP objective and constraints aren't tied to the particular domain of evaluation; they represent general properties that capture what

If the column k is active, at least one of the basic variables with one end in this column should be active.

$$\sum_{(t_{ijk},e)\in\mathcal{C}_{ik}} y\left(t_{ijk},e\right) \geq x\left(h_{ik}\right), \forall i,k \tag{A.17}$$

If a basic variable with one end in table i is active, the table variable is active.

$$y\left(t_{ijk},e\right) \geq x\left(T_i\right), \forall (t_{ijk},e) \in \mathcal{T}_i, \forall i \tag{A.18}$$

If the table i is active, at least one of the basic variables with one end in the table should be active.

$$\sum_{(t,e)\in\mathcal{T}_i} y\left(t,e\right) \geq x\left(T_i\right), \forall i \tag{A.19}$$

If any of the basic variables with one end in option a_m are on, the option should be active as well.

$$x\left(a_m\right) \geq y\left(x,a_m\right), \forall (e,a_m) \in \mathcal{O}_m \tag{A.20}$$

If the question option a_m is active, there is at least one active basic element connected to it

$$\sum_{(e,a)\in\mathcal{O}_m} y\left(x,a\right) \geq x\left(a_m\right) \tag{A.21}$$

If any of the basic variables with one end in the constituent q_ℓ, the constituent must be active.

$$x\left(q_\ell\right) \geq y\left(e,q_\ell\right), \forall (e,q_\ell) \in \mathcal{Q}_l \tag{A.22}$$

If the constituent q_ℓ is active, at least one basic variable connected to it must be active.

$$\sum_{(e,q_\ell)\in\mathcal{Q}_l} y\left(e,q_\ell\right) \geq x\left(q_\ell\right) \tag{A.23}$$

Choose only a single option.

$$\sum_m x\left(a_m\right) \leq 1, \quad \sum_m x\left(a_m\right) \geq 1 \tag{A.24}$$

There is an upper-bound on the number of active tables; this is to limit the solver and reduce the chance of using spurious tables.

$$\sum_i x\left(T_i\right) \leq \textsc{MaxTablesToChain} \tag{A.25}$$

The number of active rows in each table is upper-bounded.

$$\sum_j x\left(r_{ij}\right) \leq \textsc{MaxRowsPerTable}, \forall i \tag{A.26}$$

The number of active constituents in each question is lower-bounded. Clearly We need to use the question definition in order to answer a question.

$$\sum_l x\left(q_\ell\right) \geq \textsc{MinActiveQCons} \tag{A.27}$$

A cell is active if and only if the sum of coefficients of all external alignment to it is at least a minimum specified value

$$\sum_{(t_{ijk},e)\in\mathcal{E}_{i,j,k}} y\left(t_{ijk},e\right) \geq x\left(t_{ijk}\right) \times \textsc{MinActiveCellAggrAlignment}, \forall i,j,k \tag{A.28}$$

A title is active if and only if the sum of coefficients of all external alignment to it is at least a minimum specified value

$$\sum_{(e)\in\mathcal{H}_{i,k}} y\left(t_{ijk},e\right) \geq x\left(t_{ijk}\right) \times \textsc{MinActiveTitleAggrAlignment}, \forall i,k \tag{A.29}$$

If a column is active, at least one of its cells must be active as well.

$$\sum_j x\left(t_{ijk}\right) \geq x\left(\ell_{ik}\right), \forall i,k \tag{A.30}$$

At most a certain number of columns can be active for a single option

$$\sum_k y\left(\ell_{ik},a_m\right) \leq \textsc{MaxActiveChoiceColumn}, \forall i,m \tag{A.31}$$

If a column is active for a choice, the table is active too.

$$x\left(\ell_{ik}\right) \leq x\left(T_i\right), \forall i,k \tag{A.32}$$

If a table is active for a choice, there must exist an active column for choice.

$$x\left(T_i\right) \leq \sum_k x\left(\ell_{ik}\right), \forall i \tag{A.33}$$

If a table is active for a choice, there must be some non-choice alignment.

$$y\left(T_i,a_m\right) \leq \sum_{(e,e')\in\mathcal{N}_i} y\left(e,e'\right), \forall i,m \tag{A.34}$$

Answer should be present in at most a certain number of tables

$$y\left(T_i,a_m\right) \leq \textsc{MaxActiveTableChoiceAlignmets}, \forall i,m \tag{A.35}$$

If a cell in a column, or its header is aligned with a question option, the column is active for question option as well.

$$y\left(t_{ijk},a_m\right) \leq y\left(\ell_{ik},a_m\right), \forall i,k,m, \forall (t_{ijk},a_m) \in \mathcal{M}_{i,k,m} \tag{A.36}$$

If a column is active for an option, there must exist an alignment to header or cell in the column.

$$y\left(\ell_{ik},a_m\right) \leq \sum_{(t_{ijk},a_m)\in\mathcal{O}_{i,k,m}} y\left(t_{ijk},a_m\right), \forall i,m \tag{A.37}$$

At most a certain number of columns may be active for question option in a table.	$\sum_k y(\ell_{ik}, a_m) \leq$ MaxActiveChoiceColumnVars, $\forall i, m$ (A.38)
If a column is active for a choice, the table is active for an option as well.	$y(\ell_{ik}, a_m) \leq y(T_i, a_m), \forall i, k, m$ (A.39)
If the table is active for an option, at least one column is active for a choice	$y(T_i, a_m) \leq \sum_k y(\ell_{ik}, a_m), \forall i, m$ (A.40)
Create an auxiliary variable x (whichTermIsActive) with objective weight 1.5 and activate it, if there a "which" term in the question.	$\sum_l 1\{q_\ell = \text{"which"}\} \leq x(\text{whichTermIsActive})$ (A.41)
Create an auxiliary variable x (whichTermIsAligned) with objective weight 1.5. Add a boost if at least one of the table cells/title aligning to the choice happens to have a good alignment ($\{w(.,.) >$ MinAlignmentWhichTerm$\}$) with the "which" terms, i.e. WhichTermSpan constituents after "which".	$\sum_i \sum_{(e_1,e_2)\in\mathcal{T}_i} y(e_1, e_2) \geq x(\text{whichTermIsAligned})$ (A.42)
A question constituent may not align to more than a certain number of cells	$\sum_{(e,q_\ell)\in\mathcal{Q}_l} y(e, q_\ell) \leq$ MaxAlignmentsPerQCons (A.43)
Disallow aligning a cell to two question constituents if they are too far apart; in other words add the following constraint if the two constituents q_ℓ and $q_{\ell'}$ are more than qConsCoalignMaxDist apart from each other:	$y(t_{ijk}, q_\ell) + y(t_{ijk}, q_{\ell'}) \leq 1, \forall l, l', i, j, k$ (A.44)
For any two two question constraints that are not more than qConsCoalignMaxDist apart create an auxiliary binary variable x (cellProximityBoost) and set its weight in the objective function to be $1/(l - l' + 1)$, where l and l' are the indices of the two question constituents. With this we boost objective score if a cell aligns to two question constituents that are within a few words of each other	$x(\text{cellProximityBoost}) \leq y(t_{ijk}, q_\ell)$, $x(\text{cellProximityBoost}) \leq y(t_{ijk}, q_{\ell'}), \forall i, j, k$ (A.45) set
If a relation match is active, both the columns for the relation must be active	$r(\ell_{ik}, \ell_{ik'}, q_\ell, q_{\ell'}) \leq x(\ell_{ik}), r(\ell_{ik}, \ell_{ik'}, q_\ell, q_{\ell'}) \leq x(\ell_{ik'})$ (A.46)
If a column is active, a relation match connecting to the column must be active	$x(\ell_{ik}) \leq \sum_{k'}(r(\ell_{ik}, \ell_{ik'}, q_\ell, q_{\ell'}) + r(\ell_{ik'}, \ell_{ik}, q_\ell, q_{\ell'})), \forall k$ (A.47)
If a relation match is active, the column cannot align to the question in an invalid position	$r(\ell_{ik}, \ell_{ik'}, q_\ell, q_{\ell'}) \leq 1 - y(t_{ijk}, \hat{q}_\ell)$, where $\hat{q}_\ell \leq q_\ell$ and $t_{ijk} \in \ell_{ik}$ (A.48)
If a row is active, at least a certain number of its cells must be active	$\sum_k x(t_{ijk}) \geq$ MinActiveCellsPerRow $\times x(r_{ij}), \forall i, j$ (A.49)
If row is active, it must have non-choice alignments.	$x(r_{ij}) \leq \sum_{(n,n')\in\mathcal{L}_{ij}} y(n, n)$ (A.50)
If row is active, it must have non-question alignments	$x(r_{ij}) \leq \sum_{(n,n')\in\mathcal{K}_{ij}} y(n, n)$ (A.51)
If two rows of a table are active, the corresponding active cell variables across the two rows must match; in other words, the two rows must have identical activity signature	$x(r_{ij}) + x(r_{ij'}) + x(t_{ijk}) - x(t_{ij'k'}) \leq 2, \forall i, j, j', k, k'$ (A.52)
If two rows are active, then at least one active column in which they differ (in tokenized form) must also be active; otherwise the two rows would be identical in the proof graph.	$\sum_{t_{ijk}\neq t_{ijk'}} x(\ell_{ik}) - x(r_{ij}) - x(r_{ij'}) \geq -1$ (A.53)
If a table is active and another table is also active, at least one inter-table active variable must be active;	$x(T_i) + x(T_{i'}) + \sum_{j,k,j',k'} y(t_{ijk}, t_{i'j'k'}) \geq 1, \forall i, i'$ (A.54)

Table 31: The set of all constraints used in our ILP formulation. The set of variables and are defined in Table 4. More intuition about constraints is included in Section 3. The sets used in the definition of the constraints are defined in Table 30.

constitutes a well-supported answer for a given question.

Our formulation consists of multiple kinds of reasoning, encapsulating our semantic understanding of the types of knowledge (e.g., verbs, corefs, etc.) extracted from the text, the family to graphs used to represent them, and how they interact in order to provide support for an answer candidate. Each kind of reasoning is added to a general body, defined in Table 32, that is shared among different reasoning types. This general body encapsulates basic requirements such as at most one answer candidate being chosen in the resulting support graph.

In what follows we delineate various forms of reasoning, capturing different semantic abstractions and valid interactions between them.

Comb-1 (Shallow Alignment)

This reasoning captures the least-constrained formulation for answering questions. We create alignments between Tokens view of the question, and spans of Shallow-Parse view of the paragraph. Alignments are scored with the entailment module; the closer the surface strings are, the higher score their edge gets. There are variables to induce sparsity in the output; for example, penalty variables for using too many sentences in the paragraph, or using too many spans in the paragraph. To Add more structure to this, we use Coreference and Dependency views of the paragraph; for example, alignments that are closer to each other according the dependency parse, get a higher score. The Coreference links let the reasoning to jump in between sentences. Figure 31 shows an example output of this type of reasoning. As it can be seen, the alignment is using multiple terms in the questions and multiple spans in the paragraph, and the spans belong to one single sentence in the paragraph.

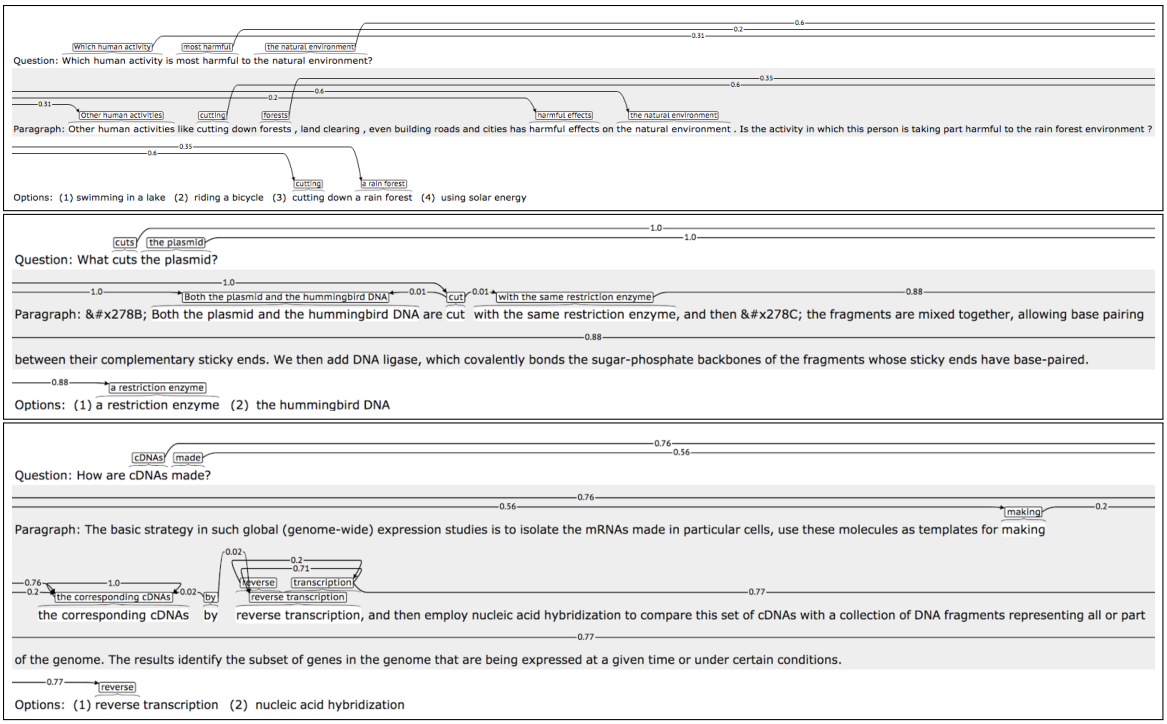


Figure 31: Examples of system output for (1) top: Comb-1 (Shallow alignment) (2) middle: Comb-2 (Verb-SRL alignment) (3) bottom: Comb-5 (Verb-SRL+ Prep-SRL alignment).

Comb-2 (Verb-SRL Alignment)

This reasoning is using Verb-SRL views in both question and paragraph. The core of this alignment is creating connections between the predicate/arguments and the predicate/arguments of the paragraph, respectively. The edges are scored with our entailment system; hence the bigger the edge weight is, the higher chance of appearing in the output. An example outputs are in Figure 31.

Comb-5 (Verb-SRL+Prep-SRL Alignment)

In this type of reasoning we observe the combination of Verb-SRL and Prep-SRL. This can be considered as an extension of Comb-2 (Verb-SRL alignment), we the alignment is in between Verb-SRL in question, Verb-SRL and Prep-SRL in the paragraph. The arguments of the Verb-SRLin the question are aligned to the arguments of either Verb-SRL and Prep-SRL in the

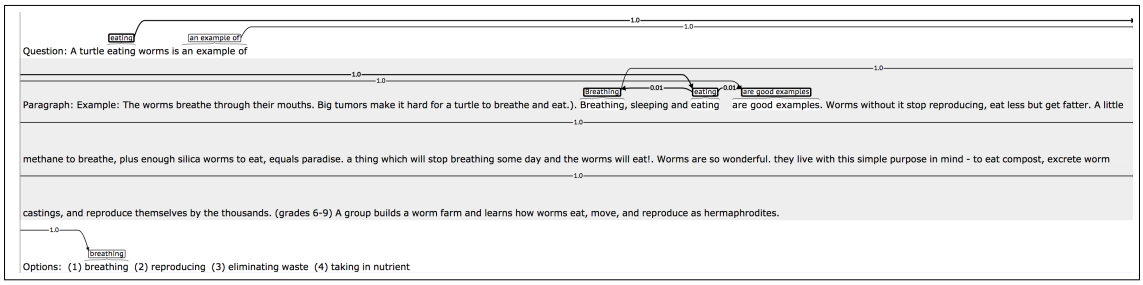


Figure 32: Example output of a question SemanticILP answered incorrectly due to a mistake in annotations. "eating" in the paragraph is incorrectly recognized as a verb predicate, with "breathing" as subject, resulting in this incorrect alignment.

paragraph. Predicates of the Verb-SRL are aligned to the predicates Verb-SRL. An example output prediction is shown in Figure 31.

Similar to the combinations explained, one can identify different combinations, such as Comb-3(Verb-SRL+Coreference Alignment), Comb-4(Verb-SRL+Comma-SRL Alignment) or Verb-SRL+Nom-SRL Alignment.

* *Variables:*
Active answer option variable
Active answer option chunk variable
* *Active variable constraints:*
If any edge connected to any active variable that belongs to answer option, the answer should be active
If an answer is active, there must be at least one active edge connected to it.
* *Consistency constraints:*
Only one answer option should be active.

Table 32: Generic template used as part of each reasoning

A.3. Supplementary Details for Chapter 5

A.3.1. Features of the ET *Classifier*

Here we explain the structure of our classifier. In our design we give references to the NLP/Machine Learning frameworks we used to build the system, as well as a short account of our features.

In this work use the Saul framework (Kordjamshidi et al., 2015; Khashabi et al., 2017) [1]

[1] Available at: `https://github.com/CogComp/saul`

* *Basic variables:*
Active question-terms
Active paragraph-spans
Active paragraph sentence
Question-term to paragraph-span alignment variable
Paragraph-span alignment to answer-option term alignment variable
* *Active variable constraints:*
Question-term should be active, if any edge connected to it is active.
If a question-term is active, at least one edge connected to it should be active.
Sentence should be active, if anything connected to it is active.
If a sentence is active, at least one incoming edge to one of its terms/spans should be active.
* *Question sparsity-inducing variables:*
More than k active question-term penalty. (for $k = 1, 2, 3$)
More than k active alignments to each question-term penalty. (for $k = 1, 2, 3$)
* *Paragraph sparsity-inducing variables:*
- Active sentence penalty variable.
* *Proximity-inducing variables:*
Active dependency-edge boost variable: if two variables are connected in dependency path, and are both active, this variable can be active.
- Word-pair distance $\leq k$ words boost: variable between any two word-pair with distance less than k and active if both ends are active. (for $k = 1, 2, 3$)
* *Sparsity-inducing variables in answer options:*
- More than k active chunks in the active answer-option. (for $k = 1, 2, 3$)
- More than k active edges connected to each chunk of the active answer option. (for $k = 1, 2, 3$)

Table 33: Comb-1 (Shallow Alignment)

to our problem. This framework gives ability for easy NLP feature definitions and integrations with machine learning models. We use IllinoisPipline[2] to annotated raw documents with different views using such as Tokenizer (Tok), Lemmatizer (Lem), Part-of-Speech tagging (POS), Named-Entity-Recognition (NER), Stanford Dependency Parsing (Dependency) Marneffe et al. (2006), and Chunker (Chk).

We include the details of features we used in our classifier. A majority of features are extracted via Edison (Sammons et al., 2016), a library of feature extractor and group of hand-crafted features. Since Edisonalready covers many standard features, our work here mostly has focused on selecting relevant ones, and creating their conjunction. These features cover a wide range of syntactic, semantic features and their combinations. In Figure 33 we show the frequency in which terms with certain POS tags labeled as essential or not. We provide some interesting statistics from distribution in Section 2.1. As it can be seen, POS labels alone are not good discriminants of the target labels, while their combinations with

[2]Available at: `https://github.com/CogComp/cogcomp-nlp`

* *Variables:*
Active Verb-SRL constituents in question (both predicates and arguments), 0.01.
Active Verb-SRL constituents in the paragraph (including predicates and arguments), with weight 0.01.
Edge between question-Verb-SRL-argument and paragraph-Verb-SRL-argument (if the entailment score > 0.6).
Edge between question-Verb-SRL-predicate and paragraph-Verb-SRL-predicate (if the entailment score > 0.6).
Edge between paragraph-Verb-SRL-argument and answer-options (if the entailment score > 0.6).
Edges between predicate and its arguments argument, inside each Verb-SRL frame in the paragraph, each edge with weight 0.01.

* *Consistency constraints:*
Constraints to take care of active variables (e.g. edge variable is active iff its two nodes are active).
At least $k = 2$ constituents in the question should be active.
At least $k = 1$ predicates in the question should be active.
At most $k = 1$ predicates in the paragraph should be active.
For each Verb-SRL frame in the paragraph, if the predicate is inactive, its arguments should be inactive as well.

Table 34: Comb-2 (Verb-SRL alignment)

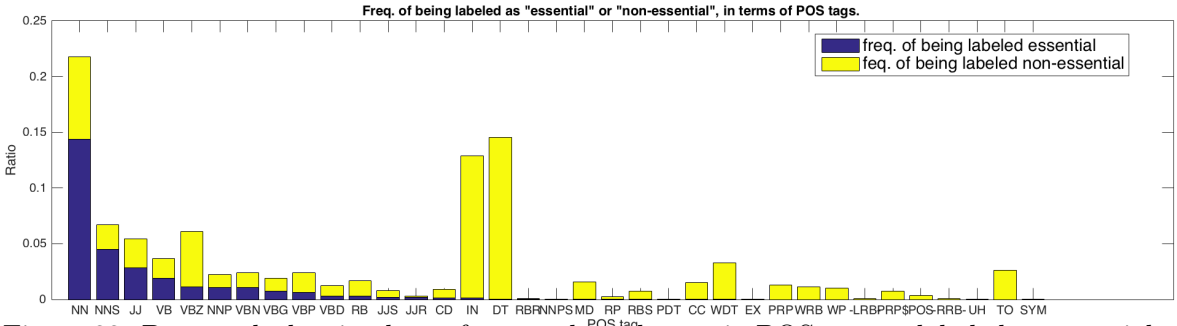


Figure 33: Bar graph showing how often words with certain POS tag are labeled as essential / non-essential.

other labels (such as lemmas) would give more informative signal.

We have included the complete list of features in Table 37. For simplicity of notation, denote the surface string feature with Surf, define Conj {. } to be a feature resulted from the conjunction of the properties in its arguments. Also let Base {. } denote a simple baseline (and feature) returns most-popular labels for its observations; for example Base {Lem } is a classifier/feature which returns the most-frequent label given the lemma of the input constituent. Denote an arbitrary feature extractor with f_i when extracting the i-th terms; similarly f_{i-1}/f_{i+1} is the same feature extractor when applied to the previous/next term. With this definition we define a neighborhood of f_i, Neighbor(f_i) be the collection

* *Variables:*
Active Verb-SRL constituents in question (both predicates and arguments), with weight 0.001.
Active Verb-SRL constituents in the paragraph (including predicates and arguments), with weight 0.001.
Active Prep-SRL constituents in the paragraph (including predicates and arguments), with weight 0.001.
Edges between any pair of Prep-SRL arguments in the paragraph and Verb-SRL arguments, with weights extracted from PARAGRAM(if these scores are > 0.7).
Edges between predicates and arguments of the Prep-SRL frames, with weight 0.02.
Edges between predicates and arguments of the Verb-SRL frames, with weight 0.01.
Edge between question-Verb-SRL-argument and paragraph coref-constituents (if cell-cell entailment score > 0.6)
Edge between question-Verb-SRL-predicate and paragraph Verb-SRL-predicate (if phrase-sim score > 0.5)
Edge between paragraph Verb-SRL-arguments and answer options (if cell-cell score is > 0.7)
* *Consistency constraints:*
Each Prep-SRL argument has to have at least an incoming edge (in addition to the edge from its predicate)
Not possible to have a Verb-SRL argument (in the paragraph)
connected to two Prep-SRL arguments of the same frame (no loop)
Exactly one Prep-SRL predicate in the paragraph
At least one Verb-SRL predicate in the paragraph
At most one Verb-SRL predicate in the paragraph
Not more than one argument of a frame can be connected to the answer option
Each Verb-SRL argument in the paragraph should have at least two active edges connected to.

Table 35: Comb-5 (Verb-SRL+Prep-SRL alignment)

of feature of the same type, applied on neighboring terms of the i-th term:

$$\begin{aligned}\textsc{Neighbor}(f_i) = \big\{ & f_i, f_{i-1}, f_{i+1}, f_{i-2}, f_{i+2}, \\ & \textsc{Conj}\big\{f_{i-1}, f_{i+1}\big\}, \\ & \textsc{Conj}\big\{f_{i-2}, f_{i-1}\big\}, \\ & \textsc{Conj}\big\{f_{i+1}, f_{i+2}\big\}\ \big\}\end{aligned}$$

All the EDISONfeatures come with prefix `ed`. The details of these features and their implementation can be directly looked up from EDISON. Using such features is simply matter of calling the feature extractor function, on top of question annotations, which makes them easy to reproduce.

Some features are crafted specifically for the purpose of this problem. We collected a set of science terms frequently used in elementary school level textbooks. The feature

* *Variables:*
Active Verb-SRL constituents in question (including predicates and arguments), with weight 0.001.
Active Verb-SRL constituents in the paragraph (including predicates and arguments), with weight 0.001.
Active Coreference constituents in the paragraph, with weight 0.001.
Active Coreference chains in the paragraph, with weight -0.0001.
Edges between any pair of Coreference-constituent in the paragraph that belong to the same Coreference chain, with weight 0.02.
Edge between question-Verb-SRL-argument and paragraph Coreference-constituents (if entailment score > 0.6)
Edge between question-Verb-SRL-predicate and paragraph Verb-SRL-predicate (if phrase-sim score > 0.4)
Edge between paragraph Verb-SRL arguments and answer options (if symmetric entailment score is > 0.65)
* *Consistency constraints:*
Constraints to take care of active variables (e.g. edge variable is active iff its two nodes are active).
At most $k = 1$ Coreference chain in the paragraph.
At least $k = 1$ constituents in the question should be active.
At most $k = 1$ Verb-SRL predicates in the paragraph. (could relax this and have alignment between multiple SRL frames in the paragraph)
The question constituents can be active, only if at least one of the edges connected to it is active.
Paragraph Verb-SRL-predicate should have at least two active edges.
If paragraph Verb-SRL-predicate is inactive, the whole frame should be inactive.

Table 36: Comb-3 (Verb-SRL+Coreference alignment)

isScienceTerm$\{.\}$ checks whether the output of its input extractor is out science terms set or not. The feature MaxPMI /SumPMI is the maximum/sum of the PMI value between the target question term, and the question options.

A.3.2. Humans as reasoning engines

Here we include the results MTurk experiments we discussed in Section 2.2 (Figure 17). Here we include the whole table for completeness. In particular for each row we have included the precision. In addition the first row of this table is an experiment without dropping any terms, just to have a sense of how precision and recall would look like if no term was dropped. As it can be seen the precisions values are mostly close to each other, which shows that humans tend to keep their retain their precision across settings, by abstaining from answering questions they are not sure.

Feature	Description
Surf	
Lem	
Conj {POS, NER }	Basic features, which are directly extracted from the annotations of the question, i.e. surface string, NER, etc, and their conjunctions.
Conj {Surf, NER }	
Conj {Surf, POS }	
Conj {Surf, NER, POS }	
Conj {Lem, POS }	
Chk	
Conj {`ed.conflatedPos`, Lem }	The features extracted by direct calls to Edisonfeature-extraction library, on top of the basic annotations of the question. For example `ed.brownClusterFeatures` extracts the Brown representation for a given word, using its internal pre-extracted Brown representation. Another example is `ed.wordnetSynsetsAllSenses`, which given a term, returns all the synsets of that word.
Conj {`ed.conflatedPos`, Surf }	
`ed.deVerbalSuffix`	
`ed.gerundMarker`	
`ed.knownPrefixes`	
`ed.nominalizationMarker`	
`ed.numberNormalizer`	
`ed.deVerbalSuffix`	
`ed.prefixSuffixes`	
`ed.parsePath`	
`ed.brownClusterFeatures`	
`ed.dependencyPathUnigram`	
`ed.dependencyPathBigram`	
`ed.isItCapitalized`	
`ed.isItLastSentence`	
`ed.wordnetSynsetsFirstSense`	
`ed.wordnetSynsetsAllSenses`	
`ed.wordnetLexicographerFileNamesAllSenses`	
`ed.wordnetLexicographerFileNamesFirstSense`	
`ed.wordnetHypernymFirstSenseLexicographerFileNames`	
`ed.wordnetHypernymAllSensesLexicographerFileNames`	
`ed.wordnetPointersFirstSense`	
`ed.wordnetPointersAllSenses`	
`ed.wordnetSynonymsFirstSense`	
`ed.wordnetSynonymsAllSense`	
Conj {`ed.wordnetExists`, `ed.wordnetSynsetsFirstSense`, `ed.wordnetLexicographerFileNamesFirstSense` }	
isScienceTerm{Surf }	We hand-craft these features based on previous features. For example isScienceTerm{Lem } checkes whether lemma of a given word is in our collected science terms or not.
isScienceTerm{Lem }	
SumPMI	
MaxPMI	
Neighbor(Base {Surf })	
Neighbor(Base {Lem })	
Neighbor(Base {Pair(Surf) })	
Neighbor(Base {Pair(Lem) })	
Neighbor(Base {Conj {POS, NER } })	
Neighbor(Base {Conj {POS, Lem } })	
Neighbor(Base {Conj {Surf, POS, Lem } })	

Table 37: List of important feature categories in our system.

	Setting	Threshold	Precision	Recall	Term-drop ratio	Test score (%)
	Nothing is dropped	–	0.868	0.970	0	0.85.02
(A) Gold Annotations	Drop terms with essentialityscore above certain threshold	0.2	0.689	0.08	0.35	28.51
		0.4	0.693	0.168	0.299	32.44
		0.6	0.804	0.3287	0.183	43.20
		0.8	0.812	0.328	0.17	43.43
		1.0	0.823	0.49	0.12	53.07
	Keep terms with essentialityscore above some threshold	0.0	0.813	0.854	0.604	73.08
		0.2	0.7857	0.805	0.676	68.125
		0.4	0.824	0.654	0.7543	62.53
		0.6	0.77	0.66	0.8	59.32
		0.8	–	–	–	–
(B) ETClassifier	Drop terms with essentialityscore above some threshold	0.0	0.7	0.017	0.873	25.76
		0.2	0.771	0.03	0.4342	26.56
		0.4	0.9066	0.0646	0.3331	29.24
		0.6	0.9024	0.255	0.176	41.63
		0.8	0.91	0.9364	0.0812	86.802
		1.0	0.9	0.98	0	88.7
	Keep terms with essentialityscore above some threshold	0.0	0.932	0.746	0.5166	75.87
		0.2	0.937	0.803	0.5166	80.16
		0.4	0.923	0.779	0.581	77.42
		0.6	0.91	0.686	0.68	70.27
		0.8	0.538	0.02	0.87	25.57
		1.0	0.57	0.04	0.88	26.28

Table 38: This table contains the exact numbers when using essentiality scores for dropping most important/least important terms in the question (Section 2.2). The setting (A) is when the gold annotations are used and setting (B) is when real-valued scores of the trained classifier are used. Precision is ratio of the questions answered correctly, if they answered at all. Recall is the ratio of the times a question is answered. Term-drop ratio is ratio of the terms dropped in the question sentence (compared to the the overall length of the question).

Question	Key
Which item causes objects to roll downhill? (A) gravity(B) friction(C) erosion(D) magnetism	A
Which sense is used to determine a sweet object's texture? (A) hearing(B) smell(C) taste(D) touch	D
An animal grows thicker hair as a season changes. This adaptation helps to (A) find food(B) keep warmer(C) grow stronger(D) escape from predators	B
What is the main energy for the water cycle? (A) electricity(B) erosion(C) gravity(D) sunlight	D
Which human activity is most damaging to the environment? (A) swimming in a lake(B) riding a bicycle(C) cutting down a rain forest(D) using solar energy	C
Which force washes away coastal soil? (A) gravity(B) friction(C) erosion(D) magnetism	C
Which unit of measurement describes an object's size? (A) meter(B) kilogram(C) liter(D) degree	A
Which form of energy is found in food? (A) chemical(B) electrical(C) sound(D) mechanical	B
Which object is nonliving? (A) bear(B) bicycle(C) bird(D) butterfly	B
Which sense is used to determine a colorful object's texture? (A) sight (B) smell(C) taste(D) touch	D
A mamal grows thicker hair as a season changes. This adaptation helps to (A) find food(B) keep warmer(C) grow stronger(D) escape from predators	B
A mammal shivers when it's cold. This adaptation is to (A) find food (B) get warmer body (C) grow stronger(D) escape from predators	B

Table 39: List of the synthesized question for Section 4.3. These question are hand-made and perturbed versions of the existing question to trick the vanilla TableILP. The design of these questions is done completely independent of the essentiality scores.

A.3.3. Synthetic questions

Here we have included the synthetic dataset with discussed in Section 4.2. As mentioned these question are hand-made and perturbed versions of the existing question to trick the vanilla TableILP. Note that their design is done completely independent of the essentialityscores. The complete list of the questions are included in Table 39.

In the following two examples we take questions which TableILP answers correct for the right reason (hence we are sure that the solver has the right knowledge to answer it, as well as almost correct fuzzy matching scores). We change two words, while retaining the general meaning of the questions, but tricking the solver into answering it incorrectly.

Original question: A fox grows thicker fur as a season changes. This adaptation helps the fox to (A) find food(B) keep warmer(C) grow stronger(D) escape from predators
Generated question: An animal grows thicker hair as a season changes. This adaptation helps to (A) find food(B) keep warmer(C) grow stronger(D) escape from predators

Original question: Which force causes rocks to roll downhill? (A) gravity(B) friction(C) erosion(D) magnetism
Generated question: Which item causes objects to roll downhill? (A) gravity(B) friction(C) erosion(D) magnetism

A.3.4. Example predictions: before and after adding ET

Here in Table 41 we show two sample questions for the analysis done in Section 4.3 and in Table 40. In the first row we have a question which is correct been answered by TableILP , but for an incorrect reason (since it's using only "plan" to answer a question about "gasses used in photosynthesis"). Augmenting the solver with ET, the prediction turns into an incorrect one, mostly probably due to an inaccurate alignment scores. One can also blame low ET score of "use" which is about 0.4.

	Correctly answered by TableILP; turned incorrect in Cascades	Inocorrect answered by TableILP; turned correct in Cascades
Correct reasoning	5	4
Partially correct	7	7
Got lucky	11	2
Sum	23	13

Table 40: Breakdown of the predictions changed from adding TableILP to Cascades, classified according to their reasons, annotated manually by inspecting the reasoning graph of each question.

In the second row we have an incorrect answer by TableILPwhich is turned into a correct prediction by adding ET, and the reasoning chain is mostly correct except a few noisy edges (e.g. "flag"-"lift objects" edge).

A.3.5. Error Analysis for TableILP+ET

We manually analyzed the predictions of TableILP vs. Cascades$(0.4, 0.6, 0.8, 1.0)$ on the *training* set of AI2Public (the test questions remain hidden). Out of 432 questions in total, the two systems differ on 57 questions. Out of these, 23 correct ones are turned into incorrect, and 13 incorrect predictions are turned correct, as a result of adding ET to TableILP. Hence, the overall performance drops.

One annotator went through the reasoning graphs of these questions and classified them into multiple groups base on how convincing the reasoning used by the solver was. Table 40 shows breakdown of these questions and their classes. The row "correct alignment" is when the required knowledge is there and it's used correctly in the reasoning graph. The row "partially correct" is when there are some noisy alignments as well as the correct alignment. The row "got lucky" are correct predictions, but for wrong reasons: not having the necessary knowledge, not performing the natural reasoning, etc.

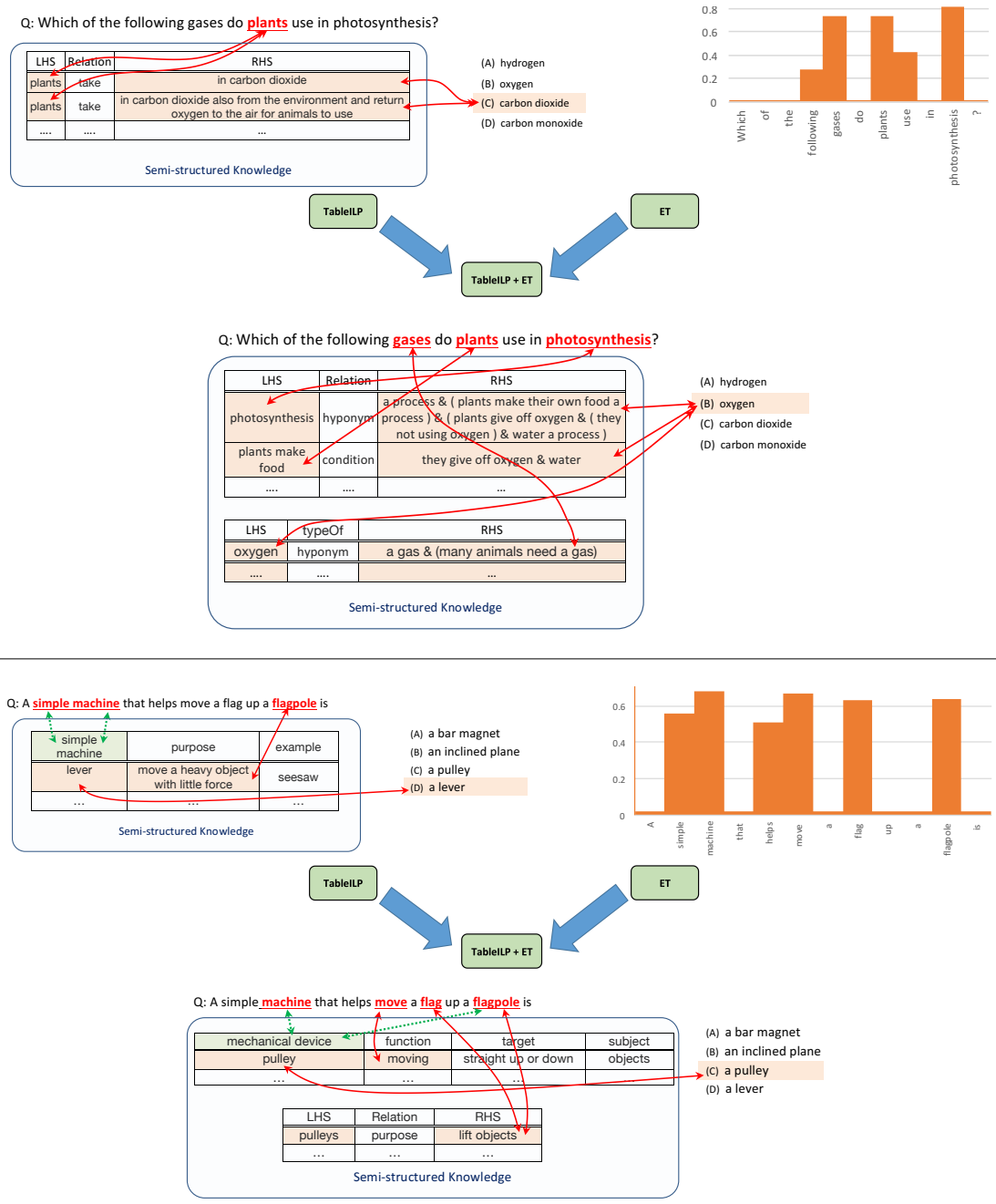


Table 41: Comparison of the reasoning graphs, for TABLEILP and CASCADES(TABLEILP+ET). In the first row, adding ET changes the correct prediction to incorrect, but in the second row, it corrects the incorrect prediction.

As can be seen, among the prediction changes, size of the resulted correct predictions are more reliable than the questions that were correct but turned incorrect after adding ET. In other words, the drop we see in performance of TableILP after adding ET is due to existence of other inaccurate solver ingredients. In Appendix IV, we provide a visualization of two questions for each of these categories to give more intuition to the reader.

A.4. Supplementary Details for Chapter 6

A.5. Supplementary Details for Chapter 7

A.5.1. Perturbing Candidate Answers

Here we provide a few missing details from *Step 3* of our annotations (Section 3). In particular, we create collections of common temporal expressions (see Table 42) to detect whether the given candidate answer contains a temporal expression or not. If a match is found within this list, we use the mappings to create perturbations of the temporal expression.

Adjectives	Frequency	Period	Typical time	Units
early:late	always:sometimes:never	night:day	now:later	second:hour:week:year
late:early	occasionally:always:never	day:night	today:yesterday	seconds:hours:weeks:years
morning:late night	often:rarely		tomorrow:yesterday	minute:day:month:century
night:early morning	usually:rarely		tonight:last night	minutes:days:months:centuries
evening:morning	rarely:always		yesterday:tomorrow	hour:second:week:year
everlasting:periodic	constantly:sometimes		am:pm	hours:seconds:weeks:years
initial:last	never:sometimes:always		pm:am	day:minute:month:century
first:last	regularly:occasionally:never		a.m.:p.m.	days:minutes:months:centuries
last:first			p.m.:a.m.	week:second:hour:year
overdue:on time			afternoon:morning	weeks:seconds:hours:years
belated:punctual			morning:evening	month:minute:day:century
long-term:short-term			night:morning	months:minutes:days:centuries
delayed:early			after:before	year:second:hour:week
punctual:belated			before:after	years:seconds:hours:weeks
				century:minute:day:month
				centuries:minutes:days:months

Table 42: Collections of temporal expressions used in creating perturbation of the candidate answers. Each mention is grouped with its variations (e.g., "first" and "last" are in the same set).

A.5.2. Performance as a function of training size

An intuition that we stated is that, the task at hand requires a successful model to bring in external world knowledge beyond what is observed in the dataset; since for a task like this, it is unlikely to compile an dataset which covers all the possible events and their attributes. In other words, the "traditional" supervised learning alone (with no pre-training or external training) is unlikely to succeed. A corollary to this observation is that, tuning a pre-training system (such as BERT Devlin et al. (2018)) likely requires very little supervision.

We plot the performance change, as a function of number of instances observed in the training time (Figure 34). Each point in the figure share the same parameters and averages of 5 distinct trials over different random sub-samples of the dataset. As it can be observed, the performance plateaus after about $2.5k$ question-answer pairs (about 20% of the whole datasets). This verifies the intuition that systems can rely on a relatively small amount of supervision to tune to task, if it models the world knowledge through pre-training. Moreover, it shows that trying to make improvement through getting more labeled data is costly and impractical.

A.5.3. Metrics

We use two question level metrics to compare performances of the systems. For a given candidate answer a that belongs to a question q, let $f(a; q) \in \{0, 1\}$ denote the correctness of the prediction made by a fixed system (1 for correct; 0 otherwise). Additionally, let D denote the collection of questions in our evaluation set.

- Exact Match (EM):

$$EM \triangleq \frac{\sum_{q \in D} \prod_{a \in q} f(a; q)}{|\{q \in D\}|}$$

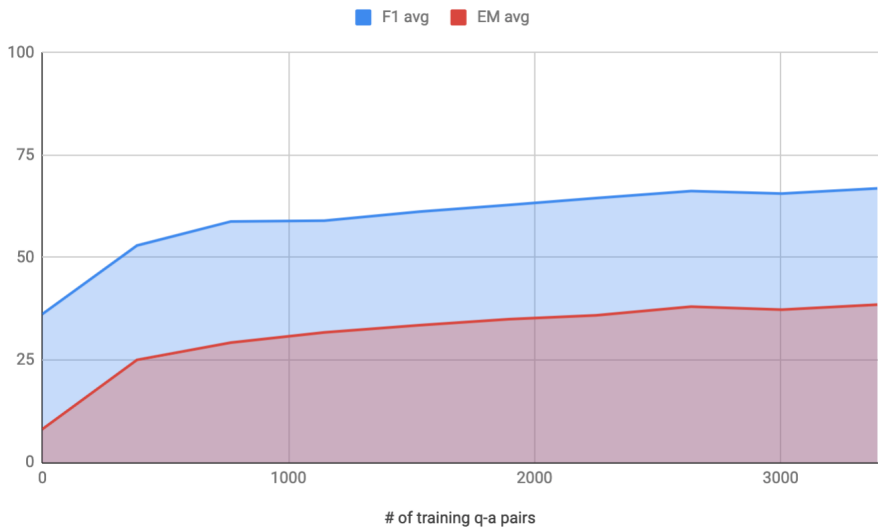


Figure 34: Performance of supervised algorithm (BERT; Section 4) as function of various sizes of observed training data. When no training data provided to the systems (left-most side of the figure), the performance measures amount to random guessing.

- Define $F1(q)$ with precision-recalls of the predictions for a fixed question.

$$P(q) = \frac{\sum_{a \in q} [f(a; q) = 1] \wedge [a \text{ is correct }]}{|\{a \text{ is correct } \wedge a \in q\}|}$$

  Similarly $R(q)$ and $F1(q)$ are defined. And the aggregate $F1$ on a dataset D is an average of question-level $F1$s:

$$F1 \triangleq \frac{\sum_{q \in D} F1(q)}{|\{q \in D\}|}$$

EM is a stricter metric since it assigns credit only if all the candidate answers are predicted correctly. In our opinion, this is more appropriate metric, since it directly measures question understanding: only if a system completely understands a question, it should get all it candidates correctly. $F1$ is a more relaxed version of EM, but it could be misleading to solely rely on this metric since the gaps are smaller.

A.6. Supplementary Details for Chapter 8

We here provide detailed proofs of the formal results, followed by additional experiments. The following observation allows a simplification of the proofs, without loss of any generality.

Remark 1. Since our procedure doesn't treat similarity edges and meaning-to-symbol noise edges differently, we can 'fold' ε_- into p_- and p_+ (by increasing edge probabilities). More generally, the results are identical whether one uses p_+, p_-, ε_- or $p'_+, p'_-, \varepsilon'_-$, as long as:

$$\begin{cases} p_+ \oplus \varepsilon_- = p'_+ \oplus \varepsilon'_- \\ p_- \oplus \varepsilon_- = p'_- \oplus \varepsilon'_- \end{cases}$$

For any p_+ and ε_-, we can find a p_+ such that $\varepsilon_- = 0$. Thus, w.l.o.g., in the following analysis we derive results only using p_+ and p_- (i.e. assume $\varepsilon_- = 0$). Note that we expand these terms to $p_+ \oplus \varepsilon_-$ and $p_- \oplus \varepsilon_-$ respectively in the final results.

A.6.1. Proofs: Possibility of Accurate Connectivity Reasoning

In this section we provide the proofs of the additional lemmas necessary for proving the intermediate results. First we introduce a few useful lemmas, and then move on to the proof of Theorem 1.

We introduce the following lemmas which will be used in connectivity analysis of the clusters of the nodes $\mathcal{O}(m)$.

Lemma 3 (Connectivity of a random graph (Gilbert, 1959))**.** Let P_n denote the probability of the event that a random undirected graph $\mathsf{G}(\mathsf{n}, \mathsf{p})$ ($p > 0.5$) is connected. This probability can be lower-bounded as following:

$$P_n \geq 1 - \left[q^{n-1} \left\{ (1 + q^{(n-2)/2})^{n-1} - q^{(n-2)(n-1)/2} \right\} + q^{n/2} \left\{ (1 + q^{(n-2)/2})^{n-1} - 1 \right\} \right],$$

where $q = 1 - p$.

See Gilbert (1959) for a proof of this lemma. Since $q \in (0,1)$, this implies that $P_n \to 1$ as n increases. The following lemma provides a simpler version of the above probability:

Corollary 2 (Connectivity of a random graph (Gilbert, 1959))**.** The random-graph connectivity probability P_n (Lemma 3) can be lower-bounded as following:

$$P_n \geq 1 - 2e^3 q^{n/2}$$

Proof. We use the following inequality:

$$(1 + \frac{3}{n})^n \leq e^3$$

Given that $q \leq 0.5, n \geq 1$, one can verify that $q^{(n-2)/2} \leq 3/n$. Combining this with the above inequality gives us, $(1 + q^{n-2/2})^{n-1} \leq e^3$.

With this, we bound the two terms within the two terms of the target inequality:

$$\begin{cases} (1 + q^{(n-2)/2})^{n-1} - q^{(n-2)(n-1)/2} \leq e^3 \\ (1 + q^{(n-2)/2})^{n-1} - 1 \leq e^3 \end{cases}$$

$$\left[q^{n-1}\left\{(1 + q^{(n-2)/2})^{n-1} - q^{(n-2)(n-1)/2}\right\} + q^{n/2}\left\{(1 + q^{(n-2)/2})^{n-1} - 1\right\}\right] \leq e^3 q^{n-1} + e^3 q^{n/2} \leq 2e^3 q^{n/2}$$

which concludes the proof.

□

We show a lower-bound on the probability of s and s' being connected given the connectivity of their counterpart nodes in the meaning graph. This lemma will be used in the proof of Theorem 1:

Lemma 4 (Lower bound)**.** $\mathbb{P}\left[s \overset{\tilde{d}}{\rightsquigarrow} s' | m \overset{d}{\rightsquigarrow} m'\right] \geq \left(1 - 2e^3\varepsilon_+^{\lambda/2}\right)^{d+1} \cdot \left(1 - (1-p_+)^{\lambda^2}\right)^d$.

Proof. We know that m and m' are connected through some intermediate nodes $m_1, m_2, \cdots, m_\ell$ ($\ell < d$). We show a lower-bound on having a path in the symbol-graph between s and s', through clusters of nodes $\mathcal{O}(m_1), \mathcal{O}(m_2), \cdots, \mathcal{O}(m_\ell)$. We decompose this into two events:

$$\begin{cases} e_1[v] & \text{For a given meaning node } v \text{ its cluster in the symbol-graph, } \mathcal{O}(v) \text{ is connected.} \\ e_2[v,u] & \text{For any two connected nodes } (u,v) \text{ in the meaning graph, there is at least an edge} \\ & \text{connecting their clusters } \mathcal{O}(u), \mathcal{O}(v) \text{ in the symbol-graph.} \end{cases}$$

The desired probability can then be refactored as:

$$\begin{aligned} \mathbb{P}\left[s \overset{\tilde{d}}{\rightsquigarrow} s' | m \overset{d}{\rightsquigarrow} m'\right] &\geq \mathbb{P}\left[\left(\bigcap_{v \in \{s, m_1, \ldots, m_\ell, s'\}} e_1[v]\right) \cap \left(\bigcap_{(v,u) \in \{(s,m_1), \ldots, (m_\ell, s')\}} e_2[v,u]\right)\right] \\ &\geq \mathbb{P}\left[e_1\right]^{d+1} \cdot \mathbb{P}\left[e_2\right]^d. \end{aligned}$$

We split the two probabilities and identify lower bounds for each. Based on Corollary 2, $\mathbb{P}\left[e_1\right] \geq 1 - 2e^3\varepsilon_+^{\lambda/2}$, and as a result $\mathbb{P}\left[e_1\right]^{d+1} \geq \left(1 - 2e^3\varepsilon_+^{\lambda/2}\right)^{d+1}$. The probability of connectivity between pair of clusters is $\mathbb{P}\left[e_2\right] = 1 - (1-p_+)^{\lambda^2}$. Thus, similarly, $\mathbb{P}\left[e_2\right]^d \geq \left(1 - (1-p_+)^{\lambda^2}\right)^d$. Combining these two, we obtain:

$$\mathbb{P}\left[s \overset{\tilde{d}}{\rightsquigarrow} s' | m \overset{d}{\rightsquigarrow} m'\right] \geq \left(1 - 2e^3\varepsilon_+^{\lambda/2}\right)^{d+1} \cdot \left(1 - (1-p_+)^{\lambda^2}\right)^d \tag{A.55}$$

□

The connectivity analysis of G_S can be challenging since the graph is a non-homogeneous combination of positive and negative edges. For the sake of simplifying the probabilistic arguments, given symbol graph G_S, we introduce a non-unique simple graph $\tilde{G}_S$ as follows.

Definition 11. Consider a special partitioning of V_G such that the d-neighbourhoods of s

and s' form two of the partitions and the rest of the nodes are arbitrarily partitioned in a way that the diameter of each component does not exceed $\tilde{d}$.

- The set of nodes $V_{\tilde{G}_S}$ of $\tilde{G}_S$ corresponds to the aforementioned partitions.
- There is an edge $(u, v) \in E_{\tilde{G}_S}$ if and only if at least one node-pair from the partitions of V_G corresponding to u and v, respectively, is connected in E_{G_S}.

In the following lemma we give an upper-bound on the connectivity of neighboring nodes in $\tilde{G}_S$:

Lemma 5. When G_S is drawn at random, the probability that an edge connects two arbitrary nodes in $\tilde{G}_S$ is at most $(\lambda\mathcal{B}(d))^2 p_-$.

Proof. Recall that a pair of nodes from $\tilde{G}_S$, say (u, v), are connected when at least one pair of nodes from corresponding partitions in G_S are connected. Each d-neighbourhood in the meaning graph has at most $\mathcal{B}(d)$ nodes. It implies that each partition in $\tilde{G}_S$ has at most $\lambda\mathcal{B}(d)$ nodes. Therefore, between each pair of partitions, there are at most $(\lambda\mathcal{B}(d))^2$ possible edges. By union bound, the probability of at least one edge being present between two partitions is at most $(\lambda\mathcal{B}(d))^2 p_-$. □

Let $v_s, v_{s'} \in V_{\tilde{G}_S}$ be the nodes corresponding to the components containing s and s' respectively. The following lemma establishes a relation between connectivity of $s, s' \in V_{G_S}$ and the connectivity of $v_s, v_{s'} \in V_{\tilde{G}_S}$:

Lemma 6. $\mathbb{P}\left[s \overset{\tilde{d}}{\rightsquigarrow} s' | m \rightsquigarrow m'\right] \leq \mathbb{P}\left[\text{There is a path from } v_s \text{ to } v_{s'} \text{ in } \tilde{G}_S \text{ with length } \tilde{d}\right]$.

Proof. Let L and R be the events in the left hand side and right hand side respectively. Also for a permutation of nodes in G_S, say p, let F_p denote the event that all the edges of p are present, i.e., $L = \cup F_p$. Similarly, for a permutation of nodes in $\tilde{G}_S$, say q, let H_q denote the event that all the edges of q are present. Notice that $F_p \subseteq H_q$ for $q \subseteq p$, because if all

the edges of p are present the edges of q will be present. Thus,

$$L = \bigcup_p F_p \subseteq \bigcup_p H_{p \cap E_{\tilde{G}_S}} = \bigcup_q H_q = R.$$

This implies that $\mathbb{P}[L] \leq \mathbb{P}[R]$. □

Lemma 7 (Upper bound)**.** If $(\lambda \mathcal{B}(d))^2 p_- \leq \frac{1}{2en}$, then $\mathbb{P}\left[s \overset{\leq \tilde{d}}{\nsim} s' \mid m \sim\!\!\!\!\!\!\sim m'\right] \leq 2en(\lambda \mathcal{B}(d))^2 p_-$.

Proof. To identify the upper bound on $\mathbb{P}\left[s \overset{\leq \tilde{d}}{\nsim} s' | m \sim\!\!\!\!\!\!\sim m'\right]$, recall the definition of $\tilde{G}_S$, given an instance of G_S (as outlined in Lemmas 5 and 6, for $\tilde{p} = (\lambda \mathcal{B}(d))^2 p_-$). Lemma 6 relates the connectivity of s and s' to a connectivity event in $\tilde{G}_S$, i.e., $\mathbb{P}\left[s \overset{\leq \tilde{d}}{\nsim} s' \mid m \sim\!\!\!\!\!\!\sim m'\right] \leq \mathbb{P}\left[\text{there is a path from } v_s \text{ to } v_{s'} \text{ in } \tilde{G}_S \text{ with length } \tilde{d}\right]$, where $v_s, v_{s'} \in V_{\tilde{G}_S}$ are the nodes corresponding to the components containing s and s' respectively. Equivalently, in the following, we prove that the event $\text{dist}(v_s, v_{s'}) \leq \tilde{d}$ happens with a small probability:

$$\mathbb{P}\left[s \overset{\leq \tilde{d}}{\nsim} s'\right] = \mathbb{P}\left[\bigvee_{\ell=1,\cdots,\tilde{d}} s \overset{\ell}{\nsim} s'\right] \leq \sum_{\ell \leq \tilde{d}} \binom{n}{\ell} \tilde{p}^\ell \leq \sum_{\ell \leq \tilde{d}} (\frac{en}{\ell})^\ell \tilde{p}^\ell$$

$$\leq \sum_{\ell \leq \tilde{d}} (en)^\ell \tilde{p}^\ell \leq en\tilde{p} \frac{(en\tilde{p})^{\tilde{d}} - 1}{en\tilde{p} - 1} \leq \frac{en\tilde{p}}{1 - en\tilde{p}} \leq 2en\tilde{p}.$$

where the final inequality uses the assumption that $\tilde{p} \leq \frac{1}{2en}$.

□

Armed with the bounds in Lemmas 4 and 7, we are ready to provide the main proof:

Proof of Theorem 1. Recall that the algorithm checks for connectivity between two given nodes s and s', i.e., $s \overset{\leq \tilde{d}}{\nsim} s'$. With this observation, we aim to infer whether the two nodes in the meaning graph are connected ($m \overset{\leq d}{\nsim} m'$) or not ($m \sim\!\!\!\!\!\!\sim m'$). We prove the theorem

by using lower and upper bound for these two probabilities, respectively:

$$\begin{aligned}\gamma &= \mathbb{P}\left[s \overset{\leq\tilde{d}}{\leftrightsquigarrow} s' | m \overset{d}{\leftrightsquigarrow} m'\right] - \mathbb{P}\left[s \overset{\leq\tilde{d}}{\leftrightsquigarrow} s' | m \not\leftrightsquigarrow m'\right] \\ &\geq LB\left(\mathbb{P}\left[s \overset{\leq\tilde{d}}{\leftrightsquigarrow} s' | m \overset{d}{\leftrightsquigarrow} m'\right]\right) - UB\left(\mathbb{P}\left[s \overset{\leq\tilde{d}}{\leftrightsquigarrow} s' | m \not\leftrightsquigarrow m'\right]\right) \\ &\geq \left(1 - 2e^3\varepsilon_+^{\lambda/2}\right)^{d+1} \cdot \left(1 - (1-p_+)^{\lambda^2}\right)^d - 2en(\lambda\mathcal{B}(d))^2 p_-.\end{aligned}$$

where the last two terms of the above inequality are based on the results of Lemmas 4 and 7, with the assumption for the latter that $(\lambda\mathcal{B}(d))^2 p_- \leq \frac{1}{2en}$. To write this result in its general form we have to replace p_+ and p_-, with $p_+ \oplus \varepsilon_-$ and $p_- \oplus \varepsilon_-$, respective (see Remark 1).

□

A.6.2. Proofs: Limitations of Connectivity Reasoning

We provide the necessary lemmas and intuitions before proving the main theorem.

A random graph is an instance sampled from a distribution over graphs. In the $\mathsf{G}(\mathsf{n},\mathsf{p})$ Erdős-Renyi model, a graph is constructed in the following way: Each edge is included in the graph with probability p, independent of other edges. In such graphs, on average, the length of the path connecting any node-pair is short (logarithmic in the number of nodes).

Lemma 8 (Diameter of a random graph, Corollary 1 of (Chung and Lu, 2002))**.** If $n \cdot p = c > 1$ for some constant c, then almost-surely the diameter of $\mathsf{G}(\mathsf{n},\mathsf{p})$ is $\Theta(\log n)$.

We use the above lemma to prove Theorem 2. Note that the overall noise probably (i.e., p in Lemma 8) in our framework is $p_- \oplus \varepsilon_-$.

Proof of Theorem 2. Note that the $|V_{G_S}| = \lambda \cdot n$. By Lemma 8, the symbol graph has diameter $\Theta(\log \lambda n)$. This means that for any pair of nodes $s, s' \in V_{G_S}$, we have $s \overset{\Theta(\log \lambda n)}{\leftrightsquigarrow} s'$. Since $\tilde{d} \geq \lambda d \in \Omega(\log \lambda n)$, the multi-hop reasoning algorithm finds a path between s and s'

in symbol graph and returns connected regardless of the connectivity of m and m'. □

A.6.3. Proofs: Limitations of General Reasoning

The proof of the theorem follows after introducing necessary lemmas.

In the following lemma, we show that the spectral differences between the two symbol graphs in the locality of the target nodes are small. For ease of exposition, we define an intermediate notation, for a normalized version of the Laplacians: $\tilde{L} = L/\|L\|_2$ and $\tilde{L}' = L'/\|L'\|_2$.

Lemma 9. The norm-2 of the Laplacian matrix corresponding to the nodes participating in a cut, can be upper-bounded by the number of the edges participating in the cut (with a constant factor).

Proof of Lemma 9. Using the definition of the Laplacian:

$$\|L_C\|_2 \leq \|A - D\|_2 \leq \|A\|_2 + \|D\|_2$$

where A is the adjacency matrix and D is a diagonal matrix with degrees on the diagonal. We bound the norms of the matrices based on size of the cut (i.e., number of the edges in the cut). For the adjacency matrix we use the Frobenius norm:

$$\|A\|_2 \leq \|A\|_F = \sqrt{\sum_{ij} a_{ij}} = 2 \cdot |C|$$

where $|C|$ denotes the number of edges in C. To bound the matrix of degrees, we use the fact that norm-2 is equivalent to the biggest eigenvalue, which is the biggest diagonal element in a diagonal matrix:

$$\|D\|_2 = \sigma_{\max}(D) = \max_i deg(i) \leq |C|$$

With this we have shown that: $\|L_C\|_2 \leq 3|C|$. □

For sufficiently large values of p, $\mathsf{G}(\mathsf{n},\mathsf{p})$ is a connected graph, with a high probability. More formally:

Lemma 10 (Connectivity of random graphs)**.** In a random graph $\mathsf{G}(\mathsf{n},\mathsf{p})$, for any p bigger than $\frac{(1+\varepsilon)\ln n}{n}$, the graph will almost surely be connected.

The proof can be found in (Erdos and Rényi, 1960).

Lemma 11 (Norm of the adjacency matrix in a random graph)**.** For a random graph $\mathsf{G}(\mathsf{n},\mathsf{p})$, let L be the adjacency matrix of the graph. For any $\varepsilon > 0$:

$$\lim_{n\to+\infty} \mathbb{P}\left(\left|\|L\|_2 - \sqrt{2n\log n}\right| > \varepsilon\right) \to 0$$

Proof of Lemma 11. From Theorem 1 of (Ding et al., 2010) we know that:

$$\frac{\sigma_{\max}(L)}{\sqrt{n\log n}} \xrightarrow{P} \sqrt{2}$$

where $\xrightarrow{P}$ denote *convergence in probability*. And also notice that norm-2 of a matrix is basically the size of its biggest eigenvalue, which concludes our proof. □

Lemma 12. For any pair of meaning-graphs G and G' constructed according to Definition 9, and,

- $d > \log n$,
- $p_- \oplus \varepsilon_- \geq c\log n/n$ for some constant c,
- $\tilde{d} \geq \lambda d$,

with L and L' being the Laplacian matrices corresponding to the $\tilde{d}$-neighborhoods of the corresponding nodes in the surface-graph; we have:

$$\frac{\|L - L'\|_2}{\|L\|_2} \leq \frac{\sqrt{\lambda}\mathcal{B}(1)}{\sqrt{2n\log(n\lambda)}},$$

with a high-probability.

Proof of Lemma 12. In order to simplify the exposition, w.l.o.g. assume that $\varepsilon_- = 0$ (see Remark 1). Our goal is to find an upper-bound to the fraction $\frac{\|L-L'\|_2}{\|L\|_2}$. Note that the Laplacians contain only the local information, i.e., $\tilde{d}-$neighborhood. First we prove an upper bound on the nominator. By eliminating an edge in a meaning-graph, the probability of edge appearance in the symbol graph changes from p_+ to p_-. The effective result of removing edges in C would appear as i.i.d. Bern$(p_+ - p_-)$. Since by definition, $\mathcal{B}(1)$ is an upper bound on the degree of meaning nodes, the size of minimum cut should also be upper bounded by $\mathcal{B}(1)$. Therefore, the maximum size of the min-cut C separating two nodes $m \overset{d}{\rightsquigarrow} m'$ is at most $\mathcal{B}(1)$. To account for vertex replication in symbol-graph, the effect of cut would appear on at most $\lambda\mathcal{B}(1)$ edges in the symbol graph. Therefore, we have$\|L - L'\|_2 \leq \lambda\mathcal{B}(1)$ using Lemma 9.

As for the denominator, the size of the matrix L is the same as the size of $\tilde{d}$-neighborhood in the symbol graph. We show that if $\tilde{d} > \log(\lambda n)$ the neighborhood almost-surely covers the whole graph. While the growth in the size of the $\tilde{d}$-neighborhood is a function of both p_+ and p_-, to keep the analysis simple, we underestimate the neighborhood size by replacing p_+ with p_-, i.e., the size of the $\tilde{d}$-neighborhood is lower-bounded by the size of a $\tilde{d}$-neighborhood in $\mathsf{G}(\lambda \cdot n, p_-)$.

By Lemma 10 the diameters of the symbol-graphs G_S and G'_S are both $\Theta(\log(\lambda n))$. Since $\tilde{d} \in \Omega(\log(\lambda n))$, $\tilde{d}$-neighborhood covers the whole graph for both G_S and G'_S.

Next, we use Lemma 11 to state that $\|L\|_2$ converges to $\sqrt{2\lambda n \log(\lambda n)}$, in probability.

Combining numerator and denominator, we conclude that the fraction, for sufficiently large n, is upper-bounded by: $\frac{\lambda\mathcal{B}(1)}{\sqrt{2\lambda n \log(\lambda n)}}$, which can get arbitrarily small, for a big-enough choice of n.

□

Proof of Lemma 1. We start by proving an upper bound on $\tilde{L} - \tilde{L}'$ in matrix inequality notation. Similar upper-bound holds for $\tilde{L}' - \tilde{L}$ which concludes the theorem.

$$\begin{aligned}
\tilde{L} - \tilde{L}' &= \frac{L}{\|L\|} - \frac{L'}{\|L'\|} \\
&\preceq \frac{L}{\|L\|} - \frac{L'}{\|L - L'\| + \|L\|} \\
&= \frac{L \cdot \|L - L'\|}{\|L\|^2} + \frac{L - L'}{\|L\|} \\
&\preceq \frac{\sqrt{\lambda}\mathcal{B}(1)}{\sqrt{2n\log(n\lambda)}} I + \frac{\sqrt{\lambda}\mathcal{B}(1)}{\sqrt{2n\log(n\lambda)}} I.
\end{aligned}$$

The last inequality is due to Lemma 12. By symmetry the same upper-bound holds for $\tilde{L}' - \tilde{L} \preceq 2\frac{\sqrt{\lambda}\mathcal{B}(1)}{\sqrt{2n\log(n\lambda)}} I$. This means that $\|\tilde{L} - \tilde{L}'\| \leq \frac{2\sqrt{\lambda}\mathcal{B}(1)}{\sqrt{2n\log(n\lambda)}}$. □

Lemma 13. Suppose f is an indicator function on an open set[3], it is always possible to write it as composition of two functions:

- A continuous and Lipschitz function: $g : \mathbb{R}^d \to (0, 1)$,
- A thresholding function: $H(x) = \mathbf{1}\{x > 0.5\}$.

such that: $\forall x \in \mathbb{R}^d : \quad f(x) = h(g(x))$.

Proof of Lemma 13. Without loss of generality, we assume that the threshold function is defined as $H(x) = \mathbf{1}\{x > 0.5\}$. One can verify that a similar proof follows for $H(x) = \mathbf{1}\{x \geq 0.5\}$. We use notation $f^{-1}(A)$ the set of pre-images of a function f, for the set of outputs A.

First let's study the collection of inputs that result in output of 1 in f function. Since $f = h \circ g$, then $f^{-1}(\{1\}) = g^{-1}(h^{-1}(\{1\})) = g^{-1}((0.5, 1))$ and $f^{-1}(\{0\}) = g^{-1}(h^{-1}(\{0\})) = g^{-1}((0, 0.5))$. Define C_0 and C_1, such that $C_i \triangleq f^{-1}(\{i\})$; note that since g is continuous

[3] `https://en.wikipedia.org/wiki/Indicator_function`

and $(0.5, 1)$ is open C_1 is an open set (hence C_1 is closed). Let $d : \mathbb{R}^n \to \mathbb{R}$ be defined by,

$$d(x) \triangleq \text{dist}(x, C_0) = \inf_{c \in C_0} \|x - c\|.$$

Since C_0 is closed, it follows $d(x) = 0$ if and only if $x \in C_0$. Therefore, letting

$$g(x) = \frac{1}{2} + \frac{1}{2} \cdot \frac{d(x)}{1 + d(x)},$$

then $g(x) = \frac{1}{2}$ when $x \in C_0$, while $g(x) > \frac{1}{2}$ when $x \notin C_0$. This means that letting $h(x) = 1$ when $x > \frac{1}{2}$ and $h(x) = 0$ when $x \leq \frac{1}{2}$, then $f = h \circ g$. One can also verify that this construction is 1/2-Lipschitz; this follows because $d(x)$ is 1-Lipschitz, which can be proved using the triangle inequality

Hence the necessary condition to have such decomposition is $f^{-1}(\{1\})$ and $f^{-1}(\{0\})$ be open or closed. □

Proof of Lemma 2. Note that f maps a high dimensional continuous space to a discrete space. To simplify the argument about f, we decompose it to two functions: a continuous function g mapping matrices to $(0, 1)$ and a threshold function H (e.g. $0.5 + 0.5\text{sgn}(.)$) which maps to one if g is higher than a threshold and to zero otherwise. Without loss of generality we also normalize g such that the gradient is less than one. Formally,

$$f = H \circ g, \text{ where } g : \mathbb{R}^{|\mathcal{U}| \times |\mathcal{U}|} \to (0, 1), \|\nabla g\Big|_{\tilde{L}}\| \leq 1.$$

Lemma 13 gives a proof of existence for such decompositon, which depends on having open or closed pre-images.

One can find a differentiable and Lipschitz function g such that it intersects with the threshold specified by H, in the borders where f changes values.

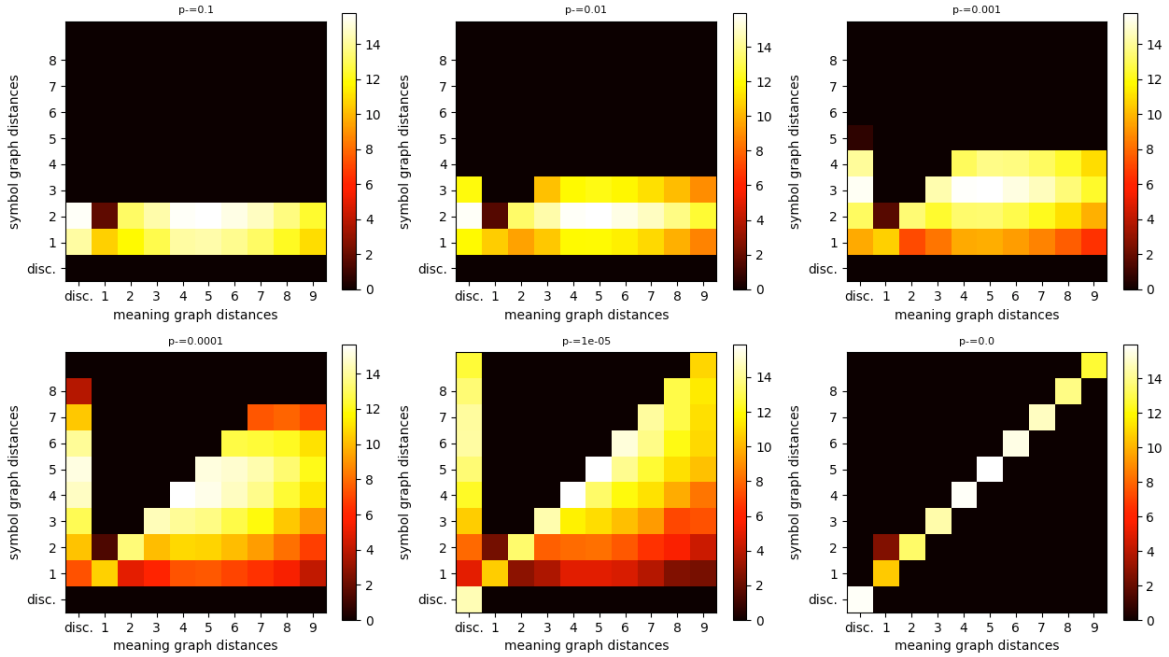


Figure 35: With varied values for p_- a heat map representation of the distribution of the average distances of node-pairs in symbol graph based on the distances of their corresponding meaning nodes is presented.

With g being Lipschitz, one can upper-bound the variations on the continuous function:

$$\|g(\tilde{L}) - g(\tilde{L}')\| \leq M\|\tilde{L} - \tilde{L}'\|.$$

According to Lemma 1, $\|\tilde{L} - \tilde{L}'\|$ is upper-bounded by a decreasing function in n.

For uniform choices $(G, G', m, m') \sim \mathcal{G}$ the Laplacian pairs $(\tilde{L}, \tilde{L}')$ are randomly distributed in a high-dimensional space, and for big enough n, there are enough portion of the $(\tilde{L}, \tilde{L}')$ (to satisfy $1 - \beta$ probability) that appear in the same side of the hyper-plane corresponding to the threshold function (i.e. $f(\tilde{L}) = f(\tilde{L}')$). □

A.6.4. Further experiments

To evaluate the impact of the other noise parameters in the sampling process, we compare the average distances between nodes in the symbol graph for a given distance between the

meaning graph nodes. In the Figure 35, we plot these graphs for decreasing values of p_- (from top left to bottom right). With high p_- (top left subplot), nodes in the symbol graph at distances lower than two, regardless of the distance of their corresponding node-pair in the meaning graph. As a result, any reasoning algorithm that relies on connectivity can not distinguish symbolic nodes that are connected in the meaning space from those that are not. As the p_- is set to lower values (i.e. noise reduces), the distribution of distances get wider, and correlation of distance between the two graphs increases. In the bottom middle subplot, when p_- has a very low value, we observe a significant correlation that can be reliably utilized by a reasoning algorithm.

BIBLIOGRAPHY

T. Achterberg. SCIP: solving constraint integer programs. *Math. Prog. Computation*, 1(1): 1–41, 2009.

G. Angeli and C. D. Manning. NaturalLI: Natural Logic Inference for Common Sense Reasoning. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2014.

N. Arivazhagan, C. Christodoulopoulos, and D. Roth. Labeling the semantic roles of commas. In *AAAI*, 2016.

C. F. Baker, C. J. Fillmore, and J. B. Lowe. The berkeley framenet project. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 86–90, 1998.

D. Bamman, B. O'Connor, and N. A. Smith. Learning Latent Personas of Film Characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, Volume 1: Long Papers*, pages 352–361, 2013. URL `http://aclweb.org/anthology/P/P13/P13-1035.pdf`.

L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, M. Palmer, and N. Schneider. Abstract meaning representation for sembanking. In *Linguistic Annotation Workshop and Interoperability with Discourse*, 2013.

M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open Information Extraction from the Web. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

R. Bar-Haim, I. Dagan, and J. Berant. Knowledge-Based Textual Inference via Parse-Tree Transformations. *J. Artif. Intell. Res.(JAIR)*, 54:1–57, 2015.

L. Bauer, Y. Wang, and M. Bansal. Commonsense for Generative Multi-Hop Question Answering Tasks. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 4220–4230, 2018.

L. Bentivogli, P. Clark, I. Dagan, and D. Giampiccolo. The Sixth PASCAL Recognizing Textual Entailment Challenge. In *TAC*, 2008.

J. Berant, I. Dagan, and J. Goldberger. Global learning of focused entailment graphs. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 1220–1229, 2010.

J. Berant, V. Srikumar, P.-C. Chen, A. V. Linden, B. Harding, B. Huang, P. Clark, and C. D. Manning. Modeling Biological Processes for Reading Comprehension. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2014.

V. W. Berninger, W. Nagy, and S. Beers. Child writers construction and reconstruction of single sentences and construction of multi-sentence texts: Contributions of syntax and transcription to translation. *Reading and writing*, 24(2):151–182, 2011.

A. M. Bisantz and K. J. Vicente. Making the abstraction hierarchy concrete. *International Journal of human-computer studies*, 40(1):83–117, 1994.

D. G. Bobrow. Natural language input for a computer problem solving system. Technical report, MIT, 1964.

K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *ICMD*, pages 1247–1250. ACM, 2008.

R. Brachman, D. Gunning, S. Bringsjord, M. Genesereth, L. Hirschman, and L. Ferro. Selected Grand Challenges in Cognitive Science. Technical report, MITRE Technical Report 05-1218, 2005.

E. Brill, S. Dumais, and M. Banko. An analysis of the AskMSR question-answering system. In *Proceedings of EMNLP*, pages 257–264, 2002.

P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

J. G. Carbonell and R. D. Brown. Anaphora resolution: a multi-strategy approach. In *Proceedings of the 12th conference on Computational linguistics-Volume 1*, pages 96–101. Association for Computational Linguistics, 1988.

A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell. Toward an Architecture for Never-Ending Language Learning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2010.

K.-W. Chang, S. Upadhyay, M.-W. Chang, V. Srikumar, and D. Roth. Illinois-SL: A JAVA library for structured prediction. *arXiv preprint arXiv:1509.07179*, 2015.

M.-W. Chang, L. Ratinov, N. Rizzolo, and D. Roth. Learning and inference with constraints. In *Proc. of the Conference on Artificial Intelligence (AAAI)*, 7 2008. URL `http://cogcomp.org/papers/CRRR08.pdf`.

M.-W. Chang, D. Goldwasser, D. Roth, and V. Srikumar. Discriminative Learning over Constrained Latent Representations. *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT 2010)*, (June):429–437, 2010.

M.-W. Chang, L. Ratinov, and D. Roth. Structured learning with constrained conditional models. *Machine Learning*, 88(3):399–431, 6 2012. URL `http://cogcomp.org/papers/ChangRaRo12.pdf`.

D. Chen, J. Bolton, and C. D. Manning. A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Volume 1: Long Papers*, 2016. URL `http://aclweb.org/anthology/P/P16/P16-1223.pdf`.

Q. Chen, X. Zhu, Z. Ling, S. Wei, H. Jiang, and D. Inkpen. Enhanced LSTM for Natural Language Inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, Vancouver, July 2017. ACL.

T. Chklovski and P. Pantel. VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In *EMNLP*, 2004.

F. Chung and L. Lu. The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences*, 99(25):15879–15882, 2002.

K. W. Church and P. Hanks. Word Association Norms, Mutual Information and Lexicography. In *27thACL*, pages 76–83, 1989.

P. Clark. Elementary School Science and Math Tests as a Driver for AI: Take the Aristo Challenge! In *29th AAAI/IAAI*, pages 4019–4021, Austin, TX, 2015.

P. Clark and O. Etzioni. My Computer is an Honor Student but how Intelligent is it? Standardized Tests as a Measure of AI. *AI Magazine*, 2016. (To appear).

P. Clark, N. Balasubramanian, S. Bhakthavatsalam, K. Humphreys, J. Kinkead, A. Sabharwal, and O. Tafjord. Automatic Construction of Inference-Supporting Knowledge Bases. In *4thAKBC Workshop*, Montreal, Canada, 2014.

P. Clark, O. Etzioni, T. Khot, A. Sabharwal, O. Tafjord, P. Turney, and D. Khashabi. Combining Retrieval, Statistics, and Inference to Answer Elementary Science Questions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2016.

P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *CoRR*, abs/1803.05457, 2018.

J. Clarke and M. Lapata. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429, 2008.

J. Clarke, D. Goldwasser, M.-W. Chang, and D. Roth. Driving semantic parsing from the world's response. In *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*, 7 2010. URL `http://cogcomp.org/papers/CGCR10.pdf`.

A. Cocos, V. Wharton, E. Pavlick, M. Apidianaki, and C. Callison-Burch. Learning Scalar Adjective Intensity from Paraphrases. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1752–1762, 2018.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2009.

I. Dagan, D. Roth, M. Sammons, and F. M. Zanzoto. Recognizing textual entailment: Models and applications. 7 2013.

B. Dalvi, S. Bhakthavatsalam, and P. Clark. IKE - An Interactive Tool for Knowledge Extraction. In *5thAKBC Workshop*, 2016.

H. T. Dang and M. Palmer. The role of semantic roles in disambiguating verb senses. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 42–49. Association for Computational Linguistics, 2005.

H. A. Davidson. *Alfarabi, Avicenna, and Averroes on intellect: their cosmologies, theories of the active intellect, and theories of human intellect*. Oxford University Press, 1992.

E. Davis. The Limitations of Standardized Science Tests as Benchmarks for Artificial Intelligence Research: Position Paper. *CoRR*, abs/1411.1629, 2014. URL `http://arxiv.org/abs/1411.1629`.

R. Dechter. Reasoning with Probabilistic and Deterministic Graphical Models: Exact Algorithms. In *Reasoning with Probabilistic and Deterministic Graphical Models: Exact Algorithms*, 2013.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

X. Ding, T. Jiang, et al. Spectral distributions of adjacency and Laplacian matrices of random graphs. *The annals of applied probability*, 20(6):2086–2117, 2010.

A. N. P. DivyeKhilnani and S. B. D. Jurafsky. Using Query Patterns to Learn the Duration of Events. *Computational Semantics IWCS 2011*, page 145, 2011.

Q. Do, Y. S. Chan, and D. Roth. Minimally supervised event causality identification. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Edinburgh, Scotland, 7 2011. URL `http://cogcomp.org/papers/DoChaRo11.pdf`.

Q. Do, W. Lu, and D. Roth. Joint inference for event timeline construction. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2012. URL `http://cogcomp.org/papers/DoLuRo12.pdf`.

L. Dong, F. Wei, M. Zhou, and K. Xu. Question Answering over Freebase with Multi-Column Convolutional Neural Networks. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, 2015.

P. Erdos and A. Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

O. Etzioni, M. Banko, S. Soderland, and D. Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, 2008.

J. S. B. Evans, S. E. Newstead, and R. M. Byrne. *Human reasoning: The psychology of deduction*. Psychology Press, 1993.

A. Fader, L. Zettlemoyer, and O. Etzioni. Open question answering over curated and extracted knowledge bases. In *Proceedings of SIGKDD*, pages 1156–1165, 2014.

D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, et al. Building Watson: An overview of the DeepQA project. *AI Magazine*, 31(3):59–79, 2010.

R. Fikes and T. Kehler. The role of frame-based representation in reasoning. *Communications of the ACM*, 28(9):904–920, 1985.

C. J. Fillmore. Scenes-and-frames semantics. *Linguistic structures processing*, 59:55–88, 1977.

J. L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

M. Forbes and Y. Choi. Verb Physics: Relative Physical Knowledge of Actions and Objects. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 266–276, 2017.

R. M. French. The Turing Test: the first 50 years. *Trends in cognitive sciences*, 4(3): 115–122, 2000.

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

K.-I. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2(3):183–192, 1989.

E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJcAI*, volume 7, pages 1606–1611, 2007.

M. Gardner, P. Talukdar, and T. Mitchell. Combining vector space embeddings with symbolic logical inference over open-domain text. In *AAAI spring symposium*, 2015.

M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. Liu, M. Peters, M. Schmitz, and L. Zettlemoyer. AllenNLP: A Deep Semantic Natural Language Processing Platform. 2018.

E. N. Gilbert. Random graphs. *The Annals of Mathematical Statistics*, 30(4):1141–1144, 1959.

D. Gildea and D. Jurafsky. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288, 2002.

D. Goldwasser and D. Roth. Learning from natural instructions. *Machine Learning*, 94(2): 205–232, 2 2014. URL `http://cogcomp.org/papers/GoldwasserRo14.pdf`.

M. Granroth-Wilding and S. Clark. What happens next? event prediction using a compositional neural network model. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2727–2733. AAAI Press, 2016.

D. Gunning, V. Chaudhri, P. Clark, K. Barker, J. Chaw, and M. Greaves. Project Halo Update - Progress Toward Digital Aristotle. *AI Magazine*, 31(3), 2010.

S. Gururangan, S. Swayamdipta, O. Levy, R. Schwartz, S. Bowman, and N. A. Smith. Annotation Artifacts in Natural Language Inference Data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 107–112, 2018.

S. Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3): 335–346, 1990.

S. Harnad. The Turing Test is not a trick: Turing indistinguishability is a scientific criterion. *ACM SIGART Bulletin*, 3(4):9–10, 1992.

K. M. Hermann, T. Kociský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching Machines to Read and Comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, pages 1693–1701, 2015.

J. Hernandez-Orallo. Beyond the Turing test. *Journal of Logic, Language and Information*, 9(4):447–466, 2000.

L. Hirschman, M. Light, E. Breck, and J. D. Burger. Deep Read: A Reading Comprehension System. In *27th Annual Meeting of the Association for Computational Linguistics, ACL 1999*, 1999. URL `http://www.aclweb.org/anthology/P99-1042`.

J. R. Hobbs, M. E. Stickel, P. A. Martin, and D. Edwards. Interpretation as Abduction. *Artif. Intell.*, 63:69–142, 1988.

J. R. Hobbs, M. E. Stickel, D. E. Appelt, and P. Martin. Interpretation as abduction. *Artificial intelligence*, 63(1-2):69–142, 1993.

J. H. Holland, K. J. Holyoak, R. E. Nisbett, and P. R. Thagard. *Induction: Processes of inference, learning, and discovery*. MIT press, 1989.

C. Hori and F. Sadaoki. Speech summarization: an approach through word extraction and a method for evaluation. *IEICE TRANSACTIONS on Information and Systems*, 87(1): 15–25, 2004.

M. J. Hosseini, H. Hajishirzi, O. Etzioni, and N. Kushman. Learning to Solve Arithmetic Word Problems with Verb Categorization. In *2014EMNLP*, pages 523–533, 2014.

D. Howell. *Statistical methods for psychology*. Cengage Learning, 2012.

M. Hu, Y. Peng, Z. Huang, X. Qiu, F. Wei, and M. Zhou. Reinforced mnemonic reader for machine reading comprehension. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4099–4106. AAAI Press, 2018.

N. Ide and K. Suderman. Integrating Linguistic Resources: The American National Corpus Model. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation, LREC 2006*, pages 621–624, 2006. URL `http://www.lrec-conf.org/proceedings/lrec2006/pdf/560_pdf.pdf`.

N. Ide, C. F. Baker, C. Fellbaum, C. J. Fillmore, and R. J. Passonneau. MASC: the Manually Annotated Sub-Corpus of American English. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008*, 2008. URL `http://www.lrec-conf.org/proceedings/lrec2008/summaries/617.html`.

P. Jansen, N. Balasubramanian, M. Surdeanu, and P. Clark. What's in an Explanation? Characterizing Knowledge and Inference Requirements for Elementary Science Exams. In *Proc. the International Conference on Computational Linguistics (COLING)*, pages 2956–2965, 2016.

P. Jansen, R. Sharp, M. Surdeanu, and P. Clark. Framing QA as Building and Ranking Intersentence Answer Justifications. *Computational Linguistics*, 2017.

P. A. Jansen. A Study of Automatically Acquiring Explanatory Inference Patterns from Corpora of Explanations: Lessons from Elementary Science Exams. In *AKBC*, 2016.

P. A. Jansen, E. Wainwright, S. Marmorstein, and C. T. Morrison. WorldTree: A Corpus of Explanation Graphs for Elementary Science Questions supporting Multi-Hop Inference. *CoRR*, abs/1802.03052, 2018.

M. E. Janzen and K. J. Vicente. Attention allocation within the abstraction hierarchy. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 41, pages 274–278. SAGE Publications, 1997.

P. Jia and P. Liang. Adversarial Examples for Evaluating Reading Comprehension Systems. *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2017.

T. Joachims. Text categorization with support vector machines: Learning with many relevant features. *Machine learning: ECML-98*, pages 137–142, 1998.

A. Johnson and R. W. Proctor. *Attention: Theory and practice*. Sage Publications, 2004.

P. N. Johnson-Laird. Mental models in cognitive science. *Cognitive science*, 4(1):71–115, 1980.

M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Volume 1: Long Papers*, pages 1601–1611, 2017. doi: 10.18653/v1/P17-1147. URL `https://doi.org/10.18653/v1/P17-1147`.

M. Kaisser and B. Webber. Question answering based on semantic roles. In *Proceedings of the workshop on deep linguistic processing*, pages 41–48, 2007.

R. M. Kaplan, J. Bresnan, et al. Lexical-functional grammar: A formal system for grammatical representation. In *The Mental Representation of Grammatical Relations*. The MIT Press, 1982.

R. J. Kate and R. J. Mooney. Probabilistic Abduction using Markov Logic Networks. In *In: IJCAI-09 Workshop on Plan, Activity, and Intent Recognition*, 2009.

D. Kaushik and Z. C. Lipton. How Much Reading Does Reading Comprehension Require? A Critical Investigation of Popular Benchmarks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5010–5015, 2018.

A. Kembhavi, M. Seo, D. Schwenk, J. Choi, A. Farhadi, and H. Hajishirzi. Are You Smarter Than A Sixth Grader? Textbook Question Answering for Multimodal Machine Comprehension. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

D. Khashabi, T. Khot, A. Sabharwal, P. Clark, O. Etzioni, and D. Roth. Question answering via integer programming over semi-structured knowledge. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2016. URL `http://cogcomp.org/papers/KKSCER16.pdf`.

D. Khashabi, T. Khot, A. Sabharwal, and D. Roth. Learning what is essential in questions. In *The Conference on Computational Natural Language Learning (Proc. of the Conference on Computational Natural Language Learning (CoNLL))*, 2017. URL `http://cogcomp.org/papers/2017_conll_essential_terms.pdf`.

D. Khashabi, S. Chaturvedi, M. Roth, S. Upadhyay, and D. Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018a. URL `http://www.aclweb.org/anthology/N18-1023`.

D. Khashabi, T. Khot, A. Sabharwal, and D. Roth. Question answering as global reasoning over semantic abstractions. In *Proceedings of The Conference on Artificial In-*

telligence (Proc. of the Conference on Artificial Intelligence (AAAI)), 2018b. URL `http://cogcomp.org/papers/2018_aaai_semanticilp.pdf`.

D. Khashabi, M. Sammons, B. Zhou, T. Redman, C. Christodoulopoulos, V. Srikumar, N. Rizzolo, L. Ratinov, G. Luo, Q. Do, C.-T. Tsai, S. Roy, S. Mayhew, Z. Feng, J. Wieting, X. Yu, Y. Song, S. Gupta, S. Upadhyay, N. Arivazhagan, Q. Ning, S. Ling, and D. Roth. Cogcompnlp: Your swiss army knife for nlp. In *11th Language Resources and Evaluation Conference*, 2018c. URL `http://cogcomp.org/papers/2018_lrec_cogcompnlp.pdf`.

D. Khashabi, E. S. Azer, T. Khot, A. Sabharwal, and D. Roth. On the capabilities and limitations of reasoning for natural language understanding, 2019. under review.

T. Khot, N. Balasubramanian, E. Gribkoff, A. Sabharwal, P. Clark, and O. Etzioni. Exploring Markov Logic Networks for Question Answering. In *2015EMNLP*, Lisbon, Portugal, 2015.

T. Khot, A. Sabharwal, and P. Clark. Answering Complex Questions Using Open Information Extraction. *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, 2017.

P. Kingsbury and M. Palmer. From TreeBank to PropBank. In *LREC*, pages 1989–1993, 2002.

G. S. Kirk, J. E. Raven, and M. Schofield. *The presocratic philosophers: A critical history with a selcetion of texts*. Cambridge University Press, 1983.

K. Knight and D. Marcu. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107, 2002.

J. Ko, E. Nyberg, and L. Si. A probabilistic graphical model for joint answer ranking in question answering. In *Proceedings of SIGIR*, pages 343–350, 2007.

P. Kordjamshidi, D. Roth, and H. Wu. Saul: Towards Declarative Learning Based Programming. In *Proc. 24th Int. Joint Conf. on Artificial Intelligence IJCAI*, 7 2015.

Z. Kozareva and E. Hovy. Learning temporal information for states and events. In *Fifth International Conference on Semantic Computing*, pages 424–429. IEEE, 2011.

J. Krishnamurthy, O. Tafjord, and A. Kembhavi. Semantic parsing to probabilistic programs for situated question answering. *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2016.

C. C. T. Kwok, O. Etzioni, and D. S. Weld. Scaling question answering to the Web. In *The International World Wide Web Conference*, 2001.

G. Lai, Q. Xie, H. Liu, Y. Yang, and E. H. Hovy. RACE: Large-scale ReAding Comprehension Dataset From Examinations. In *Proceedings of the 2017 Conference on Empirical*

Methods in Natural Language Processing, EMNLP 2017, pages 785–794, 2017. URL https://aclanthology.info/papers/D17-1082/d17-1082.

H. Lee, Y. Peirsman, A. Chang, N. Chambers, M. Surdeanu, and D. Jurafsky. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *CONLL Shared Task*, pages 28–34, 2011.

H. Lee, A. Chang, Y. Peirsman, N. Chambers, M. Surdeanu, and D. Jurafsky. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4):885–916, 2013.

K. Lee, Y. Artzi, J. Dodge, and L. Zettlemoyer. Context-dependent semantic parsing for time expressions. In *ACL (1)*, pages 1437–1447, 2014.

A. Leeuwenberg and M.-F. Moens. Temporal Information Extraction by Predicting Relative Time-lines. *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2018.

W. G. Lehnert. *The Process of Question Answering.* PhD thesis, Yale University, 1977.

D. B. Lenat. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

O. Levy and Y. Goldberg. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the eighteenth conference on computational natural language learning*, pages 171–180, 2014.

F. Li, X. Zhang, J. Yuan, and X. Zhu. Classifying What-Type Questions by Head Noun Tagging. In *Proceedings 22nd International Conference on Computational Linguistics (COLING)*, 2007.

X. Li and D. Roth. Learning Question Classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

Y. Li, L. Xu, F. Tian, L. Jiang, X. Zhong, and E. Chen. Word Embedding Revisited: A New Representation Learning and Explicit Matrix Factorization Perspective. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3650–3656, 2015.

Z. Li, X. Ding, and T. Liu. Constructing Narrative Event Evolutionary Graph for Script Event Prediction. *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

X. V. Lin, R. Socher, and C. Xiong. Multi-Hop Knowledge Graph Reasoning with Reward Shaping. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2018.

H. Liu and P. Singh. ConceptNeta practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226, 2004.

X. Liu, Y. Shen, K. Duh, and J. Gao. Stochastic answer networks for machine reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1694–1704, 2018.

A. A. Mahabal, D. Roth, and S. Mittal. Robust handling of polysemy via sparse representations. In **SEM*, 2018. URL `http://cogcomp.org/papers/MahabalRoMi18.pdf`.

M.-C. D. Marneffe, B. MacCartney, C. D. Manning, et al. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454. Genoa, 2006.

A. McCallum, A. Neelakantan, R. Das, and D. Belanger. Chains of Reasoning over Entities, Relations, and Text using Recurrent Neural Networks. In *EACL*, pages 132–141, 2017.

J. McCarthy. *Programs with common sense*. Defense Technical Information Center, 1963.

J. McCarthy. An example for natural language understanding and the AI problems it raises. *Formalizing Common Sense: Papers by John McCarthy*, 355, 1976.

J. McCarthy and M. I. Levin. *LISP 1.5 programmer's manual*. MIT press, 1965.

J. McCarthy and V. Lifschitz. *Formalizing common sense: papers*, volume 5. Intellect Books, 1990.

J. F. McCarthy. Using decision trees for coreference resolution. In *Proc. 14th International Joint Conf. on Artificial Intelligence (IJCAI), Quebec, Canada, Aug. 1995*, 1995.

E. Merkhofer, J. Henderson, D. Bloom, L. Strickhart, and G. Zarrella. MITRE at SemEval-2018 Task 11: Commonsense Reasoning without Commonsense Knowledge. In *Proceedings of the International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, USA, 2018.

A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. The NomBank project: An interim report. In *HLT-NAACL 2004 workshop: Frontiers in corpus annotation*, volume 24, page 31, 2004.

R. Mihalcea and A. Csomai. Wikify!: linking documents to encyclopedic knowledge. In *CIKM*, pages 233–242, 2007.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

S. Milgram. Six degrees of separation. *Psychology Today*, 2:60–64, 1967.

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

S. Min, M. J. Seo, and H. Hajishirzi. Question Answering through Transfer Learning from Large Fine-grained Supervision Data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Volume 2: Short Papers*, pages 510–517, 2017. URL `https://doi.org/10.18653/v1/P17-2081`.

M. Minsky. A Framework for Representing Knowledge. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1974.

M. Minsky. *Society of mind*. Simon and Schuster, 1988.

M. Minsky and S. Papert. Perceptron: an introduction to computational geometry. *The MIT Press, Cambridge, expanded edition*, 19:88, 1969.

T. M. Mitchell, J. Betteridge, A. Carlson, E. Hruschka, and R. Wang. Populating the semantic web by macro-reading internet text. In *International Semantic Web Conference*, pages 998–1002. Springer, 2009.

D. Moldovan, M. Paşca, S. Harabagiu, and M. Surdeanu. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems (TOIS)*, 21(2):133–154, 2003.

P. Moreda, H. Llorens, E. S. Boró, and M. Palomar. Combining semantic information in question answering systems. *Inf. Process. Manage.*, 47:870–885, 2011.

K. Narasimhan and R. Barzilay. Machine Comprehension with Discourse Relations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, Volume 1: Long Papers*, pages 1253–1262, 2015. URL `http://aclweb.org/anthology/P/P15/P15-1121.pdf`.

B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234, 1995.

T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *CoRR*, abs/1611.09268, 2016. URL `http://arxiv.org/abs/1611.09268`.

J. Ni, C. Zhu, W. Chen, and J. McAuley. Learning to attend on essential terms: An enhanced retriever-reader model for scientific question answering. *arXiv preprint arXiv:1808.09492*, 2018.

Q. Ning, H. Wu, H. Peng, and D. Roth. Improving Temporal Relation Extraction with a Globally Acquired Statistical Resource. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 841–851,

New Orleans, Louisiana, 6 2018a. Association for Computational Linguistics. URL http://cogcomp.org/papers/NingWuPeRo18.pdf.

Q. Ning, B. Zhou, Z. Feng, H. Peng, and D. Roth. CogCompTime: A Tool for Understanding Time in Natural Language. In *EMNLP (Demo Track)*, Brussels, Belgium, 11 2018b. Association for Computational Linguistics. URL http://cogcomp.org/papers/NZFPR18.pdf.

G. Novak. Representations of Knowledge in a Program for Solving Physics Problems. In *IJCAI-77*, 1977.

S. Ostermann, M. Roth, A. Modi, S. Thater, and M. Pinkal. SemEval-2018 Task 11: Machine Comprehension using Commonsense Knowledge. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 747–757, 2018.

M. Palmer, D. Gildea, and P. Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106, 2005.

A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit. A Decomposable Attention Model for Natural Language Inference. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2016.

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

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988. ISBN 1558604790.

C. S. Peirce. A Theory of Probable Inference. In *Studies in Logic by Members of the Johns Hopkins University*, pages 126–181. Little, Brown, and Company, 1883.

J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237, 2018.

L. A. Pizzato and D. Mollá. Indexing on semantic roles for question answering. In *2nd workshop on Information Retrieval for Question Answering*, pages 74–81, 2008.

A. Poliak, J. Naradowsky, A. Haldar, R. Rudinger, and B. V. Durme. Hypothesis Only Baselines in Natural Language Inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, 2018.

D. Poole. A methodology for using a default and abductive reasoning system. *Int. J. Intell. Syst.*, 5:521–548, 1990.

V. Punyakanok and D. Roth. The use of classifiers in sequential inference. In *Proc. of the Conference on Neural Information Processing Systems (NIPS)*, pages 995–1001. MIT Press, 2001. URL `http://cogcomp.org/papers/nips01.pdf`.

V. Punyakanok, D. Roth, and W. Yih. Mapping Dependencies Trees: An Application to Question Answering. *AIM*, 1 2004. URL `http://cogcomp.org/papers/PunyakanokRoYi04a.pdf`.

V. Punyakanok, D. Roth, and W. tau Yih. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 2008.

M. R. Quillan. Semantic memory. Technical report, BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA, 1966.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2016.

P. Rajpurkar, R. Jia, and P. Liang. Know What You Don't Know: Unanswerable Questions for SQuAD. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, 2018.

H. Rashkin, M. Sap, E. Allaway, N. A. Smith, and Y. Choi. Event2Mind: Commonsense Inference on Events, Intents, and Reactions. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 463–473, 2018.

J. Rasmussen. The role of hierarchical knowledge representation in decisionmaking and system management. *Systems, Man and Cybernetics, IEEE Transactions on*, pages 234–243, 1985.

L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*, 6 2009. URL `http://cogcomp.org/papers/RatinovRo09.pdf`.

L. Ratinov, D. Roth, D. Downey, and M. Anderson. Local and global algorithms for disambiguation to wikipedia. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2011. URL `http://cogcomp.org/papers/RRDA11.pdf`.

S. Reddy, O. Täckström, S. Petrov, M. Steedman, and M. Lapata. Universal Semantic Parsing. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 89–101, 2017.

T. Redman, M. Sammons, and D. Roth. Illinois Named Entity Recognizer: Addendum to Ratinov and Roth '09 reporting improved results, 2016. URL `http://cogcomp.org/papers/ner-addendum.pdf`. Tech Report.

M. Richardson and P. Domingos. Markov Logic Networks. *Machine learning*, 62(1–2): 107–136, 2006.

M. Richardson, C. J. C. Burges, and E. Renshaw. MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013*, pages 193–203, 2013. URL `http://aclweb.org/anthology/D/D13/D13-1020.pdf`.

F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

D. Roth and W. Yih. A linear programming formulation for global inference in natural language tasks. In H. T. Ng and E. Riloff, editors, *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*, pages 1–8. Association for Computational Linguistics, 2004. URL `http://cogcomp.org/papers/RothYi04.pdf`.

D. Roth and D. Zelenko. Part of speech tagging using a network of linear separators. In *ACL-COLING*, 1998.

M. Roth and M. Lapata. Neural semantic role labeling with dependency path embeddings. *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, 2016.

S. Roy, T. Vieira, and D. Roth. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics (TACL)*, 3, 2015. URL `http://cogcomp.org/papers/RoyViRo15.pdf`.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5, 1988.

M. Sammons, C. Christodoulopoulos, P. Kordjamshidi, D. Khashabi, V. Srikumar, P. Vijayakumar, M. Bokhari, X. Wu, and D. Roth. Edison: Feature extraction for nlp, simplified. In N. C. C. Chair), K. Choukri, T. Declerck, M. Grobelnik, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, and S. Piperidis, editors, *Proc. of the International Conference on Language Resources and Evaluation (LREC)*. European Language Resources Association (ELRA), 2016. URL `http://cogcomp.org/papers/SCKKSVBWR16.pdf`.

R. C. Schank. Conceptual dependency: A theory of natural language understanding. *Cognitive psychology*, 3(4):552–631, 1972.

R. C. Schank and R. P. Abelson. Scripts, plans, and knowledge. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 151–157, 1975.

B. Selman and H. J. Levesque. Abductive and Default Reasoning: A Computational Core. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1990.

M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. Bidirectional attention flow for machine comprehension. *ICLR*, 2016.

D. Shen and M. Lapata. Using Semantic Roles to Improve Question Answering. In *EMNLP-CoNLL*, pages 12–21, 2007.

R. Socher, D. Chen, C. D. Manning, and A. Y. Ng. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*, 2013.

V. Srikumar and D. Roth. A Joint Model for Extended Semantic Role Labeling. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, Edinburgh, Scotland, 2011. URL `http://cogcomp.org/papers/SrikumarRo11.pdf`.

V. Srikumar and D. Roth. Modeling semantic relations expressed by prepositions. 1:231–242, 2013. URL `http://cogcomp.org/papers/SrikumarRo13.pdf`.

M. Steedman and J. Baldridge. Combinatory categorial grammar. *Non-Transformational Syntax: Formal and explicit models of grammar*, pages 181–224, 2011.

A. Stern, R. Stern, I. Dagan, and A. Felner. Efficient search for transformation-based inference. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 283–291, 2012.

M. Steup. Epistemology. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. `http://plato.stanford.edu/archives/spr2014/entries/epistemology/`, spring 2014 edition, 2014.

K. Sun, D. Yu, D. Yu, and C. Cardie. Improving machine reading comprehension with general reading strategies. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, 2019.

W. t. Yih, X. He, and C. Meek. Semantic Parsing for Single-Relation Question Answering. In *52ndACL*, pages 643–648. Citeseer, 2014.

M. Taddeo and L. Floridi. Solving the symbol grounding problem: a critical review of fifteen years of research. *Journal of Experimental & Theoretical Artificial Intelligence*, 17 (4):419–445, 2005.

P. P. Talukdar, M. Jacob, M. S. Mehmood, K. Crammer, Z. G. Ives, F. Pereira, and S. Guha. Learning to create data-integrating queries. *Proceedings of the VLDB Endowment*, 1(1): 785–796, 2008.

P. P. Talukdar, Z. G. Ives, and F. Pereira. Automatically incorporating new sources in keyword search-based data integration. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 387–398. ACM, 2010.

N. Tandon, B. Dalvi, J. Grus, W. tau Yih, A. Bosselut, and P. Clark. Reasoning about Actions and State Changes by Injecting Commonsense Knowledge. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 57–66, 2018.

K. Toutanova and D. Chen. Observed versus latent features for knowledge base and text inference. In *CVSC workshop*, 2015.

H. Trivedi, H. Kwon, T. Khot, A. Sabharwal, and N. Balasubramanian. Entailment-based Question Answering over Multiple Sentences. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, 2019.

A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433, 1950.

P. D. Turney. Distributional semantics beyond words: Supervised learning of analogy and paraphrase. *TACL*, 1:353–366, 2013.

P. D. Turney and P. Pantel. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188, 2010.

K. Tymoshenko, D. Bonadiman, and A. Moschitti. Convolutional Neural Networks vs. Convolution Kernels: Feature Engineering for Answer Sentence Reranking. In *HLT-NAACL*, 2016.

C. Unger, L. Bühmann, J. Lehmann, A.-C. N. Ngomo, D. Gerber, and P. Cimiano. Template-based question answering over RDF data. In *Proceedings of the 21st international conference on World Wide Web*, pages 639–648. ACM, 2012.

A. Vempala, E. Blanco, and A. Palmer. Determining Event Durations: Models and Error Analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 164–168, 2018.

B. Wang, K. Liu, and J. Zhao. Inner Attention based Recurrent Neural Networks for Answer Selection. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, 2016.

C. Wang, N. Xue, S. Pradhan, and S. Pradhan. A Transition-based Algorithm for AMR Parsing. In *HLT-NAACL*, pages 366–375, 2015.

H. Wang, D. Yu, K. Sun, J. Chen, D. Yu, D. Roth, and D. McAllester. Evidence Sentence Extraction for Machine Reading Comprehension. *arXiv preprint arXiv:1902.08852*, 2019.

W. Wang, M. Yan, and C. Wu. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1705–1714, 2018.

D. J. Watts and S. H. Strogatz. Collective dynamics of small-worldnetworks. *nature*, 393 (6684):440, 1998.

J. Wieting, M. Bansal, K. Gimpel, K. Livescu, and D. Roth. From Paraphrase Database to Compositional Paraphrase Model and Back. *TACL*, 3:345–358, 2015.

J. Williams. Extracting fine-grained durations for verbs from Twitter. In *Proceedings of ACL 2012 Student Research Workshop*, pages 49–54. Association for Computational Linguistics, 2012.

T. Winograd. Understanding natural language. *Cognitive psychology*, 3(1):1–191, 1972.

W. A. Woods. Progress in natural language understanding: an application to lunar geology. In *Proceedings of the June 4-8, 1973, national computer conference and exposition*, pages 441–450. ACM, 1973.

S. Yang, L. Zou, Z. Wang, J. Yan, and J.-R. Wen. Efficiently Answering Technical Questions-A Knowledge Graph Approach. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 3111–3118, 2017.

Y. Yang, W. Yih, and C. Meek. WikiQA: A Challenge Dataset for Open-Domain Question Answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, pages 2013–2018, 2015. URL `http://aclweb.org/anthology/D/D15/D15-1237.pdf`.

Y. Yang, L. Birnbaum, J.-P. Wang, and D. Downey. Extracting Commonsense Properties from Embeddings with Limited Human Guidance. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 644–649, 2018.

X. Yao and B. V. Durme. Information extraction over structured data: Question answering with Freebase. In *52ndACL*, 2014.

W. Yin, S. Ebert, and H. Schütze. Attention-based convolutional neural network for machine comprehension. In *NAACL HCQA Workshop*, 2016.

L. A. Zadeh. The concept of a linguistic variable and its application to approximate reasoningI. *Information sciences*, 8(3):199–249, 1975.

L. A. Zadeh. PRUFa meaning representation language for natural languages. *International Journal of man-machine studies*, 10(4):395–460, 1978.

R. Zellers, Y. Bisk, R. Schwartz, and Y. Choi. SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, 2018.

L. S. Zettlemoyer and M. Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *UAI*, 2005.

S. Zhang, R. Rudinger, K. Duh, and B. V. Durme. Ordinal Common-sense Inference. *Transactions of the Association of Computational Linguistics*, 5(1):379–395, 2017.

B. Zhou, D. Khashabi, Q. Ning, and D. Roth. "going on a vacation" takes longer than

"going for a walk": A study of temporal commonsense understanding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

L. Zou, R. Huang, H. Wang, J. X. Yu, W. He, and D. Zhao. Natural language question answering over RDF: a graph data driven approach. In *SIGMOD*, pages 313–324, 2014.