The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Question Answering as Global Reasoning over Semantic Abstractions

Daniel Khashabi[*]
University of Pennsylvania
danielkh@cis.upenn.edu

Tushar Khot **Ashish Sabharwal**
Allen Institute for Artificial Intelligence (AI2)
tushark,ashishs@allenai.org

Dan Roth[*]
University of Pennsylvania
danroth@cis.upenn.edu

Abstract

We propose a novel method for exploiting the semantic structure of text to answer multiple-choice questions. 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, coreference resolvers, and dependency parsers. 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%.

Introduction

Question answering (QA) is a fundamental challenge in AI, particularly in natural language processing (NLP) and reasoning. 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. This setting applies naturally to reading comprehension questions, as well as to open format multiple-choice questions for which information retrieval (IR) techniques are able to obtain relevant text from a corpus.

Several state-of-the-art QA systems excel at locating the correct answer in P based on proximity to question words, the distributional semantics of the text, answer type, etc (Clark et al. 2016), however, often struggle with questions that require some form of reasoning or appeal to a more subtle understanding of the supporting text or the question. 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.

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*.

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 1 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 exam-

[*] Part of the work was done when the first and last authors were affiliated with the University of Illinois, Urbana-Champaign.

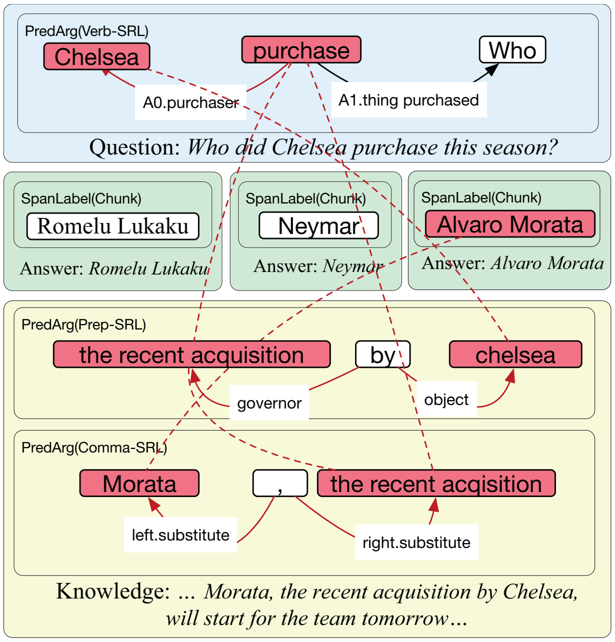

Figure 1: Depiction of SEMANTICILP reasoning for the example paragraph given in the text. Semantic abstractions of the question, answers, knowledge snippet are shown in different colored boxes (blue, green, and yellow, resp.). Red nodes and edges are the elements that are aligned (used) for supporting the correct answer. There are many other unaligned (unused) annotations associated with each piece of text that are omitted for clarity.

ple, 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[1] in a domain independent fashion, as illustrated for an example in Figure 1. 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 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.

This formalism may be viewed as a generalization of systems that reason over tables of knowledge (Cohen 2000; Khashabi et al. 2016): 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, Sabharwal, and Clark 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,[2] 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 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 gen-

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

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

eralizes 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.

Related Work

There are numerous QA systems operating over large knowledge-bases. Our work, however, is most closely related to systems that work either directly on the input text or on a semantic representation derived on-the-fly from it. In the former category are IR and statistical correlation based methods with surprisingly strong performance (Clark et al. 2016), as well as a number of recent neural architectures such as BiDAF (Seo et al. 2016), Decomposable Attention Model (Parikh et al. 2016), etc. In the latter category are approaches that perform *structured reasoning* over some abstraction of text. For instance, Khashabi et al. (2016) perform reasoning on the knowledge tables constructed using semi-automatical methods, Khot, Sabharwal, and Clark (2017) use Open IE subject-verb-object relations (Etzioni et al. 2008), Banarescu et al. (2013) use AMR annotators (Wang et al. 2015), and Krishnamurthy, Tafjord, and Kembhavi (2016) use a semantic parser (Zettlemoyer and Collins 2005) to answer a given question. Our work differs in that we use of a wide variety of semantic abstractions simultaneously, and perform joint reasoning over them to identify which abstractions are relevant for a given question and how best to combine information from them.

Our formalism can be seen as an extension of some of the prior work on structured reasoning over semi-structured text. For instance, in our formalism, each table used by Khashabi et al. (2016) 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, Sabharwal, and Clark 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, Roth, and Yih 2008) using linguistic resources such as FrameNet (Baker, Fillmore, and Lowe 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, Gildea, and Kingsbury 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, Roth, and Yih 2004). These approaches often define inference rules, which can generate new trees starting from a base tree. Bar-Haim, Dagan, and Berant (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.

Another aspect of our approach is the graphical representation of knowledge and the reasoning over it. We model the QA problem as a search in the space of potential support graphs. Search-based reasoning systems have been successful in several NLP areas (Roth and Yih 2004; Chang et al. 2010; Berant, Dagan, and Goldberger 2010; Srikumar and Roth 2011; Goldwasser and Roth 2011; Schüller 2014; Fei et al. 2015). Compared to previous works, we use a larger set of annotators to account for diverse linguistic phenomena in questions and to handle errors of individual annotators.

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.

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 itself, embellished with various SemanticGraphs attached to it, as depicted in Figure 2.

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 predi-

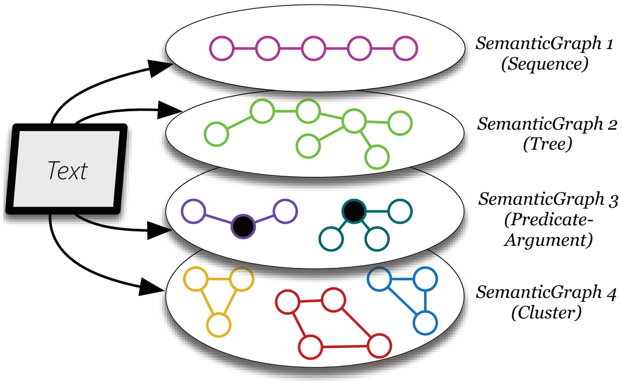

Figure 2: Knowledge Representation used in our formulation. Raw text is associated with a collection of SemanticGraphs, which convey certain information about the text. There are implicit similarity edges among the nodes of the connected components of the graphs, and from nodes to the corresponding raw-text spans.

cates 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.

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.[3]

- Sequence or labels for sequence of tokens; for example Lemma and POS (Roth and Zelenko 1998).
- Span which can contains labels for spans of text; we instantiated Shallow-Parse (Punyakanok and Roth 2001), Quantities (Roy, Vieira, and Roth 2015), NER (Ratinov and Roth 2009; Redman, Sammons, and Roth 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, Roth, and Yih 2008; Roth and Lapata 2016), Prep-SRL (Srikumar and Roth 2013), Comma-SRL (Arivazhagan, Christodoulopoulos, and Roth 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.

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 $\mathcal{K}(P)$.[4] 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 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 1. 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:[5]

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

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 2. 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:

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

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

[5] 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 1: 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:[6]

   $$G \cap \mathcal{K}(Q) \neq \emptyset, \quad G \cap \mathcal{K}(P) \neq \emptyset, \quad \exists!\, i : G \cap \mathcal{K}(A_i) \neq \emptyset$$
3. G satisfies structural properties per each connected component, as summarized in Table 1.

Definition 2 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 2, 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:

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

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. (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 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:

$$G^* = \arg\max_{G \in \mathcal{G}} \text{score}(G) \quad (2)$$

ILP Formulation

Our QA system, SemanticILP, models the above support graph search of Eq. (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 1 are implemented as weighted linear constraints using the variables defined for the nodes and edges.

- 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 2: The set of preferences functions in the objective.

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).

The final optimization is done on Eq. (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. (1) contains a set of preferences $\mathcal{C}$, summarized in Table 2, 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.

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

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

best system for each domain. These datasets and systems are described next, followed by results.

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, Sabharwal, and Clark 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).[7]

For the second domain, we use the ProcessBank[8] 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.[9]. 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.

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 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 3. For all of these annotator combinations, we let the representation of the answers be $\mathcal{K}(A) = \{\text{Shallow-Parse}, \text{Tokens}\}$. 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. (2).

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

Table 3: The semantic annotator combinations used in our implementation of SemanticILP.

Each simplified solver associated with an annotator combination in Table 3 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, Seo, and Hajishirzi 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, pretrained 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[10], 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, Sabharwal, and Clark (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

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

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

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

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

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 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.

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 4, 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.

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 4: 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*.

Biology Exam. The results on the PROCESSBANK dataset are summarized in Table 5. 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 baselines, makes it limited to this dataset, which is also why we don't include it in Table 4.

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 5: Biology test scores as a percentage. SEMANTICILP outperforms various baselines on the PROCESSBANK dataset and roughly matches the specialized best method.

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 6 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 6: SemanticILP statistics averaged across questions, as compared to TableILP and TupleInf statistics.

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 7. 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 7: 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 3.

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 3 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 correctly (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 4 and 5, 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 3). 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.

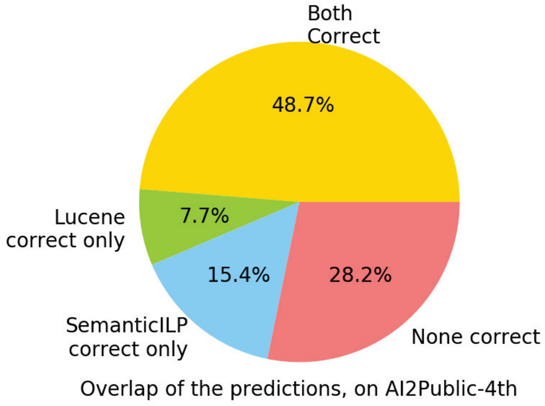

Figure 3: Overlap of the predictions of SemanticILP and IR on 50 randomly-chosen questions from AI2Public 4th.

The results are reported in Table 8. 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.

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

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 4 (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.

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.

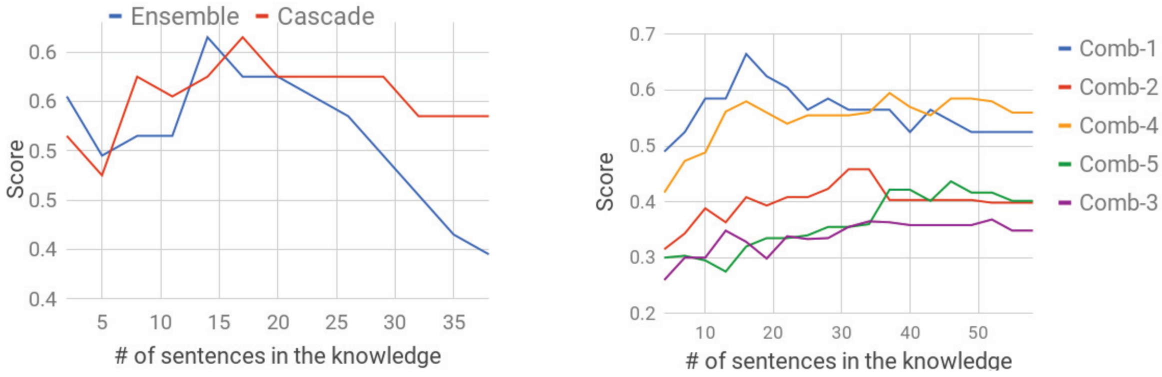

Figure 4: Performance change for varying knowledge length.

Figure 4 (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.

Conclusion

Question answering is challenging in the presence of linguistic richness and diversity, especially when arriving at the correct answer requires some level of reasoning. Departing from the currently popular paradigm of generating a very large dataset and learning "everything" from it in an end-to-end fashion, we argue—and demonstrate via our QA system—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. Bringing these ideas to practice, our system, SEMANTICILP, achieves state-of-the-art performance on two domains with very different characteristics, outperforming both traditional and neural models.

Acknowledgments

The authors would like to thank Peter Clark, Bhavana Dalvi, Minjoon Seo, Oyvind Tafjord, Niket Tandon, and Luke Zettlemoyer for valuable discussions and help.

This work is supported by a gift from AI2 and by contract FA8750-13-2-0008 with the US Defense Advanced Research Projects Agency (DARPA). The views expressed are those of the authors and do not reflect the official policy or position of the U.S. Government.

References

Achterberg, T. 2009. SCIP: solving constraint integer programs. *Math. Prog. Computation* 1(1):1–41.

Arivazhagan, N.; Christodoulopoulos, C.; and Roth, D. 2016. Labeling the semantic roles of commas. In *AAAI*, 2885–2891.

Baker, C. F.; Fillmore, C. J.; and Lowe, J. B. 1998. The berkeley framenet project. In *ACL*, 86–90.

Banarescu, L.; Bonial, C.; Cai, S.; Georgescu, M.; Griffitt, K.; Hermjakob, U.; Knight, K.; Palmer, M.; and Schneider, N. 2013. Abstract meaning representation for sembanking. In *Linguistic Annotation Workshop and Interoperability with Discourse*.

Banko, M.; Cafarella, M. J.; Soderland, S.; Broadhead, M.; and Etzioni, O. 2007. Open information extraction from the web. In *IJCAI*.

Bar-Haim, R.; Dagan, I.; and Berant, J. 2015. Knowledge-based textual inference via parse-tree transformations. *J. Artif. Intell. Res.(JAIR)* 54:1–57.

Berant, J.; Srikumar, V.; Chen, P.-C.; Vander Linden, A.; Harding, B.; Huang, B.; Clark, P.; and Manning, C. D. 2014. Modeling biological processes for reading comprehension. In *EMNLP*.

Berant, J.; Dagan, I.; and Goldberger, J. 2010. Global learning of focused entailment graphs. In *ACL*, 1220–1229.

Chang, M.-W.; Goldwasser, D.; Roth, D.; and Srikumar, V. 2010. Discriminative learning over constrained latent representations. In *NAACL*, 429–437.

Chang, K.-W.; Upadhyay, S.; Chang, M.-W.; Srikumar, V.; and Roth, D. 2015. Illinois-SL: A java library for structured prediction. *arXiv preprint arXiv:1509.07179*.

Clark, P.; Etzioni, O.; Khot, T.; Sabharwal, A.; Tafjord, O.; Turney, P.; and Khashabi, D. 2016. Combining retrieval, statistics, and inference to answer elementary science questions. In *AAAI*.

Cohen, W. W. 2000. Data integration using similarity joins and a word-based information representation language. *ACM Transactions on Information Systems* 18(3):288–321.

Dong, L.; Wei, F.; Zhou, M.; and Xu, K. 2015. Question answering over Freebase with multi-column convolutional neural networks. In *ACL*.

Etzioni, O.; Banko, M.; Soderland, S.; and Weld, D. S. 2008. Open information extraction from the web. *Communications of the ACM* 51(12):68–74.

Fei, Z.; Khashabi, D.; Peng, H.; Wu, H.; and Roth, D. 2015. Illinois-Profiler: knowledge schemas at scale. *Workshop on Cognitive Knowledge Acquisition and Applications (Cognitum)*.

Gildea, D., and Jurafsky, D. 2002. Automatic labeling of semantic roles. *Computational linguistics* 28(3):245–288.

Goldwasser, D., and Roth, D. 2011. Learning from natural instructions. In *IJCAI*.

Jia, P., and Liang, P. 2017. Adversarial examples for evaluating reading comprehension systems. *EMNLP*.

Kaisser, M., and Webber, B. 2007. Question answering based on semantic roles. In *Proceedings of the workshop on deep linguistic processing*, 41–48.

Kembhavi, A.; Seo, M.; Schwenk, D.; Choi, J.; Farhadi, A.; and Hajishirzi, H. 2017. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. *CVPR*.

Khashabi, D.; Khot, T.; Sabharwal, A.; Clark, P.; Etzioni, O.; and Roth, D. 2016. Question answering via integer programming over semi-structured knowledge. In *IJCAI*.

Khot, T.; Sabharwal, A.; and Clark, P. 2017. Answering complex questions using open information extraction. *ACL*.

Kingsbury, P., and Palmer, M. 2002. From treebank to propbank. In *LREC*, 1989–1993.

Krishnamurthy, J.; Tafjord, O.; and Kembhavi, A. 2016. Semantic parsing to probabilistic programs for situated question answering. *EMNLP*.

Lee, H.; Peirsman, Y.; Chang, A.; Chambers, N.; Surdeanu, M.; and Jurafsky, D. 2011. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *CONLL Shared Task*, 28–34.

Lee, H.; Chang, A.; Peirsman, Y.; Chambers, N.; Surdeanu, M.; and Jurafsky, D. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics* 39(4):885–916.

Meyers, A.; Reeves, R.; Macleod, C.; Szekely, R.; Zielinska, V.; Young, B.; and Grishman, R. 2004. The nombank project: An interim report. In *HLT-NAACL 2004 workshop: Frontiers in corpus annotation*, volume 24, 31.

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

Min, S.; Seo, M. J.; and Hajishirzi, H. 2017. Question answering through transfer learning from large fine-grained supervision data. In *ACL*.

Moreda, P.; Llorens, H.; Boró, E. S.; and Palomar, M. 2011. Combining semantic information in question answering systems. *Inf. Process. Manage.* 47:870–885.

Natarajan, B. K. 1995. Sparse approximate solutions to linear systems. *SIAM journal on computing* 24(2):227–234.

Palmer, M.; Gildea, D.; and Kingsbury, P. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics* 31(1):71–106.

Parikh, A. P.; Täckström, O.; Das, D.; and Uszkoreit, J. 2016. A decomposable attention model for natural language inference. In *EMNLP*.

Pizzato, L. A., and Mollá, D. 2008. Indexing on semantic roles for question answering. In *2nd workshop on Information Retrieval for Question Answering*, 74–81.

Punyakanok, V., and Roth, D. 2001. The use of classifiers in sequential inference. In *NIPS*, 995–1001. MIT Press.

Punyakanok, V.; Roth, D.; and Yih, W. 2004. Mapping dependencies trees: An application to question answering. *AIM*.

Punyakanok, V.; Roth, D.; and Yih, W. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics* 34(2).

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Ratinov, L., and Roth, D. 2009. Design challenges and misconceptions in named entity recognition. In *CONLL*.

Redman, T.; Sammons, M.; and Roth, D. 2016. Illinois named entity recognizer: Addendum to Ratinov and Roth '09 reporting improved results. Tech Report.

Roth, M., and Lapata, M. 2016. Neural semantic role labeling with dependency path embeddings. *ACL*.

Roth, D., and Yih, W. 2004. A linear programming formulation for global inference in natural language tasks. In *CONLL*, 1–8.

Roth, D., and Zelenko, D. 1998. Part of speech tagging using a network of linear separators. In *COLING-ACL*, 1136–1142.

Roy, S.; Vieira, T.; and Roth, D. 2015. Reasoning about quantities in natural language. *TACL* 3.

Schüller, P. 2014. Tackling winograd schemas by formalizing relevance theory in knowledge graphs. In *International Conference on the Principles of Knowledge Representation and Reasoning*.

Seo, M.; Kembhavi, A.; Farhadi, A.; and Hajishirzi, H. 2016. Bidirectional attention flow for machine comprehension. *ICLR*.

Shen, D., and Lapata, M. 2007. Using semantic roles to improve question answering. In *EMNLP-CoNLL*, 12–21.

Srikumar, V., and Roth, D. 2011. A joint model for extended semantic role labeling. In *EMNLP*.

Srikumar, V., and Roth, D. 2013. Modeling semantic relations expressed by prepositions. *TACL* 1:231–242.

Stern, A.; Stern, R.; Dagan, I.; and Felner, A. 2012. Efficient search for transformation-based inference. In *ACL*, 283–291.

Wang, C.; Xue, N.; Pradhan, S.; and Pradhan, S. 2015. A transition-based algorithm for amr parsing. In *HLT-NAACL*, 366–375.

Wieting, J.; Bansal, M.; Gimpel, K.; Livescu, K.; and Roth, D. 2015. From paraphrase database to compositional paraphrase model and back. *TACL* 3:345–358.

Zettlemoyer, L. S., and Collins, M. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *UAI*.