Question Answering via Integer Programming over Semi-Structured Knowledge

Daniel Khashabi[†], **Tushar Khot**[‡], **Ashish Sabharwal**[‡], **Peter Clark**[‡], **Oren Etzioni**[‡], **Dan Roth**[†]
[†]University of Illinois at Urbana-Champaign, IL, U.S.A.
{*khashab2,danr*}*@illinois.edu*
[‡]Allen Institute for Artificial Intelligence (AI2), Seattle, WA, U.S.A.
{*tushark,ashishs,peterc,orene*}*@allenai.org*

Abstract

Answering science questions posed in natural language is an important AI challenge. Answering such questions often requires non-trivial inference and knowledge that goes beyond factoid retrieval. Yet, most systems for this task are based on relatively shallow Information Retrieval (IR) and statistical correlation techniques operating on large unstructured corpora. We propose a structured inference system for this task, formulated 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). It also improves upon a previous ILP formulation by 17.7%. 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.

1 Introduction

Answering questions posed in natural language is a fundamental AI task, with a large number of impressive QA systems built over the years. 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 reasoning,

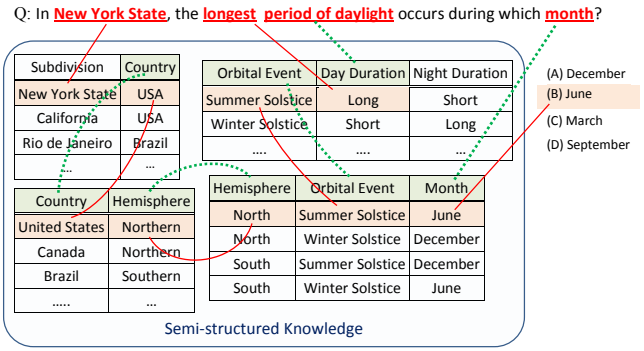

Figure 1: TableILP searches for the best support graph (chains of reasoning) connecting the question to an answer, in this case June. Constraints on the graph define what constitutes valid support and how to score it (Section 3.3).

such as chaining together multiple facts in order to arrive at a conclusion. Arguably, such reasoning is a cornerstone of human intelligence, and is a key ability evaluated by standardized science exams given to students. For example, consider a question from the NY Regents 4th Grade Science Test:

> 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 1 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 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.[1] 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 1, 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 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%.

2 Related Work

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; Yih *et al.*, 2014; Yao and Van 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.

The task of Recognizing Textual Entailment (RTE) [Dagan *et al.*, 2010; 2013] is also closely related, as QA can be cast as entailment (Does *corpus* entail *question+answer*? [Bentivogli *et al.*, 2008]). However, RTE has primarily focused on the task of *linguistic equivalence*, and has not addressed questions where some form of scientific reasoning is required. Recent work on Natural Logic [Angeli and Manning, 2014; MacCartney, 2009] has extended RTE to account for the *logical structure* within language. Our work can be seen as going one step further, to add a layer of structured reasoning on top of this; in fact, we use an RTE engine as a basic subroutine for comparing individual table cells in our ILP formulation.

ILP based discrete optimization has been successful in several NLP tasks [Roth and Yih, 2004; Chang *et al.*, 2010; Berant *et al.*, 2010; Srikumar and Roth, 2011; Goldwasser and Roth, 2011]. While our ILP formulation also operates on natural language text, our focus is on the use of a specific semi-structured table representation for QA. Cohen [2000] studied tables with natural language text requiring soft matching, with a focus on efficiently computing the top few candidates given a database query. In contrast, our system, given a natural language question, (implicitly) seeks to generate a query that produces the most supported answer.

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.

3.1 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

[1] 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%.

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.

Although 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.

3.2 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 $\mathbf{t}$.[2] 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.

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 1: Notation for the ILP formulation.

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 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 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. A summary of the model is given below. [3] 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 1 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

[2] 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.

[3] Details of the ILP model may be found in the appendix.

BASIC PAIRWISE ACTIVITY VARIABLES	
$y\left(t_{ijk}, t_{ij'k'}\right)$	cell to cell
$y\left(t_{ijk}, q_\ell\right)$	cell to question constituent
$y\left(h_{ik}, q_\ell\right)$	header to question constituent
$y\left(t_{ijk}, a_m\right)$	cell to answer option
$y\left(h_{ik}, a_m\right)$	header to answer option
$y\left(\ell_{ik}, a_m\right)$	column to answer option
$y\left(T_i, a_m\right)$	table to answer option
$y\left(\ell_{ik}, \ell_{ik'}\right)$	column to column relation
HIGH-LEVEL UNARY VARIABLES	
$x\left(T_i\right)$	active table
$x\left(r_{ij}\right)$	active row
$x\left(\ell_{ik}\right)$	active column
$x\left(h_{ik}\right)$	active column header
$x\left(q_\ell\right)$	active question constituent
$x\left(a_m\right)$	active answer option

Table 2: Variables used for defining the optimization problem for TableILP solver. All variables have domain $\{0, 1\}$.

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.

To make the definitions clearer, we introduce all basic variables used in our optimization in Table 2, and will use them later to define constraints explicitly. We use the notation $x\left(.\right)$ to refer to a unary variable parameterized by a single element of the optimization, and $y\left(.,.\right)$ to refer to a pairwise variable parameterized by a pair of elements. Unary variables represent the presence of a specific element as a node in the support graph G. For example $x\left(T_i\right) = 1$ if and only if the table T_i is active in G. Similarly, $y\left(t_{ijk}, q_\ell\right) = 1$ if and only if the corresponding edge is present in G, which we alternatively refer 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 alignment score $w(e)$ exceeds a pre-set threshold. For example, we create $y\left(t_{ijk}, t_{i'j'k'}\right)$ only if $w(t_{ijk}, t_{i'j'k'}) \geq$ MINCELLCELLALIGNMENT.[4]

The objective function is a weighted linear sum over all variables instantiated for a given question answering problem.[5] A small set of auxiliary variables is defined for linearizing complicated constraints.

Constraints are a significant part of our model, used for imposing the desired behavior on the support graph. Due to lack of space, we discuss only a representative subset here.[6]

Some constraints relate variables to each other. For example, unary variables are defined through constraints that relate them to the corresponding pairwise variables. For instance, for active row variable $x\left(r_{ij}\right)$, we ensure that it is 1 if and only if at least one cell in row j is active:

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

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

In the remainder of this section, we outline some of the important characteristics we expect in our model, and provide details of a few illustrative constraints.

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, in order to prevent one constituent or cell from having a large influence on the objective function and thereby the solution:

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

where $\mathcal{Q}_l$ is the set of all pairwise variables with one end in the 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:

$$\sum_j x\left(r_{ij}\right) \leq \text{MAXROWSPERTABLE}, \forall i$$

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.

[4] An exhaustive list of the minimum alignment thresholds for creating pairwise variables is in Table 10 in the appendix.

[5] The complete list of weights for unary and pairwise variables is included in Table 9 in the appendix.

[6] The complete list of the constraints is explained in Table 13 in the appendix.

Evidence Chaining
Questions requiring chaining of evidence from multiple tables, such as the example in Figure 1, are typically the most challenging in this domain. Chaining can be viewed as performing 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).

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.[7] 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, 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 3.1): 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.[8]

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

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

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

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.

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.

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.

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

Table 3: TableILP significantly outperforms both the prior MLN reasoner, and IR using identical knowledge as TableILP

As Table 3 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) 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 4: 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. [9] Table 4 shows the results, with the final combination at 69% representing a significant improvement over individual solvers.

ILP Solution Properties

Table 5 summarizes various ILP and support graph statistics for TableILP, averaged across all test questions.

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.[10] 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 5: 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.

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

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

4.3 Ablation Study

To quantify the importance of various components of our system, we performed several ablation experiments, summarized in Table 6 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 6: 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), thereby disallowing parallel evidence and evidence chaining (Section 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). 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 3.2). 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 3.2) 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.

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 7: 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 7 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.

5 Conclusion

Answering real science questions is a challenging task because they are posed in natural language, require extensive domain knowledge, and often require combining multiple facts together. We presented TableILP, a system that can answer such questions, 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. It also significantly boosts performance when combined with unstructured methods (IR and PMI). These results suggest that the approach is both viable and promising for natural language question answering.

Acknowledgments

D.K. is in part supported by AI2 and Google. The authors would like to thank Christos Christodoulopoulos, Sujay Jauhar, Sam Skjonsberg, and the Aristo Team at AI2 for invaluable discussions and insights.

References

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

[Angeli and Manning, 2014] G. Angeli and C. D. Manning. NaturalLI: Natural logic inference for common sense reasoning. In *EMNLP*, 2014.

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

[Bentivogli *et al.*, 2008] L. Bentivogli, P. Clark, I. Dagan, and D. Giampiccolo. The sixth pascal recognizing textual entailment challenge. In *TAC*, 2008.

[Berant *et al.*, 2010] J. Berant, I. Dagan, and J. Goldberger. Global learning of focused entailment graphs. In *ACL*, pg. 1220–1229, 2010.

[Bollacker *et al.*, 2008] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.

[Brill *et al.*, 2002] E. Brill, S. Dumais, and M. Banko. An analysis of the AskMSR question-answering system. In *Proceedings of EMNLP*, pg. 257–264, 2002.

[Carlson *et al.*, 2010] 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 *AAAI*, 2010.

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

[Church and Hanks, 1989] K. W. Church and P. Hanks. Word association norms, mutual information and lexicography. In *27th ACL*, pg. 76–83, 1989.

[Clark *et al.*, 2014] P. Clark, N. Balasubramanian, S. Bhakthavatsalam, K. Humphreys, J. Kinkead, A. Sabharwal, and O. Tafjord. Automatic construction of inference-supporting knowledge bases. In *4th AKBC Workshop*, Montreal, Canada, Dec 2014.

[Clark *et al.*, 2016] 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 *30th AAAI*, 2016.

[Clark, 2015] P. Clark. Elementary school science and math tests as a driver for AI: take the Aristo challenge! In *29th AAAI/IAAI*, pg. 4019–4021, Austin, TX, 2015.

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

[Dagan *et al.*, 2010] I. Dagan, B. Dolan, B. Magnini, and D. Roth. Recognizing textual entailment: Rational, evaluation and approaches–erratum. *Natural Language Engineering*, 16(01):105–105, 2010.

[Dagan *et al.*, 2013] I. Dagan, D. Roth, M. Sammons, and F. M. Zanzotto. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220, 2013.

[Dalvi *et al.*, 2016] B. Dalvi, S. Bhakthavatsalam, and P. Clark. IKE - an interactive tool for knowledge extraction. In *5th AKBC Workshop*, 2016.

[Fader *et al.*, 2014] A. Fader, L. Zettlemoyer, and O. Etzioni. Open question answering over curated and extracted knowledge bases. In *Proceedings of SIGKDD*, pg. 1156–1165, 2014.

[Ferrucci *et al.*, 2010] 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.

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

[Gunning *et al.*, 2010] 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.

[Howell, 2012] D. Howell. *Statistical methods for psychology*. Cengage Learning, 2012.

[Khot *et al.*, 2015] T. Khot, N. Balasubramanian, E. Gribkoff, A. Sabharwal, P. Clark, and O. Etzioni. Exploring Markov logic networks for question answering. In *2015 EMNLP*, Lisbon, Portugal, Sep 2015.

[Ko *et al.*, 2007] J. Ko, E. Nyberg, and L. Si. A probabilistic graphical model for joint answer ranking in question answering. In *Proceedings of SIGIR*, pg. 343–350, 2007.

[Kwok *et al.*, 2001] C. C. T. Kwok, O. Etzioni, and D. S. Weld. Scaling question answering to the web. In *WWW*, 2001.

[MacCartney, 2009] B. MacCartney. *Natural Language Inference*. PhD thesis, 2009.

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

[Novak, 1977] G. Novak. Representations of knowledge in a program for solving physics problems. In *IJCAI-77*, 1977.

[Richardson and Domingos, 2006] M. Richardson and P. Domingos. Markov logic networks. *Machine learning*, 62(1–2):107–136, 2006.

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

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

[Turney, 2013] P. D. Turney. Distributional semantics beyond words: Supervised learning of analogy and paraphrase. *TACL*, 1:353–366, 2013.

[Yao and Van Durme, 2014] X. Yao and B. Van Durme. Information extraction over structured data: Question answering with Freebase. In *52nd ACL*, 2014.

[Yih *et al.*, 2014] W. Yih, X. He, and C. Meek. Semantic parsing for single-relation question answering. In *ACL (2)*, pg. 643–648, 2014.

[Zou *et al.*, 2014] 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*, pg. 313–324, 2014.

A Appendix: ILP Model for TableILP

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

$$\text{maximize} \quad \mathbf{w}^{\mathrm{T}}\mathbf{x} \tag{1}$$

$$\text{subject to} \quad A\mathbf{x} \leq \mathbf{b}, \tag{2}$$

$$\text{and} \quad \mathbf{x} \in \mathbb{Z}^n, \tag{3}$$

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 1), and the constraints (A and $\mathbf{b}$ in Equation 2).

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

Table 8 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 2 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

Table 8: 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 $y\,(t_{ijk}, t_{i'j'k'})$ 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 10.

Table 2 also includes some high level unary variables, which help conveniently impose structural 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 1) are set according to Table 9. 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 13. 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 15, Table 13).

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 27, Table 13. Another example is that, G should contain at least a certain number of constituents in the question, which is modeled by constraint 30, Table 13.

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 28, Table 13), since letting the inference use any table could lead to unreasonably long, and likely error-prone, answer chains.

B Appendix: Features 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
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

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

Table 9: 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.

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 10: 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 11: Some of the important constants and their values in our model.

Collection of basic variables connected to header column k of table i:	$\mathcal{H}_{ik} = \{(h_{ik}, q_\ell); \forall l\} \cup \{(h_{ik}, a_m); \forall m\}$ (4)
Collection of basic variables connected to cell j, k of table i:	$\mathcal{E}_{ijk} = \{(t_{ijk}, t_{ij'k'}); \forall i', j', k'\} \cup \{(t_{ijk}, a_m); \forall m\} \cup \{(t_{ijk}, q_\ell); \forall l\}$ (5)
Collection of basic variables connected to column k of table i	$\mathcal{C}_{ik} = \mathcal{H}_{ik} \cup \left(\bigcup_j \mathcal{E}_{ijk}\right)$ (6)
Collection of basic variables connected to row j of table i:	$\mathcal{R}_{ij} = \bigcup_k \mathcal{E}_{ijk}$ (7)
Collection of non-choice basic variables connected to row j of table i:	$\mathcal{L}_{ij} = \{(t_{ijk}, t_{ij'k'}); \forall k, i', j', k'\} \cup \{(t_{ijk}, q_\ell); \forall k, l\}$ (8)
Collection of non-question basic variables connected to row j of table i:	$\mathcal{K}_{ij} = \{(t_{ijk}, t_{ij'k'}); \forall k, i', j', k'\} \cup \{(t_{ijk}, a_m); \forall k, m\}$ (9)
Collection of basic variables connected to table i:	$\mathcal{T}_i = \bigcup_k \mathcal{C}_{ik}$ (10)
Collection of non-choice basic variables connected to table i:	$\mathcal{N}_i = \{(h_{ik}, q_\ell); \forall l\} \cup \{(t_{ijk}, t_{ij'k'}); \forall j, k, i', j', k'\} \cup \{(t_{ijk}, q_\ell); \forall j, k, l\}$ (11)
Collection of basic variables connected to question constituent q_ℓ:	$\mathcal{Q}_l = \{(t_{ijk}, q_\ell); \forall i, j, k\} \cup \{(h_{ik}, q_\ell); \forall i, k\}$ (12)
Collection of basic variables connected to option m	$\mathcal{O}_m = \{(t_{ijk}, a_m); \forall i, j, k\} \cup \{(h_{ik}, a_m); \forall i, k\}$ (13)
Collection of basic variables in column k of table i connected to option m:	$\mathcal{M}_{i,k,m} = \{(t_{ijk}, a_m); \forall j\} \cup \{(h_{ik}, a_m)\}$ (14)

Table 12: All the sets useful in definitions of the constraints in Table 13.

If any cell in row j of table i is active, the row should be active.	$x(r_{ij}) \geq y(t_{ijk}, e), \forall (t_{ijk}, e) \in \mathcal{R}_{ij}, \forall i, j, k$ (15)
If the row j of table i is active, at least one cell in that row must be active as well.	$\sum_{(t_{ijk}, e) \in \mathcal{R}_{ij}} y(t_{ijk}, e) \geq x(r_{ij}), \forall i, j$ (16)
Column j header should be active if any of the basic variables with one end in this column header are active.	$x(h_{ik}) \geq y(h_{ik}, e), \forall (h_{ik}, e) \in \mathcal{H}_{ik}, \forall i, k$ (17)
If the header of column j variable is active, at least one basic variable with one end in the end in the header	$\sum_{(h_{ik}, e) \in \mathcal{H}_{ik}} y(h_{ik}, e) \geq x(h_{ik}), \forall i$ (18)
Column k is active if at least one of the basic variables with one end in this column are active.	$x(\ell_{ik}) \geq y(t_{ijk}, e), \forall (t_{ijk}, e) \in \mathcal{C}_{ik}, \forall i, k$ (19)

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

$$\sum_{(t_{ijk},e)\in\mathcal{C}_{ik}} y\left(t_{ijk},e\right) \geq x\left(h_{ik}\right), \forall i,k \tag{20}$$

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

$$y\left(t_{ijk},e\right) \geq x\left(T_i\right), \forall (t_{ijk},e) \in \mathcal{T}_i, \forall i \tag{21}$$

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

$$\sum_{(t,e)\in\mathcal{T}_i} y\left(t,e\right) \geq x\left(T_i\right), \forall i \tag{22}$$

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

$$x\left(a_m\right) \geq y\left(x,a_m\right), \forall (e,a_m) \in \mathcal{O}_m \tag{23}$$

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

$$\sum_{(e,a)\in\mathcal{O}_m} y\left(x,a\right) \geq x\left(a_m\right) \tag{24}$$

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

$$x\left(q_\ell\right) \geq y\left(e,q_\ell\right), \forall (e,q_\ell) \in \mathcal{Q}_l \tag{25}$$

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

$$\sum_{(e,q_\ell)\in\mathcal{Q}_l} y\left(e,q_\ell\right) \geq x\left(q_\ell\right) \tag{26}$$

Choose only a single option.

$$\sum_m x\left(a_m\right) \leq 1, \quad \sum_m x\left(a_m\right) \geq 1 \tag{27}$$

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.

$$\sum_i x\left(T_i\right) \leq \textsc{MaxTablesToChain} \tag{28}$$

The number of active rows in each table is upper-bounded.

$$\sum_j x\left(r_{ij}\right) \leq \textsc{MaxRowsPerTable}, \forall i \tag{29}$$

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.

$$\sum_l x\left(q_\ell\right) \geq \textsc{MinActiveQCons} \tag{30}$$

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

$$\sum_{(t_{ijk},e)\in\mathcal{E}_{i,j,k}} y\left(t_{ijk},e\right) \geq x\left(t_{ijk}\right) \times \textsc{MinActiveCellAggrAlignment}, \forall i,j,k \tag{31}$$

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

$$\sum_{(e)\in\mathcal{H}_{i,k}} y\left(t_{ijk},e\right) \geq x\left(t_{ijk}\right) \times \textsc{MinActiveTitleAggrAlignment}, \forall i,k \tag{32}$$

If a column is active, at least one of its cells must be active as well.

$$\sum_j x\left(t_{ijk}\right) \geq x\left(\ell_{ik}\right), \forall i,k \tag{33}$$

At most a certain number of columns can be active for a single option

$$\sum_k y\left(\ell_{ik},a_m\right) \leq \textsc{MaxActiveChoiceColumn}, \forall i,m \tag{34}$$

If a column is active for a choice, the table is active too.

$$x\left(\ell_{ik}\right) \leq x\left(T_i\right), \forall i,k \tag{35}$$

If a table is active for a choice, there must exist an active column for choice.

$$x\left(T_i\right) \leq \sum_k x\left(\ell_{ik}\right), \forall i \tag{36}$$

If a table is active for a choice, there must be some non-choice alignment.

$$y\left(T_i,a_m\right) \leq \sum_{(e,e')\in\mathcal{N}_i} y\left(e,e'\right), \forall i,m \tag{37}$$

Answer should be present in at most a certain number of tables

$$y\left(T_i,a_m\right) \leq \textsc{MaxActiveTableChoiceAlignmets}, \forall i,m \tag{38}$$

If a cell in a column, or its header is aligned with a question option, the column is active for question option as well.

$$y\left(t_{ijk},a_m\right) \leq y\left(\ell_{ik},a_m\right), \forall i,k,m, \forall (t_{ijk},a_m) \in \mathcal{M}_{i,k,m} \tag{39}$$

If a column is active for an option, there must exist an alignment to header or cell in the column.

$$y\left(\ell_{ik},a_m\right) \leq \sum_{(t_{ijk},a_m)\in\mathcal{O}_{i,k,m}} y\left(t_{ijk},a_m\right), \forall i,m \tag{40}$$

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, $\forall i, m$ (41)
If a column is active for a choice, the table is active for an option as well.	$y(\ell_{ik}, a_m) \leq y(T_i, a_m), \forall i, k, m$ (42)
If the table is active for an option, at least one column is active for a choice	$y(T_i, a_m) \leq \sum_k y(\ell_{ik}, a_m), \forall i, m$ (43)
Create an auxiliary variable x (whichTermIsActive) with objective weight 1.5 and activate it, if there a "which" term in the question.	$\sum_l 1\{q_\ell = \text{"which"}\} \leq x(\text{whichTermIsActive})$ (44)
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".	$\sum_i \sum_{(e_1,e_2)\in\mathcal{T}_i} y(e_1, e_2) \geq x(\text{whichTermIsAligned})$ (45)
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 (46)
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:	$y(t_{ijk}, q_\ell) + y(t_{ijk}, q_{\ell'}) \leq 1, \forall l, l', i, j, k$ (47)
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	$x(\text{cellProximityBoost}) \leq y(t_{ijk}, q_\ell)$, $x(\text{cellProximityBoost}) \leq y(t_{ijk}, q_{\ell'}), \forall i, j, k$ (48)
If a relation match is active, both the columns for the relation must be active	$r(\ell_{ik}, \ell_{ik'}, q_\ell, q_{\ell'}) \leq x(\ell_{ik}), r(\ell_{ik}, \ell_{ik'}, q_\ell, q_{\ell'}) \leq x(\ell_{ik'})$ (49)
If a column is active, a relation match connecting to the column must be active	$x(\ell_{ik}) \leq \sum_{k'}(r(\ell_{ik}, \ell_{ik'}, q_\ell, q_{\ell'}) + r(\ell_{ik'}, \ell_{ik}, q_\ell, q_{\ell'})), \forall k$ (50)
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 $t_{ijk} \in \ell_{ik}$ (51)
If a row is active, at least a certain number of its cells must be active	$\sum_k x(t_{ijk}) \geq$ MINACTIVECELLSPERROW $\times x(r_{ij}), \forall i, j$ (52)
If row is active, it must have non-choice alignments.	$x(r_{ij}) \leq \sum_{(n,n')\in\mathcal{L}_{ij}} y(n, n)$ (53)
If row is active, it must have non-question alignments	$x(r_{ij}) \leq \sum_{(n,n')\in\mathcal{K}_{ij}} y(n, n)$ (54)
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	$x(r_{ij}) + x(r_{ij'}) + x(t_{ijk}) - x(t_{ij'k'}) \leq 2, \forall i, j, j', k, k'$ (55)
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.	$\sum_{t_{ijk}\neq t_{ijk'}} x(\ell_{ik}) - x(r_{ij}) - x(r_{ij'}) \geq -1$ (56)
If a table is active and another table is also active, at least one inter-table active variable must be active;	$x(T_i) + x(T_{i'}) + \sum_{j,k,j',k'} y(t_{ijk}, t_{i'j'k'}) \geq 1, \forall i, i'$ (57)

Table 13: The set of all constraints used in our ILP formulation. The set of variables and are defined in Table 2. More intuition about constraints is included in Section 3. The sets used in the definition of the constraints are defined in Table 12.