Answering Complex Questions Using Open Information Extraction

Tushar Khot and Ashish Sabharwal and Peter Clark
Allen Institute for Artificial Intelligence, Seattle, WA, U.S.A.
{tushark,ashishs,peterc}@allenai.org

Abstract

While there has been substantial progress in factoid question-answering (QA), answering complex questions remains challenging, typically requiring both a large body of knowledge and inference techniques. Open Information Extraction (Open IE) provides a way to generate semi-structured knowledge for QA, but to date such knowledge has only been used to answer simple questions with retrieval-based methods. We overcome this limitation by presenting a method for reasoning with Open IE knowledge, allowing more complex questions to be handled. Using a recently proposed support graph optimization framework for QA, we develop a new inference model for Open IE, in particular one that can work effectively with multiple short facts, noise, and the relational structure of tuples. Our model significantly outperforms a state-of-the-art structured solver on complex questions of varying difficulty, while also removing the reliance on manually curated knowledge.

1 Introduction

Effective question answering (QA) systems have been a long-standing quest of AI research. Structured curated KBs have been used successfully for this task (Berant et al., 2013; Berant and Liang, 2014). However, these KBs are expensive to build and typically domain-specific. Automatically constructed open vocabulary (subject; predicate; object) style tuples have broader coverage, but have only been used for simple questions where a single tuple suffices (Fader et al., 2014; Yin et al., 2015).

Our goal in this work is to develop a QA system that can perform reasoning with Open IE (Banko et al., 2007) tuples for complex multiple-choice questions that require tuples from multiple sentences. Such a system can answer complex questions in resource-poor domains where curated knowledge is unavailable. Elementary-level science exams is one such domain, requiring complex reasoning (Clark, 2015). Due to the lack of a large-scale structured KB, state-of-the-art systems for this task either rely on shallow reasoning with large text corpora (Clark et al., 2016; Cheng et al., 2016) or deeper, structured reasoning with a small amount of automatically acquired (Khot et al., 2015) or manually curated (Khashabi et al., 2016) knowledge.

Consider the following question from an Alaska state 4th grade science test:

**Which object in our solar system reflects light and is a satellite that orbits around one planet?** (A) Earth (B) Mercury (C) the Sun (D) the Moon

This question is challenging for QA systems because of its complex structure and the need for multi-fact reasoning. A natural way to answer it is by combining facts such as (Moon; is; in the solar system), (Moon; reflects; light), (Moon; is; satellite), and (Moon; orbits; around one planet).

A candidate system for such reasoning, and which we draw inspiration from, is the TABLEILP system of Khashabi et al. (2016). TABLEILP treats QA as a search for an optimal subgraph that connects terms in the question and answer via rows in a set of curated tables, and solves the optimization problem using Integer Linear Programming (ILP). We similarly want to search for an optimal subgraph. However, a large, automatically extracted tuple KB makes the reasoning context different on three fronts: (a) unlike reasoning with tables, chaining tuples is less important and reliable as join rules aren’t
available; (b) conjunctive evidence becomes paramount, as, unlike a long table row, a single tuple is less likely to cover the entire question; and (c) again, unlike table rows, tuples are noisy, making combining redundant evidence essential. Consequently, a table-knowledge centered inference model isn’t the best fit for noisy tuples.

To address this challenge, we present a new ILP-based model of inference with tuples, implemented in a reasoner called TUPLEINF. We demonstrate that TUPLEINF significantly outperforms TABLEILP by 11.8% on a broad set of over 1,300 science questions, without requiring manually curated tables, using a substantially simpler ILP formulation, and generalizing well to higher grade levels. The gains persist even when both solvers are provided identical knowledge. This demonstrates for the first time how Open IE based QA can be extended from simple lookup questions to an effective system for complex questions.

2 Related Work

We discuss two classes of related work: retrieval-based web question-answering (simple reasoning with large scale KB) and science question-answering (complex reasoning with small KB).

Web QA: There exist several systems for retrieval-based Web QA problems (Ferrucci et al., 2010; Brill et al., 2002). While structured KBs such as Freebase have been used in many (Berant et al., 2013; Berant and Liang, 2014; Kwiatkowski et al., 2013), such approaches are limited by the coverage of the data. QA systems using semi-structured Open IE tuples (Fader et al., 2013, 2014; Yin et al., 2015) or automatically extracted web tables (Sun et al., 2016; Pasupat and Liang, 2015) have broader coverage but are limited to simple questions with a single query.

Science QA: Elementary-level science QA tasks require reasoning to handle complex questions. Markov Logic Networks (Richardson and Domingos, 2006) have been used to perform probabilistic reasoning over a small set of logical rules (Khot et al., 2015). Simple IR techniques have also been proposed for science tests (Clark et al., 2016) and Gaokao tests (equivalent to the SAT exam in China) (Cheng et al., 2016).

The work most related to TUPLEINF is the aforementioned TABLEILP solver. This approach focuses on building inference chains using manually defined join rules for a small set of curated tables. While it can also use open vocabulary tuples (as we assess in our experiments), its efficacy is limited by the difficulty of defining reliable join rules for such tuples. Further, each row in some complex curated tables covers all relevant contextual information (e.g., each row of the adaptation table contains (animal, adaptation, challenge, explanation)), whereas recovering such information requires combining multiple Open IE tuples.

3 Tuple Inference Solver

We first describe the tuples used by our solver. We define a tuple as (subject; predicate; objects) with zero or more objects. We refer to the subject, predicate, and objects as the fields of the tuple.

3.1 Tuple KB

We use the text corpora (S) from Clark et al. (2016) to build our tuple KB. S contains $5 \times 10^{19}$ tokens (280 GB of plain text) extracted from Web pages as well as around 80,000 sentences from various domain-targeted sources. For each test set, we use the corresponding training questions $Q_{tr}$ to retrieve domain-relevant sentences from S. Specifically, for each multiple-choice question $(q, A) \in Q_{tr}$ and each choice $a \in A$, we use all non-stopword stemmed tokens in $q$ and $a$ as an ElasticSearch query against S. We take the top 200 hits, run Open IE v4.2 and aggregate the resulting tuples over all $a \in A$ and over all questions in $Q_{tr}$ to create the tuple KB ($T$).3

3.2 Tuple Selection

Given a multiple-choice question $qa$ with question text $q$ and answer choices $A=\{a_i\}$, we select the most relevant tuples from $T$ and $S$ as follows.

Selecting from Tuple KB: We use an inverted index to find the 1,000 tuples that have the most overlapping tokens with question tokens $tok(qa)$.4 We also filter out any tuples that overlap only with $tok(q)$ as they do not support any answer. We compute the normalized TF-IDF score by treating the question, $q$, as a query and each tuple, $t$, as a

---

1https://www.elastic.co/products/elasticsearch
2http://knowitall.github.io/openie
3Available at http://allenai.org/data.html
4All tokens are stemmed and stop-word filtered.
subgraph of $G(V, E)$ where $V$ and $E$ denote “active” nodes and edges, resp. We define an ILP optimization model to search for the best support graph (i.e., the active nodes and edges) as follows.

### Variables

The ILP has a binary variable for each qterm $(x_q)$, tuple $(x_t)$, tuple field $(x_f)$, and answer choice $(x_a)$, indicating whether the corresponding graph node is active. There is a binary activity variable $(x_e)$ for each edge $e \in E$. For efficiency, we only create a qterm→field edge and a field→choice edge if the corresponding coefficient is no smaller than a certain threshold (0.1 and 0.2, resp.).

### Objective Function

The objective function coefficient $c_e$ of each edge $e(t, h)$ is determined by a word-overlap score.\(^2\)

While TABLEILP used WordNet (Miller, 1995) paths to compute the edge weight, this measure results in unreliable scores when faced with longer phrases found in Open IE tuples.

Compared to a curated KB, it is easy to find Open IE tuples that match irrelevant parts of the questions. To mitigate this issue, we scale the coefficients $c_q$ of qterms in our ILP objective to focus on important terms. Since the later terms in a question tend to provide the most critical information, we scale qterm coefficients based on their position in the question. Also, qterms that appear in almost all of the selected tuples tend not to be discriminative as any tuple would support such a qterm. Hence we scale qterm coefficients inversely by the frequency with which they occur in the selected tuples. Appendix A describes the coefficient for qterm as well as other variables in detail.

### Constraints

Since Open IE tuples do not come with schema and join rules, we can define a substantially simpler model compared to TABLEILP. This reduces the reasoning capability but also eliminates the reliance on hand-authored join rules and regular expressions used in TABLEILP. We discovered (see empirical evaluation) that this simple model can achieve the same score as TABLEILP on the Regents test (target test set used by TABLEILP) and generalizes better to different grade levels.

We start with a few constraints defining what is an active node or edge, shown as the first groups of constraints in Table 1. To avoid positive edge coefficients in the objective function resulting in

\(^{2}\)\(w(t,h) = | tok(t) \cap tok(h) | / | tok(t) | \)

\(^{6}\)\(tok(t) \cap tok(qa) | / | tok(t) \cup tok(qa) | \)

\(^{5}\)containing not, ’nt, or except
ordering constraint (fourth group in Table 1).

object

the sample question in the introduction (“Which

(Planet; orbit; Sun)

avoid issues such as

we require that the subject must be active. To

ple expresses a fact about the tuple’s subject,

Open IE tuple structure. Since an Open IE tu-

choice (third group of constraints in Table 1).

quire each tuple to connect a qterm with an answer

expressed in the tuple, we add constraints that re-

question (or choice) or ignore the relation being

parts of the question as illustrated in Figure 1. To

pable of using multiple tuples to support different

of constraints in Table 1). Our model is also ca-

number of active edges from an active tuple, ques-

spurious edges in the support graph, we limit the

Active qterm must have < w3 edges

Support graph must have < w4 active tuples

Active tuple must have ≥ w5 active fields

Active tuple must have an edge to some qterm

Active tuple must have an edge to some choice

Active tuple must have active subject

If a tuple predicate aligns to q, the subject (object) must

align to a term preceding (following, resp.) q

Table 1: High-level ILP constraints; we report re-

Table 2: TUPLEINF is significantly better at struc-

tured reasoning than TABLEILP.  

edge as well as with additional tuples, and (b) TU-

PLEINF’s complementary approach to IR leads to an

improved ensemble. Numbers in bold indicate

statistical significance based on the Binomial ex-

act test (Howell, 2012) at p = 0.05.

We consider two question sets. (1) 4th Grade

set (1220 train, 1304 test) is a 10x larger superset

of the NY Regents questions (Clark et al., 2016),

and includes professionally written licensed ques-

ions. (2) 8th Grade set (293 train, 282 test) con-

ains 8th grade questions from various states.

We consider two knowledge sources:

(1) The Sentence corpus (S) consists of domain-targeted 80K sentences and 280 GB of plain text extracted from web pages used by Clark et al. (2016). This corpus is used as a collection of sentences by the IR solver. It is also used to create the tuple KB T (Sec. 3.1) and on-the-fly question-
specific tuples T_qa (Sec. 3.2) for TUPLEINF.

(2) TABLEILP uses ∼70 Curated tables (C) containing about 7,600 rows, designed for 4th grade NY Regents exams.

We compare TUPLEINF with two state-of-the-

art baselines. IR is a simple yet powerful

information-retrieval baseline (Clark et al., 2016) that

selects the answer option with the best matching

sentence in a corpus. TABLEILP is the state-

of-the-art structured inference baseline (Khashabi et al., 2016) developed for science questions.

4 Experiments

Comparing our method with two state-of-the-art systems for 4th and 8th grade science exams, we demonstrate that (a) TUPLEINF with only automatically extracted tuples significantly outper-

forms TABLEILP with its original curated knowl-

Table 2: Solvers 4th Grade 8th Grade

<table>
<thead>
<tr>
<th>Solver</th>
<th>4th Grade</th>
<th>8th Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLEILP(C)</td>
<td>39.9</td>
<td>34.1</td>
</tr>
<tr>
<td>TUPLEINF(T+T’)</td>
<td>51.7</td>
<td>51.6</td>
</tr>
<tr>
<td>TABLEILP(C+T)</td>
<td>42.1</td>
<td>37.9</td>
</tr>
<tr>
<td>TUPLEINF(C+T)</td>
<td>47.5</td>
<td>48.0</td>
</tr>
</tbody>
</table>

4.1 Results

Table 2 shows that TUPLEINF, with no curated

knowledge, outperforms TABLEILP on both question sets by more than 11%. The lower half of the table shows that even when both solvers are given

See the Middle School Without Diagrams set from AI2 Science Questions v1 (Feb 2016) at http://allenai.org/data/science-exam-questions.html for the 8th Grade set. For future comparisons, we also report our score on their smaller 4th Grade set: Elementary School Without Diagrams (432 train, 339 test).

TUPLEINF(T+T’) achieves a score of 56.1% on the Elementary School Without Diagrams test set (cf. Footnote 8) compared to TABLEILP(C)’s score of 46.7%.
Table 3: TUPLEINF is complementarity to IR, resulting in a strong ensemble

<table>
<thead>
<tr>
<th>Solvers</th>
<th>4th Grade</th>
<th>8th Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>IR(S)</td>
<td>52.0</td>
<td>52.8</td>
</tr>
<tr>
<td>IR(S) + TABLEILP(C)</td>
<td>53.3</td>
<td>54.5</td>
</tr>
<tr>
<td>IR(S) + TUPLEINF(T+T’)</td>
<td>55.3</td>
<td>55.1</td>
</tr>
</tbody>
</table>

Table 3 shows that while TUPLEINF achieves similar scores as the IR solver, the approaches are complementary (structured lossy knowledge reasoning vs. lossless sentence retrieval). The two solvers, in fact, differ on 47.3% of the training questions. To exploit this complementarity, we train an ensemble system (Clark et al., 2016) which, as shown in the table, provides a substantial boost over the individual solvers. Further, IR + TUPLEINF is consistently better than IR + TABLEILP.

Finally, in combination with IR and the statistical association based PMI solver (which scores 54.1% by itself) of Clark et al. (2016), TUPLEINF achieves a score of 58.2% on the 4th grade set. This compares favorably to TABLEILP’s ensemble score of 56.7%, again attesting to TUPLEINF’s strength.12

5 Error Analysis

We describe four classes of failure of TUPLEINF, and the future work they suggest.

Missing Important Words: Which material will spread out to completely fill a larger container? (A) air (B) ice (C) sand (D) water

In this question, we have tuples that support that water will spread out and fill a larger container, but miss the critical word “completely”. A method for detecting salient question words would help here.

Lossy IE: Which action is the best method to separate a mixture of salt and water? . . .

The IR solver correctly answers this question by using the sentence: Separate the salt and water mixture by evaporating the water. However, TUPLEINF is not able to answer this question as Open IE is unable to extract tuples from this imperative sentence. While the additional structure from Open IE is generally helpful for more robust matching, the conversion to tuples sometimes loses important bits of information.

Bad Alignment: Which of the following gases is necessary for humans to breathe in order to live? (A) Oxygen (B) Carbon dioxide (C) Helium (D) Water vapor

TUPLEINF returns “Carbon dioxide” as the answer because of the tuple (humans; breathe out; carbon dioxide). The chunk “to breathe” in the question has a high alignment score to the “breathe out” relation in the tuple, even though they have completely different meaning. An improved phrase alignment module can mitigate this issue.

Out of Scope: Deer live in forest for shelter. If the forest was cut down, which situation would most likely happen? . . .

Such questions require modeling a state presented in the question and reasoning over this state, which is out of scope of our solver.

6 Conclusion

We presented a new QA system, TUPLEINF, that can reason over a large, potentially noisy knowledge base of (subject, predicate, object) style tuples, in order to answer complex questions. Our results establish TUPLEINF as a new state-of-the-art structured reasoning solver for elementary-level science that does not rely on curated knowledge and generalizes to higher grade levels. Our error analysis points to lossy IE and textual misalignments as two main causes of failure, suggesting future work around incorporating tuple context and distributional similarity measures.

Acknowledgments

The authors would like to thank Oren Etzioni for valuable feedback on an early draft of this paper, and Colin Arenz and Michal Guerquin for helping us develop this system.
References


Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*.


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

Peter Clark, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Turney, and Daniel Khashabi. 2016. Combining retrieval, statistics, and inference to answer elementary science questions. In *30th AAAI*.


Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Peter Clark, Oren Etzioni, and Dan Roth. 2016. Question answering via integer programming over semi-structured knowledge. In *IJCAI*.


Huan Sun, Hao Ma, Xiaodong He, Wen tau Yih, Yu Su, and Xifeng Yan. 2016. Table cell search for question answering. In *WWW*.

Pengcheng Yin, Nan Duan, Ben Kao, Jun-Wei Bao, and Ming Zhou. 2015. Answering questions with complex semantic constraints on open knowledge bases. In *CIKM*.
A Appendix: ILP Model Details

To build the ILP model, we first obtain the questions terms (qterm) from the question by chunking the question using an in-house chunker based on the postagger from FACTORIE. \(^\text{13}\) We ignore chunks that only contain stop-words.

Variables

The ILP model has an active vertex variable for each qterm \((x_q)\), tuple \((x_t)\), tuple field \((x_f)\) and question choice \((x_a)\). Table 4 describes the coefficients of these active variables. For example, the coefficient of each qterm is a constant value \((0.8)\) scaled by three boosts. The idf boost, \(idfB\) for a qterm, \(x\) is calculated as \(\log(1 + (|T_{qa}| + |T'_{qa}|)/n_x)\) where \(n_x\) is the number of tuples in \(T_{qa} \cup T'_{qa}\) containing \(x\). The science term boost, \(scienceB\) (set to 2.0) boosts coefficients of qterms that are valid science terms based on a list of 9K terms. The location boost, \(locB\) of a qterm at index \(i\) in the question is given by \(i/tok(q)\) (where \(i=1\) for the first term). Finally the objective function of our ILP model can be written as:

\[
\sum_{q \in \text{qterms}} c_{q} x_{q} + \sum_{t \in \text{tuples}} c_{t} x_{t} + \sum_{e \in \text{edges}} c_{e} x_{e}
\]

<table>
<thead>
<tr>
<th>Description</th>
<th>Var.</th>
<th>Coefficient (c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qterm</td>
<td>(x_q)</td>
<td>0.8: (idfB)-(scienceB)-(locB)</td>
</tr>
<tr>
<td>Tuple</td>
<td>(x_t)</td>
<td>(-1 + \text{jaccardScore}(t, qa))</td>
</tr>
<tr>
<td>Tuple Field</td>
<td>(x_f)</td>
<td>0</td>
</tr>
<tr>
<td>Choice</td>
<td>(x_a)</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4: Coefficients for active variables.

Constraints

Apart from the constraints described in Table 1, we also use the which-term boosting constraints defined by TABLEILP (Eqns. 44 and 45 in Table 13 (Khashabi et al., 2016)). As described in Section B, we create a tuple from table rows by setting pairs of cells as the subject and object of a tuple. For these tuples, apart from requiring the subject to be active, we also require the object of the tuple to be active. This would be equivalent to requiring at least two cells of a table row to be active.

B Experiment Details

We use the SCIP ILP optimization engine (Achterberg, 2009) to solve our ILP model. To get the score for each answer choice \(a_i\), we force the active variable for that choice \(x_{a_i}\) to be one and use the objective function value of the ILP model as the score. 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. For evaluations, we use a 2-core 2.5 GHz Amazon EC2 linux machine with 16 GB RAM.

B.1 Using curated tables with TUPLEINF

For each question, we select the 7 best matching tables using the tf-idf score of the table w.r.t. the question tokens and top 20 rows from each table using the Jaccard similarity of the row with the question. (same as Khashabi et al. (2016)). We then convert the table rows into the tuple structure using the relations defined by TABLEILP. For every pair of cells connected by a relation, we create a tuple with the two cells as the subject and primary object with the relation as the predicate. The other cells of the table are used as additional objects to provide context to the solver. We pick top-scoring 50 tuples using the Jaccard score.

B.2 Using Open IE tuples with TABLEILP

We create an additional table, \(T_O\) in TABLEILP using all the tuples in our KB, \(T\). Since TABLEILP uses fixed-length \((subject; predicate; object)\) triples, we need to map tuples with multiple objects to this format. For each object, \(O_i\) in the input Open IE tuple \((S; P; O_1; O_2 \ldots) \in T\), we add a triple \((S; P; O_i)\) to the table, \(T_O\).

\(^{13}\)http://factorie.cs.umass.edu