Learning What is Essential in Questions

Daniel Khashabi[†] Univ. of Pennsylvania danielkh@cis.upenn.edu

Tushar Khot Allen Institute for AI tushark,ashishs@allenai.org

Ashish Sabharwal Allen Institute for AI tushark,ashishs@allenai.org

Dan Roth[†] Univ. of Pennsylvania danroth@cis.upenn.edu

Abstract

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

We also introduce a new dataset of over 2,200 crowd-sourced essential terms annotated science questions.

1 Introduction

Understanding what a question is really about is a fundamental challenge for question answering systems that operate with a natural language interface. In domains with multi-sentence questions covering a wide array of subject areas, such as standardized tests for elementary level science, the challenge is even more pronounced (Clark, 2015). Many QA systems in such domains derive significant leverage from relatively shallow Information Retrieval (IR) and statistical correlation techniques operating on large unstructured corpora (Kwok et al., 2001; Clark et al., 2016). Inference based QA systems operating on (semi-)structured knowledge formalisms have also demonstrated complementary strengths, by using optimization formalisms such as Semantic Parsing (Yih et al., 2014), Integer Linear Program (ILP) (Khashabi et al., 2016), and probabilistic logic formalisms such as Markov Logic Networks (MLNs) (Khot et al., 2015).

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

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

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

Towards this goal, we propose a system that can assign an essentiality score to each term in the question. For the above example, our system gen-

† Most of the work was done when the first and last authors were affiliated with the University of Illinois, Urbana-Champaign.

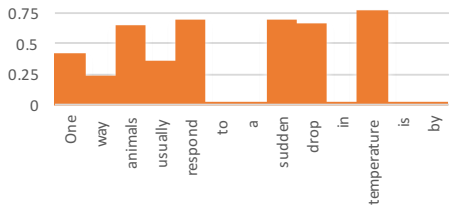

Figure 1: Essentiality scores generated by our system, which assigns high essentiality to "drop" and "temperature".

erates the scores shown in Figure 1, where more weight is put on "temperature" and "sudden drop". A QA system, when armed with such information, is expected to exhibit a more informed behavior.

We make the following contributions:

(A) We introduce the notion of *question term essentiality* and release a new dataset of 2,223 crowd-sourced essential term annotated questions (total 19K annotated terms) that capture this concept.[1] We illustrate the importance of this concept by demonstrating that humans become substantially worse at QA when even a few essential question terms are dropped.

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

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

1.1 Related Work

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

There is related work on extracting *question type* information (Li and Roth, 2002; Li et al., 2007) and applying it to the design and analysis of end-to-end QA systems (Moldovan et al., 2003). The concept of term essentiality studied in this work is different, and so is our supervised learning approach compared to the typical rule-based systems for question type identification.

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

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

One key difference from general text summarization literature is that we operate on questions, which tend to have different essentiality characteristics than, say, paragraphs or news articles. As we discuss in Section 2.1, typical indicators of essentiality such as being a proper noun or a verb (for event extraction) are much less informative for questions. Similarly, while the opening sentence of a Wikipedia article is often a good summary, it is the last sentence (in multi-sentence questions) that contains the most pertinent words.

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

2 Essential Question Terms

In this section, we introduce the notion of *essential question terms*, present a dataset annotated with these terms, and describe two experimental studies that illustrate the importance of this notion—we show that when dropping terms from questions, humans' performance degrades significantly faster if the dropped terms are essential question terms.

Given a question q, we consider each non-

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

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

2.1 Crowd-Sourced Essentiality Dataset

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

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

On average, less than one-third (29.9%) of the terms in each question were marked as essential (i.e., score > 0.5). This shows the large proportion of distractors in these science tests (as compared to traditional QA datasets), further showing the importance of this task. Next we provide some insights into these terms.

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

The best single indicator of essential terms, not surprisingly, was being a scientific term[5] (such as *precipitation* and *gravity*). 76.6% of such terms occurring in questions were marked as essential.

In summary, we have a term essentiality annotated dataset of 2,223 questions. We split this into train/development/test subsets in a 70/9/21 ratio, resulting in 483 test sentences used for per-question evaluation.

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

2.2 The Importance of Essential Terms

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

To this end, we consider both the annotated essentiality scores as well as the score produced by our trained classifier (to be presented in Section 3). We first generate candidate sets of terms to eliminate using these essentiality scores based on a threshold $\xi \in \{0, 0.2, \ldots, 1.0\}$: (a) **essential set**: terms with score $\geq \xi$; (b) **non-essential set**: terms

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

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

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

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

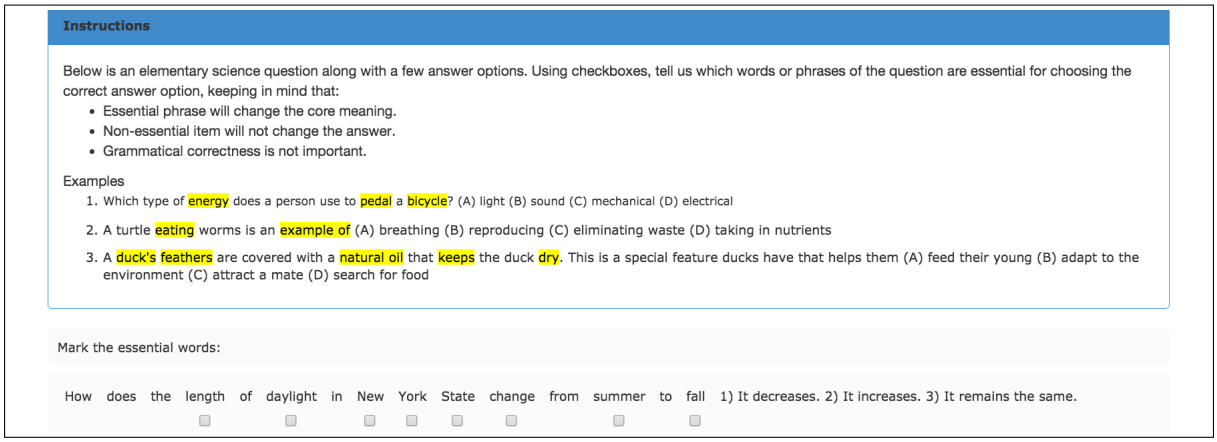

Figure 2: Crowd-sourcing interface for annotating essential terms in a question, including the criteria for essentiality and sample annotations.

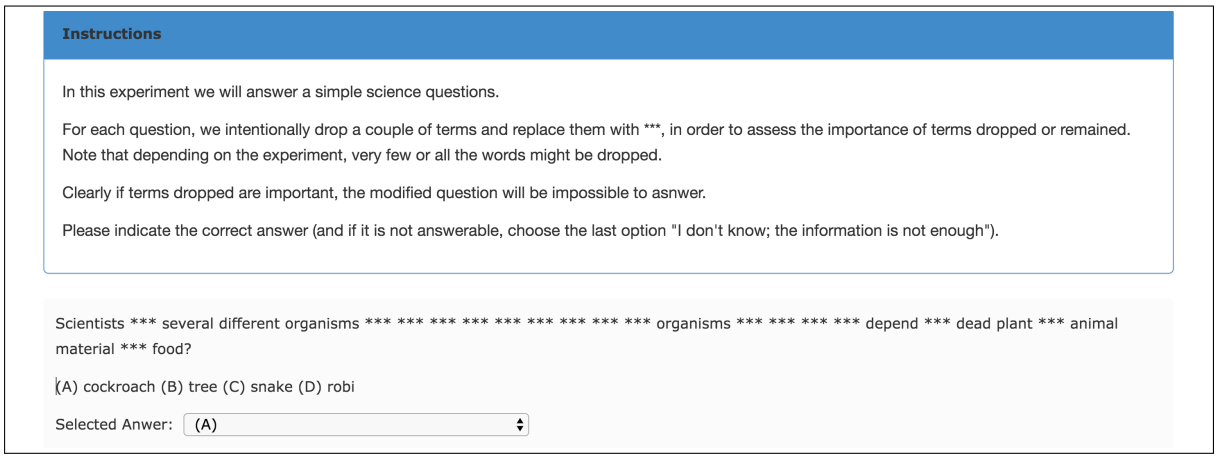

Figure 3: Crowd-sourcing interface for verifying the validity of essentiality annotations generated by the first task. Annotators are asked to answer, if possible, questions with a group of terms dropped.

with score $< \xi$. We then ask crowd workers to try to answer a question after replacing each candidate set of terms with "***". In addition to four original answer options, we now also include "I don't know. The information is not enough" (cf. Figure 3 for the user interface).[6] For each value of ξ, we obtain 5×269 annotations for 269 questions. We measure how often the workers feel there is sufficient information to attempt the question and, when they do attempt, how often do they choose the right answer.

Each value of ξ results in some fraction of terms to be dropped from a question; the exact number depends on the question and on whether we use annotated scores or our classifier's scores. In Figure 4, we plot the average fraction of terms dropped on the horizontal axis and the corresponding fraction of questions attempted on the vertical axis. Solid lines indicate annotated scores and dashed lines indicate classifier scores. Blue lines (bottom left) illustrate the effect of eliminating essential sets while red lines (top right) reflect eliminating non-essential sets.

We make two observations. First, the solid blue line (bottom-left) demonstrates that dropping even a small fraction of question terms marked as essential dramatically reduces the QA performance of humans. E.g., dropping just 12% of the terms (with high essentiality scores) makes 51% of the questions unanswerable. The solid red line (top-right), on the other hand, shows the opposite trend for terms marked as not-essential: even after drop-

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

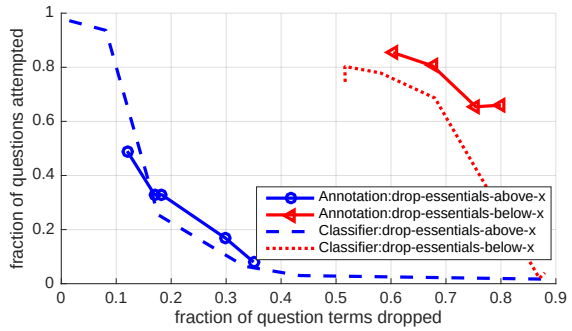

Figure 4: The relationship between the fraction of question words dropped and the fraction of the questions attempted (fraction of the questions workers felt comfortable answering). Dropping most essential terms (blue lines) results in very few questions remaining answerable, while least essential terms (red lines) allows most questions to still be answerable. Solid lines indicate human annotation scores while dashed lines indicate predicted scores.

ping 80% of such terms, 65% of the questions remained answerable.

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

3 Essential Terms Classifier

Given the dataset of questions and their terms annotated with essential scores, is it possible to learn the underlying concept? Towards this end, given a question q , answer options a, and a question term q_l, we seek a classifier that predicts whether q_l is essential for answering q. We also extend it to produce an essentiality score $et(q_l, q, a) \in [0, 1]$.[7] We use the annotated dataset from Section 2, where real-valued essentiality scores are binarized to 1 if they are at least 0.5, and to 0 otherwise.

We train a linear SVM classifier (Joachims, 1998), henceforth referred to as **ET classifier**. Given the complex nature of the task, the features of this classifier include syntactic (e.g., dependency parse based) and semantic (e.g., Brown cluster representation of words (Brown et al., 1992), a list of scientific words) properties of question words, as well as their combinations. In total, we use 120 types of features (cf. Appendix A of our Extended edition (Khashabi et al., 2017)).

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

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

First, for our supervised baseline, given (q_l, q, a) as before, we ignore q and compute how often is q_l annotated as essential in the entire dataset. In other words, the score for q_l is the proportion of times it was marked as essential in the annotated dataset. If the instance is never observer in training, we choose an arbitrary label as prediction. We refer to this baseline as *label proportion baseline* and create two variants of it: PROPSURF based on surface string and PROPLEM based on lemmatizing the surface string. For unseen q_l, this baseline makes a random guess with uniform distribution.

Our unsupervised baseline is inspired by work on sentence compression (Clarke and Lapata, 2008) and the PMI solver of Clark et al. (2016), which compute word importance based on co-occurrence statistics in a large corpus. In a corpus $\mathcal{C}$ of 280 GB of plain text (5×10^{10} tokens) extracted from Web pages,[8] we identify unigrams, bigrams, trigrams, and skip-bigrams from q and each answer option a_i. For a pair (x, y) of n-grams, their pointwise mutual information (PMI) (Church and Hanks, 1989) in $\mathcal{C}$ is defined as $\log \frac{p(x,y)}{p(x)p(y)}$ where $p(x, y)$ is the co-occurrence frequency of x and y (within some window) in $\mathcal{C}$. For a given word x, we find all pairs of question n-grams and answer option n-grams. MAXPMI and SUMPMI score the importance of a word x by max-ing or summing, resp., PMI scores $p(x, y)$ across all answer options y for q. A limitation of this baseline is its dependence on the existence of answer options, while our system makes essentiality predictions independent of the answer options.

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

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

3.1 Evaluation

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

Binary Classification of Terms. We consider all question terms pooled together as described in Section 2.1, resulting in a dataset of 19,380 terms annotated (in the context of the corresponding question) independently as essential or not. The ET classifier is trained on the train subset, and the threshold is tuned using the dev subset.

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

Table 1: Effectiveness of various methods for identifying essential question terms in the test set, including area under the PR curve (AUC), accuracy (Acc), precision (P), recall (R), and F1 score. ET classifier substantially outperforms all supervised and unsupervised (denoted with †) baselines.

For each term in the corresponding test set of 4,124 instances, we use various methods to predict whether the term is essential (for the corresponding question) or not. Table 1 summarizes the resulting performance. For the threshold-based scores, each method was tuned to maximize the F1 score based on the dev set. The ET classifier achieves an F1 score of 0.80, which is 5%-14% higher than the baselines. Its accuracy at 0.75 is statistically significantly better than all baselines based on the Binomial[9] exact test (Howell, 2012) at p-value 0.05.

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

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

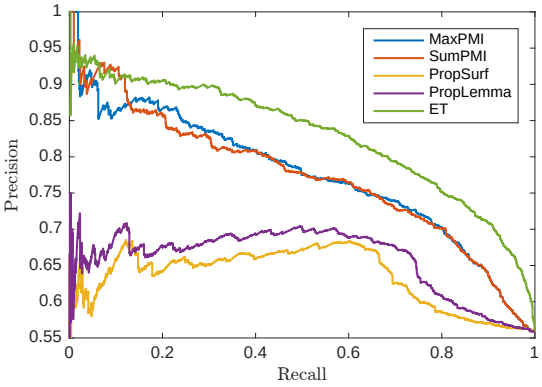

Figure 5: Precision-recall trade-off for various classifiers as the threshold is varied. ET classifier (green) is significantly better throughout.

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

Table 2: Generalization to unseen terms: Effectiveness of various methods, using the same metrics as in Table 1. As expected, supervised methods perform poorly, similar to a random baseline. Unsupervised methods generalize well, but the ET classifier again substantially outperforms them.

As a second study, we assess how well our classifier **generalizes to unseen terms**. For this, we consider only the 559 test terms that do not appear in the train set.[10] Table 2 provides the resulting performance metrics. We see that the frequency based supervised baselines, having never seen the test terms, stay close to the default precision of 0.5. The unsupervised baselines, by nature, generalize much better but are substantially dominated by our ET classifier, which achieves an F1 score of 78%. This is only 2% below its own F1 across all seen and unseen terms, and 6% higher than the second best baseline.

Ranking Question Terms by Essentiality. Next, we investigate the performance of the ET classifier as a system that ranks all terms within a question in the order of essentiality. Thus,

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

System	MAP
MAXPMI †	0.87
SUMPMI †	0.85
PROPSURF	0.85
PROPLEM	0.86
ET Classifier	**0.90**

Table 3: Effectiveness of various methods for ranking the terms in a question by essentiality. † indicates unsupervised method. Mean-Average Precision (MAP) numbers reflect the mean (across all test set questions) of the average precision of the term ranking for each question. ET classifier again substantially outperforms all baselines.

unlike the previous evaluation that pools terms together across questions, we now consider each question as a unit. For the ranked list produced by each classifier for each question, we compute the average precision (AP).[11] We then take the mean of these AP values across questions to obtain the mean average precision (MAP) score for the classifier.

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

4 Using ET Classifier in QA Solvers

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

Question Sets. We use three question sets of 4-way multiple choice questions.[12] REGENTS and AI2PUBLIC are two publicly available elementary school science question set. REGENTS comes with 127 training and 129 test questions; AI2PUBLIC contains 432 training and 339 test questions that subsume the smaller question sets used previously (Clark et al., 2016; Khashabi et al., 2016). REGTSPERTD set, introduced by Khashabi et al. (2016), has 1,080 questions obtained by automatically perturbing incorrect answer choices for 108 New York Regents 4th grade science questions. We split this into 700 train and 380 test questions.

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

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

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

4.1 IR solver + ET

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

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

Table 4 validates this hypothesis, showing the result of incorporating ET in IR, as IR + ET($\xi = 0.36$), where ξ was selected by optimizing end-to-end performance on the training set. We observe a 5% boost in the score on REGTSPERTD, showing that incorporating the notion of essentiality makes

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

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

Dataset	Basic IR	IR + ET
REGENTS	59.11	**60.85**
AI2PUBLIC	57.90	**59.10**
REGTSPERTD	61.84	**66.84**

Table 4: Performance of the IR solver without (Basic IR) and with (IR + ET) essential terms. The numbers are solver scores (%) on the test sets of the three datasets.

the system more robust to perturbations.

Adding ET to IR also improves its performance on standard test sets. On the larger AI2PUBLIC question set, we see an improvement of 1.2%. On the smaller REGENTS set, introducing ET improves IRsolver's score by 1.74%, bringing it close to the state-of-the-art solver, TableILP, which achieves a score of 61.5%. This demonstrates that the notion of essential terms can be fruitfully exploited to improve QA systems.

4.2 TableILP solver + ET

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

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

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

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

While these generated questions should arguably remain correctly answerable by TableILP, we found that this is often not the case. To investigate this, we curate a small dataset Q_R with 12 questions (cf. Appendix C of the extended version (Khashabi et al., 2017)) on each of which, despite having the required knowledge and a plausible reasoning chain, TableILP fails.

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

Following the notation of Khashabi et al. (2016), let $x(q_l)$ be a binary variable that denotes whether or not the l-th term of the question is used in the final reasoning graph. We enforce that terms with essentiality score above a threshold ξ must be used: $x(q_l) = 1, \forall l \in \omega(\xi)$. Let TableILP+ET($\xi$) denote the resulting system which can now be used in a cascading architecture.

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

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

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

5 Conclusion

We introduced the concept of essential question terms and demonstrated its importance for question answering via two empirical findings: (a) humans becomes substantially worse at QA even when a few essential question terms are dropped, and (b) state-of-the-art QA systems can be improved by incorporating this notion. While text summarization has been studied before, questions have different characteristics, requiring new training data to learn a reliable model of essentiality. We introduced such a dataset and showed that our classifier trained on this dataset substantially outperforms several baselines in identifying and ranking question terms by the degree of essentiality.

Acknowledgments

The authors would like to thank Peter Clark, Oyvind Tafjord, and Peter Turney for valuable discussions and insights.

This work is supported by DARPA under agreement number FA8750-13-2-0008. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

References

P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics* 18(4):467–479.

K. W. Church and P. Hanks. 1989. Word association norms, mutual information and lexicography. In *27th Annual Meeting of the Association for Computational Linguistics*. pages 76–83.

P. 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.

P. Clark, O. Etzioni, T. Khot, A. Sabharwal, O. Tafjord, P. Turney, and D. Khashabi. 2016. Combining retrieval, statistics, and inference to answer elementary science questions. In *30th AAAI*.

J. Clarke and M. Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research* 31:399–429.

M.-C. De Marneffe, B. MacCartney, C. D. Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*. Genoa, volume 6, pages 449–454.

J. L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin* 76(5):378.

C. Hori and F. Sadaoki. 2004. Speech summarization: an approach through word extraction and a method for evaluation. *IEICE TRANSACTIONS on Information and Systems* 87(1):15–25.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *2014 EMNLP*. pages 523–533.

D. Howell. 2012. *Statistical methods for psychology*. Cengage Learning.

P. Jansen, R. Sharp, M. Surdeanu, and P. Clark. 2017. Framing qa as building and ranking intersentence answer justifications. *Computational Linguistics* .

T. Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. *Machine learning: ECML-98* pages 137–142.

D. Khashabi, T. Khot, A. Sabharwal, P. Clark, O. Etzioni, and D. Roth. 2016. Question answering via integer programming over semi-structured knowledge (extended version). In *Proc. 25th Int. Joint Conf. on Artificial Intelligence* (IJCAI).

D. Khashabi, T. Khot, A. Sabharwal, and D. Roth. 2017. Learning what is essential in questions (extended version).

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

K. Knight and D. Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence* 139(1):91–107.

P. Kordjamshidi, D. Khashabi, C. Christodoulopoulos, B. Mangipudi, S. Singh, and Dan Roth. 2016. Better call saul: Flexible programming for learning and inference in nlp. In *International Conference on Computational Linguistics (COLING)*.

P. Kordjamshidi, D. Roth, and H. Wu. 2015. Saul: Towards declarative learning based programming. In *Proc. 24th Int. Joint Conf. on Artificial Intelligence* (IJCAI).

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

F. Li, X. Zhang, J. Yuan, and X. Zhu. 2007. Classifying what-type questions by head noun tagging. In *Proc. 22nd Int. Conf. on Comput. Ling.* (COLING).

X. Li and D. Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '02, pages 1–7.

D. Moldovan, M. Paşca, S. Harabagiu, and M. Surdeanu. 2003. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems (TOIS)* 21(2):133–154.

J. H. Park and W. B. Croft. 2015. Using key concepts in a translation model for retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, pages 927–930.

M. Sammons, C. Christodoulopoulos, P. Kordjamshidi, D. Khashabi, Srikumar V., P. Vijayakumar, M. Bokhari, X. Wu, and D. Roth. 2016. Edison: Feature extraction for NLP, simplified. In *Proc. of the 10th International Conference on Language Resources and Evaluation* (LREC).

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

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2016. Convolutional neural networks vs. convolution kernels: Feature engineering for answer sentence reranking. In *HLT-NAACL*.

B. Wang, K. Liu, and J. Zhao. 2016. Inner attention based recurrent neural networks for answer selection. In *ACL*.

W.-t. Yih, X. He, and C. Meek. 2014. Semantic parsing for single-relation question answering. In *Proc. 52nd Annual Meeting of the Ass. for Comp. Linguistics* (ACL). pages 643–648.

W. Yin, S. Ebert, and H. Schütze. 2016. Attention-based convolutional neural network for machine comprehension. In *NAACL HCQA Workshop*.

A Features of the ET Classifier

Here we explain the structure of our classifier. In our design we give references to the NLP/Machine Learning frameworks we used to build the system, as well as a short account of our features.

In this work use the Saul framework (Kordjamshidi et al., 2015, 2016) [13] to our problem. This framework gives ability for easy NLP feature definitions and integrations with machine learning models. We use ILLINOISPIPLINE[14] to annotated raw documents with different views using such as Tokenizer (TOK), Lemmatizer (LEM), Part-of-Speech tagging (POS), Named-Entity-Recognition (NER), Stanford Dependency Parsing (DEP) (De Marneffe et al., 2006), and Chunker (CHK).

We include the details of features we used in our classifier. A majority of features are extracted via EDISON (Sammons et al., 2016), a library of feature extractor and group of hand-crafted features. Since EDISON already covers many standard features, our work here mostly has focused on selecting relevant ones, and creating their conjunction. These features cover a wide range of syntactic, semantic features and their combinations. In Figure 6 we show the frequency in which terms with certain POS tags labeled as essential or not. We provide some interesting statistics from distribution in Section 2.1. As it can be seen, POS labels alone are not good discriminants of the target labels, while their combinations with other labels (such as lemmas) would give more informative signal.

We have included the complete list of features in Table 5. For simplicity of notation, denote the surface string feature with SURF, define CONJ $\{.\}$ to be a feature resulted from the conjunction of the properties in its arguments. Also let BASE $\{.\}$ denote a simple baseline (and feature) returns most-popular labels for its observations; for example BASE $\{$LEM $\}$ is a classifier/feature which returns the most-frequent label given the lemma of the input constituent. Denote an arbitrary feature extractor with f_i when extracting the i-th terms; similarly f_{i-1}/f_{i+1} is the same feature extractor when applied to the previous/next term. With this definition we define a neighborhood of f_i, NEIGHBOR(f_i) be the collection of feature of the same type, applied on neighboring terms of the i-th term:

$$\begin{aligned}\text{NEIGHBOR}(f_i) = \{&f_i, f_{i-1}, f_{i+1}, f_{i-2}, f_{i+2},\\ &\text{CONJ}\{f_{i-1}, f_{i+1}\},\\ &\text{CONJ}\{f_{i-2}, f_{i-1}\},\\ &\text{CONJ}\{f_{i+1}, f_{i+2}\}\ \}\end{aligned}$$

All the EDISON features come with prefix `ed`. The details of these features and their implementation can be directly looked up from EDISON . Using such features is simply matter of calling the feature extractor function, on top of question annotations, which makes them easy to reproduce.

Some features are crafted specifically for the purpose of this problem. We collected a set of science terms frequently used in elementary school level textbooks. The feature ISSCIENCETERM$\{.\}$ checks whether the output of its input extractor is out science terms set or not. The feature MAXPMI /SUMPMI is the maximum/sum of the PMI value between the target question term, and the question options.

B Humans as reasoning engines

Here we include the results MTurk experiments we discussed in Section 2.2 (Figure 4). Here we include the whole table for completeness. In particular for each row we have included the precision. In addition the first row of this table is an experiment without dropping any terms, just to have a sense of how precision and recall would look like if no term was dropped. As it can be seen the precisions values are mostly close to each other, which shows that humans tend to keep their retain their precision across settings, by abstaining from answering questions they are not sure.

C Synthetic questions

Here we have included the synthetic dataset with discussed in Section 4.2. As mentioned these question are hand-made and perturbed versions of the existing question to trick the vanilla TableILP. Note that their design is done completely independent of the essentialityscores. The complete list of the questions are included in Table 7.

In the following two examples we take questions which TableILP answers correct for the right reason (hence we are sure that the solver has the right knowledge to answer it, as well as almost correct fuzzy matching scores). We change two

[13] Available at: https://github.com/CogComp/saul

[14] Available at: https://github.com/CogComp/cogcomp-nlp

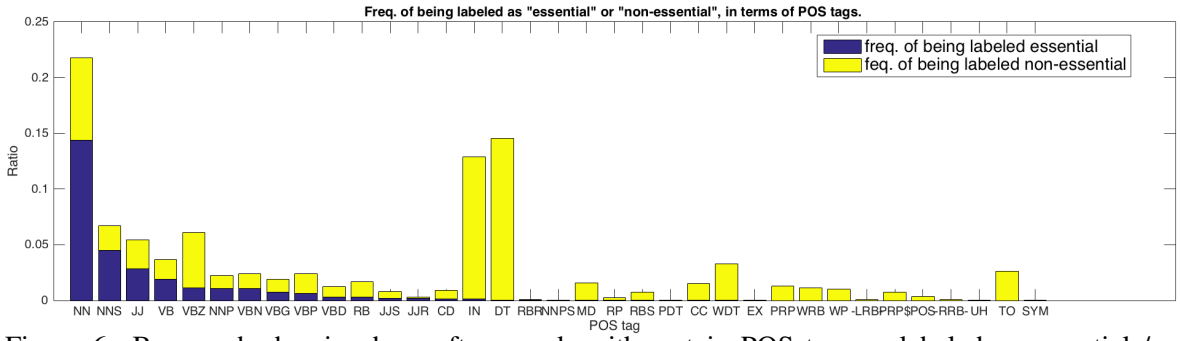

Figure 6: Bar graph showing how often words with certain POS tag are labeled as essential / non-essential.

Feature	Description
SURF LEM CONJ {POS, NER } CONJ {SURF, NER } CONJ {SURF, POS } CONJ {SURF, NER, POS } CONJ {LEM, POS } CHK	Basic features, which are directly extracted from the annotations of the question, i.e. surface string, NER, etc, and their conjunctions.
CONJ {ed.conflatedPos, LEM } CONJ {ed.conflatedPos, SURF } ed.deVerbalSuffix ed.gerundMarker ed.knownPrefixes ed.nominalizationMarker ed.numberNormalizer ed.deVerbalSuffix ed.prefixSuffixes ed.parsePath ed.brownClusterFeatures ed.dependencyPathUnigram ed.dependencyPathBigram ed.isItCapitalized ed.isItLastSentence ed.wordnetSynsetsFirstSense ed.wordnetSynsetsAllSenses ed.wordnetLexicographerFileNamesAllSenses ed.wordnetLexicographerFileNamesFirstSense ed.wordnetHypernymFirstSenseLexicographerFileNames ed.wordnetHypernymAllSensesLexicographerFileNames ed.wordnetPointersFirstSense ed.wordnetPointersAllSenses ed.wordnetSynonymsFirstSense ed.wordnetSynonymsAllSense CONJ {ed.wordnetExists, ed.wordnetSynsetsFirstSense, ed.wordnetLexicographerFileNamesFirstSense }	The features extracted by direct calls to EDISON feature-extraction library, on top of the basic annotations of the question. For example ed.brownClusterFeatures extracts the Brown representation for a given word, using its internal pre-extracted Brown representation. Another example is ed.wordnetSynsetsAllSenses, which given a term, returns all the synsets of that word.
ISSCIENCETERM{SURF } ISSCIENCETERM{LEM } SUMPMI MAXPMI NEIGHBOR(BASE {SURF }) NEIGHBOR(BASE {LEM }) NEIGHBOR(BASE {PAIR(SURF) }) NEIGHBOR(BASE {PAIR(LEM) }) NEIGHBOR(BASE {CONJ {POS, NER } }) NEIGHBOR(BASE {CONJ {POS, LEM } }) NEIGHBOR(BASE {CONJ {SURF, POS, LEM } })	We hand-craft these features based on previous features. For example ISSCIENCETERM{LEM } checkes whether lemma of a given word is in our collected science terms or not.

Table 5: List of important feature categories in our system.

	Setting	Threshold	Precision	Recall	Term-drop ratio	Test score (%)
	Nothing is dropped	–	0.868	0.970	0	0.85.02
(A) Gold Annotations	Drop terms with essentialityscore above certain threshold	0.2	0.689	0.08	0.35	28.51
		0.4	0.693	0.168	0.299	32.44
		0.6	0.804	0.3287	0.183	43.20
		0.8	0.812	0.328	0.17	43.43
		1.0	0.823	0.49	0.12	53.07
	Keep terms with essentialityscore above some threshold	0.0	0.813	0.854	0.604	73.08
		0.2	0.7857	0.805	0.676	68.125
		0.4	0.824	0.654	0.7543	62.53
		0.6	0.77	0.66	0.8	59.32
		0.8	–	–	–	–
(B) ETClassifier	Drop terms with essentialityscore above some threshold	0.0	0.7	0.017	0.873	25.76
		0.2	0.771	0.03	0.4342	26.56
		0.4	0.9066	0.0646	0.3331	29.24
		0.6	0.9024	0.255	0.176	41.63
		0.8	0.91	0.9364	0.0812	86.802
		1.0	0.9	0.98	0	88.7
	Keep terms with essentialityscore above some threshold	0.0	0.932	0.746	0.5166	75.87
		0.2	0.937	0.803	0.5166	80.16
		0.4	0.923	0.779	0.581	77.42
		0.6	0.91	0.686	0.68	70.27
		0.8	0.538	0.02	0.87	25.57
		1.0	0.57	0.04	0.88	26.28

Table 6: This table contains the exact numbers when using essentiality scores for dropping most important/least important terms in the question (Section 2.2). The setting (A) is when the gold annotations are used and setting (B) is when real-valued scores of the trained classifier are used. Precision is ratio of the questions answered correctly, if they answered at all. Recall is the ratio of the times a question is answered. Term-drop ratio is ratio of the terms dropped in the question sentence (compared to the the overall length of the question).

Question	Key
Which item causes objects to roll downhill? (A) gravity(B) friction(C) erosion(D) magnetism	A
Which sense is used to determine a sweet object's texture? (A) hearing(B) smell(C) taste(D) touch	D
An animal grows thicker hair as a season changes. This adaptation helps to (A) find food(B) keep warmer(C) grow stronger(D) escape from predators	B
What is the main energy for the water cycle? (A) electricity(B) erosion(C) gravity(D) sunlight	D
Which human activity is most damaging to the environment? (A) swimming in a lake(B) riding a bicycle(C) cutting down a rain forest(D) using solar energy	C
Which force washes away coastal soil? (A) gravity(B) friction(C) erosion(D) magnetism	C
Which unit of measurement describes an object's size? (A) meter(B) kilogram(C) liter(D) degree	A
Which form of energy is found in food? (A) chemical(B) electrical(C) sound(D) mechanical	B
Which object is nonliving? (A) bear(B) bicycle(C) bird(D) butterfly	B
Which sense is used to determine a colorful object's texture? (A) sight (B) smell(C) taste(D) touch	D
A mamal grows thicker hair as a season changes. This adaptation helps to (A) find food(B) keep warmer(C) grow stronger(D) escape from predators	B
A mammal shivers when it's cold. This adaptation is to (A) find food (B) get warmer body (C) grow stronger(D) escape from predators	B

Table 7: List of the synthesized question for Section 4.3. These question are hand-made and perturbed versions of the existing question to trick the vanilla TableILP. The design of these questions is done completely independent of the essentiality scores.

	Correctly answered by TableILP; turned incorrect in CASCADES	Inocorrect answered by TableILP; turned correct in CASCADES
Correct reasoning	5	4
Partially correct	7	7
Got lucky	11	2
Sum	23	13

Table 8: Breakdown of the predictions changed from adding TableILP to CASCADES, classified according to their reasons, annotated manually by inspecting the reasoning graph of each question.

words, while retaining the general meaning of the questions, but tricking the solver into answering it incorrectly.

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

Original question: Which force causes rocks to roll downhill? (A) gravity(B) friction(C) erosion(D) magnetism
Generated question: Which item causes objects to roll downhill? (A) gravity(B) friction(C) erosion(D) magnetism

D Example predictions: before and after adding ET

Here in Table 9 we show two sample questions for the analysis done in Section 4.3 and in Table 8. In the first row we have a question which is correct been answered by TableILP , but for an incorrect reason (since it's using only "plan" to answer a question about "gasses used in photosynthesis"). Augmenting the solver with ET, the prediction turns into an incorrect one, mostly probably due to an inaccurate alignment scores. One can also blame low ET score of "use" which is about 0.4.

In the second row we have an incorrect answer by TableILPwhich is turned into a correct prediction by adding ET, and the reasoning chain is mostly correct except a few noisy edges (e.g. "flag"-"lift objects" edge).

E Error Analysis for TableILP+ET

We manually analyzed the predictions of TableILP vs. CASCADES$(0.4, 0.6, 0.8, 1.0)$ on the *training* set of AI2PUBLIC (the test questions remain hidden). Out of 432 questions in total, the two systems differ on 57 questions. Out of these, 23 correct ones are turned into incorrect, and 13 incorrect predictions are turned correct, as a result of adding ET to TableILP. Hence, the overall performance drops.

One annotator went through the reasoning graphs of these questions and classified them into multiple groups base on how convincing the reasoning used by the solver was. Table 8 shows breakdown of these questions and their classes. The row "correct alignment" is when the required knowledge is there and it's used correctly in the reasoning graph. The row "partially correct" is when there are some noisy alignments as well as the correct alignment. The row "got lucky" are correct predictions, but for wrong reasons: not having the necessary knowledge, not performing the natural reasoning, etc.

As can be seen, among the prediction changes, size of the resulted correct predictions are more reliable than the questions that were correct but turned incorrect after adding ET. In other words, the drop we see in performance of TableILP after adding ET is due to existence of other inaccurate solver ingredients. In Appendix IV, we provide a visualization of two questions for each of these categories to give more intuition to the reader.

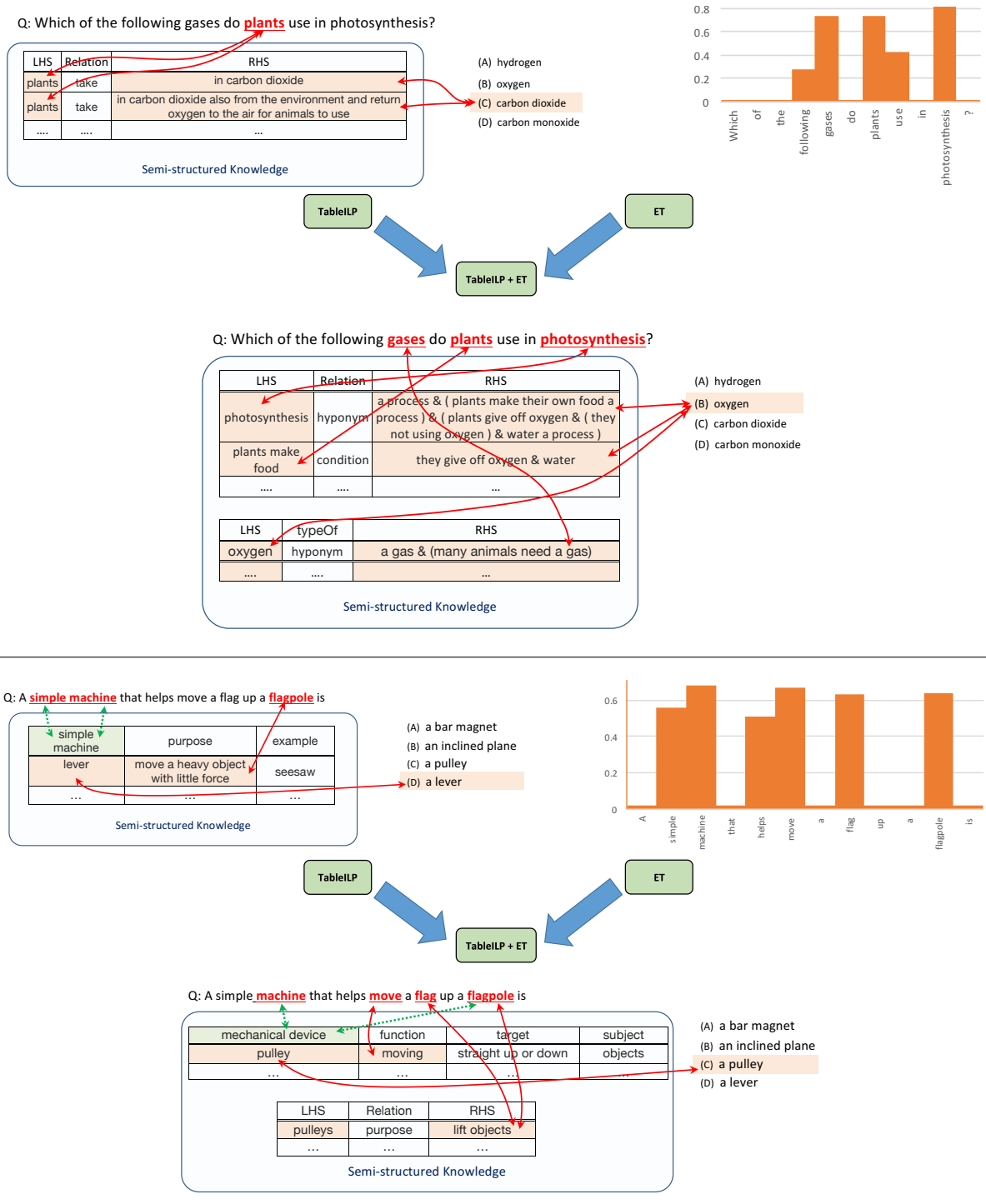

Table 9: Comparison of the reasoning graphs, for TableILP and CASCADES(TableILP+ET). In the first row, adding ET changes the correct prediction to incorrect, but in the second row, it corrects the incorrect prediction.