Parsing to Programs: A Framework for Situated QA

A System for Solving Pre-University Level Newtonian Physics Problems

Mrinmaya Sachan and Eric P. Xing
Carnegie Mellon University
{mrinmays,epxing}@cs.cmu.edu

ABSTRACT

This paper introduces *Parsing to Programs*, a framework that combines ideas from parsing and probabilistic programming for situated question answering. As a case study, we build a system that solves pre-university level Newtonian physics questions. Our approach represents domain knowledge of Newtonian physics as programs. When presented with a novel question, the system learns a formal representation of the question by combining interpretations from the question text and any associated diagram. Finally, the system uses this formal representation to solve the questions using the domain knowledge. We collect a new dataset of Newtonian physics questions from a number of textbooks and use it to train our system. The system achieves near human performance on held-out textbook questions and section 1 of AP Physics C mechanics – both on practice questions as well as on freely available actual exams held in 1998 and 2012.

CCS CONCEPTS

• **Information systems** → **Data mining**; Data analytics; Expert systems; • **Computing methodologies** → *Machine learning*;

KEYWORDS

Question Answering, Expert Systems, Semantic Parsing;Diagram Parsing

ACM Reference Format:
Mrinmaya Sachan and Eric P. Xing. 2018. Parsing to Programs: A Framework for Situated QA: A System for Solving Pre-University Level Newtonian Physics Problems. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3219972

1 INTRODUCTION

A number of AI challenges have been proposed and attempted in the recent years. AI systems can now compete against professional human players at games such as *Chess* [9], *Shogi* [1] and *Go* [59]. AI can now also compete against human champions in quiz shows [21] and trivia games [8]. One such ambitious goal is to build AI systems that can pass school or university exams [4, 7, 13, 23]. Given the aforementioned marquee successes of AI, we might be led to believe that building AI systems that can pass these exams would be easy. However, this is not the case. In the recently concluded 8^{th} grade science question answering challenge[1] conducted by the *Allen Institute for AI*, the best performing submission had an accuracy of less than 60% on a four choice question answering task. An AI system that can pass these examinations must possess the required domain knowledge of the subject, be able to accurately process and represent the (textual and possibly visual) information presented in the questions and be able to perform deep causal or counter factual reasoning based on the domain knowledge.

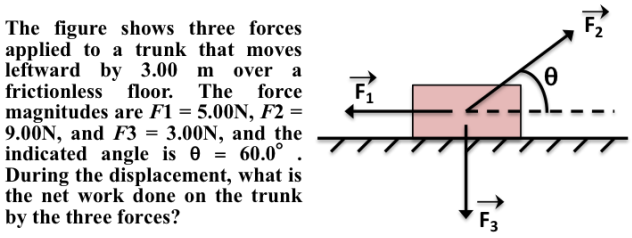

Figure 1: A sample question from our dataset.

This paper presents an approach to solve pre-university level Newtonian physics questions such as the one shown in Figure 1. These questions typically consist of a paragraph size piece of text describing the question and (often) an associated diagram. An AI system that can answer these questions must be able to interpret both the question text as well as the associated diagram. These questions are quite diverse, offering a number of challenges. The questions text is usually ambiguous, posing a number of NLP challenges. The diagrams associated with these questions represent complex physical and mathematical concepts; not often represented in natural images. Hence, traditional computer vision technology cannot easily extract and represent the semantics of these diagrams. Finally, solving these questions requires the system to reason with domain knowledge of Physics (axioms, theorems, etc.). This knowledge must be provided to the system in a form that it be used to reason and solve these questions.

Our system is provided with structured domain knowledge of Newtonian physics in the forms of programs. Given this domain knowledge, our approach answers the questions in two stages: (1) *Question Parsing* – which parses the question text and any associated diagram and represents it as a (weighted) logical expression, and (2) *Programmatic Solving* – which given an encoding of the

[1] https://www.kaggle.com/c/the-allen-ai-science-challenge

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219972

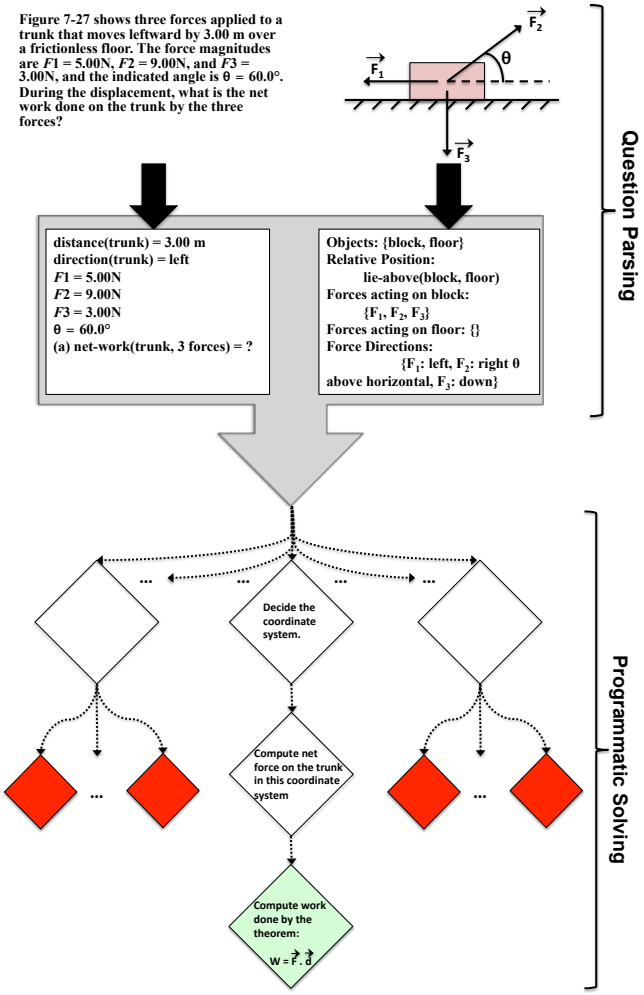

Figure 2: An illustrative representation of our approach on the sample question. The approach solves the question in two stages. The first stage, *Question Parsing*, parses the question text and any associated diagram into an equivalent (weighted) logical expression in a typed first-order logic language. The logical expression for this example is shown in the two rectangular white boxes. Ideally, each literal in this expression is weighted. However, the weights are not shown in this figure for simplicity. The second stage, *Programmatic Solving* takes this formal representation of the question and solves it by performing a (probabilistic) search over a set of pre-defined programs. One of the paths in the search (which corresponds to the process required to solve the question) leads to the solution (shown in green), whereas others do not lead to any solution and are rejected (shown in red).

domain knowledge in the form of programs and the logical expression representing the question, searches for a set of programs which be can executed to obtain the answer of the question. Figure 2 pictorially describes this process for the sample question. The *Question Parsing* stage maps the question (question text as well as any associated diagram) to a formal representation. This is achieved by generating a weighted first-order logic formula (a conjunction

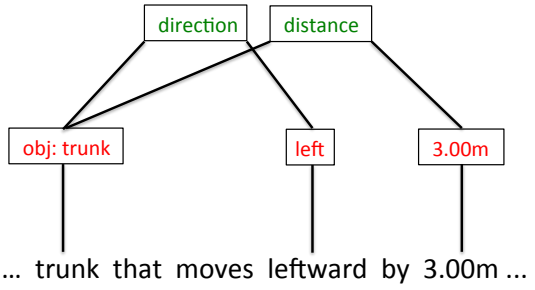

Figure 3: Hypergraph representation of the phrase "trunk that moves leftward by 3.00m". Our approach maps the words "trunk", "leftward" and "3.00m" to corresponding concepts (shown in red). Then, it identifies relations among these concepts ("distance" and "direction" shown in green), leading to the parse: direction(trunk, left) ∧ distance(trunk, 3.00m).

of literals) that corresponds to the question and associating a confidence score with each literal. The *Programmatic Solving* stage takes this formal representation of the question and solves it by performing a probabilistic search over the set of programs.

We collect a large dataset of question-answer pairs from physics textbooks widely used by students in India. We evaluate our system on a held out dataset of questions from these textbooks and on practice and actual questions from the AP physics C mechanics exam. Our system achieves an overall accuracy of 68% on held-out questions from textbooks. In contrast, a small user study conducted by us on 10 students studying physics found that the average student score was only around 63%. On the AP Physics C mechanics practice and actual exams (1998 and 2012), our system correctly answered 50%, 42% and 54% of the questions, respectively. These scores are close to the average human performance on these exams. We plan to release our system for public consumption on platforms such as MOOCs. Our system can potentially be a used to help students learn Newtonian physics on these platforms. We provide some small user studies to back this claim.

2 FORMAL LANGUAGE

We choose our logical language as a subset of typed first-order logic comprising of *constants* (3.00 m, 5.00 N, 60°, etc.), *variables* (F_1, θ, etc.), and a hand-picked set of 138 predicates (*equals*, *mass*, *distance*, *force*, *speed*, *velocity*, *work*, etc.). Each element in the language is strongly typed. For example, the predicate *mass* takes the first argument of the type "object" and the second argument of the type "mass-quantity" such as *2kg*, *3g*, etc. As shown in Figure 2, the question text and diagram parses are represented as logical formulas over constants, variables, existential quantifiers and conjunctions over applications of predicates to a set of arguments.

3 QUESTION PARSING

The *Question Parsing* stage maps the question text and the associated diagram into the formal representation. It comprises of two components: *Text Parsing* and *Diagram Parsing*. We describe the two components below:

3.1 Text Parsing

As the first step of question parsing, we introduce our text parsing approach which learns to map words or phrases in the question text to weighted formulas over the logical language.

Supervision: For building a question text parser, we assume supervision in the form of questions paired with their formal representations. In other words, our model is provided with logical formulas for training questions such as the logical formula for the question text shown in Figure 2. This is used to learn a question text parser as described below:

Learning: We propose a two-stage statistical model for question text parsing. This process is pictorially shown in Figure 3 which represents a logical formula as a hypergraph [22, 31]. Each node in the graph corresponds to a concept in the logical language (i.e. constants, variables, functions, or predicates). The edges capture relations between concepts; concept nodes are connected if one concept is an argument of the other in the logical language. In order to interpret the question text, the question parser first identifies concepts evoked by the words or phrases in the input text. Then, it identifies relations over the identified concepts, provided some type constraints for the arguments are satisfied. For instance, the *distance* relation in Figure 3 must take a constant which has the type *length* such as 3.00m as the second argument. Similarly, the *direction* relation in the Figure 3 must take one of the constants among $\{left, right, up, down\}$ as the second argument. We choose a part-based log-linear model for parsing. Let $p = \{p_1, p_2\}$ where p_1 denotes the concepts identified in p and p_2 denotes the identified relations. The log-linear model factorizes into two components for concept and relation identification:

$$P(p|t;\boldsymbol{\theta}) = \frac{1}{Z(t;\boldsymbol{\theta})} \exp\left(\boldsymbol{\theta}^T \boldsymbol{\phi}(p,t)\right)$$
$$where, \quad \boldsymbol{\theta}^T \boldsymbol{\phi}(p,t) = \boldsymbol{\theta}_1^T \boldsymbol{\phi_1}(p_1,t) + \boldsymbol{\theta}_2^T \boldsymbol{\phi}_2(p_2,t)$$

Here, $Z(t;\boldsymbol{\theta})$ is the partition function of the log-linear model and $\boldsymbol{\phi}$ is the concatenation $[\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2]$. Search for the highest scoring latent parse is NP hard. Hence, we use beam search with a fixed beam size (10) for inference. We first infer p_1 to identify a beam of concepts. Then, we infer p_2 to identify relations among candidate concepts, ensuring that the type constraints for induced predicate arguments are satisfied. We find the optimal parameters $\boldsymbol{\theta}^*$ using maximum-likelihood estimation with L2 regularization:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_{(t,p)\in Train} \log P(p|t;\boldsymbol{\theta}) - \lambda ||\boldsymbol{\theta}||_2^2$$

We use L-BFGS to optimize the above objective. Table 1 describes the features used in our parser – which are inspired from some previous work on solving geometry problems [57].

As a post-processing step, we build a simple rule-based math parser to handle mathematical formulas and equations. This parser takes in a math expression such as $F = ma$ and parses it into our formal representation "equals(F, prod(m, a))". As another post procession step, we identify anaphoric and coreferential expressions in the text using Stanford CoreNLP and replace all coreferential expressions in the learnt logical formula with their corresponding antecedents.

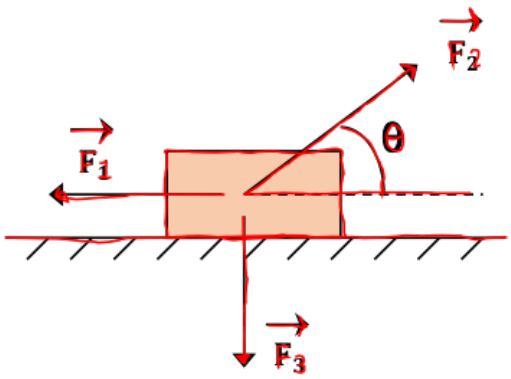

Figure 4: Diagram Annotation (phase 1) for our sample question. We annotate object elements (blocks, wedges, etc.), textual elements (force, velocity, etc.), low-level diagrammatic elements (lines, arrowheads, etc.), high-level diagrammatic elements (axes, blocks, pulleys, etc.), plots and any other decorative elements using a web-based annotation interface.

3.2 Diagram Parsing

Pre-university physics questions are often accompanied by diagrams. In our datasets, about 20-30 percent of the questions are accompanied by diagrams. These diagrams are pretty complex and diverse. The diagrams typically have a large number of object categories and depict complex higher-order physical phenomena (such as force, velocity, acceleration, etc.) which goes well beyond what natural images usually convey. The diagrams typically include object elements (e.g. drawings of blocks, wedges, etc.), textual elements (e.g. annotations of force, velocity, acceleration, etc.), low-level diagrammatic elements (e.g. arrowheads, lines, etc.), high-level diagrammatic elements (e.g. axes, blocks, pulleys, etc.), plots and possibly other decorative elements. All these elements of the diagram must be recognized and organized in context to achieve a complete logical representation of the diagram. We used a pipeline approach for recognizing various diagram elements. This approach is inspired by [30] who worked on food-web diagrams.

Supervision: Annotating diagrams with the rich information: object elements, textual elements, diagrammatic and decorative elements, and correspondences between the diagram elements is a complicated task. Thus we divided the annotation into three phases. In the first phase, we annotated various diagram elements using a web-based annotation interface as shown in Figure 4. In the second phase, we annotated label associations. For example, the force $\vec{F}_1$ refers to the leftward arrow in Figure 4. In the third phase, we annotated objects and mapped them to an object category mentioned in the question text.

Detecting Low-level Diagrammatic Elements: Physics diagrams have a number of diagrammatic elements such as arrows, lines, arcs, etc. We detected diagrammatic elements in three steps: In the first step, we applied a weak Gaussian blur on the raw image and then binarized it using a threshold selection method proposed in [41]. Then, we over-generated a large number of candidate diagrammatic element proposals using the boundary detection and grouping method from [32], Hough transforms [18] and by detecting parallel curved edge segments in a canny edge map. In the third

ϕ_1	Lexicon Map	Indicator that the word or phrase maps to a predicate in a lexicon created by us. We derive correspondences between words/phrases and keywords and concepts in the logical language using manual annotations in the training data. For instance, our lexicon contains ("direction:left", *left*, *leftward*) including all possible realizations for the concept "direction:left".
	Regex for constants and explicit variables	Indicator that the word or phrase satisfies a regular expression to detect numbers or explicit variables (e.g. "3.00m", "2gm", "g m/s^2"). These regular expressions were built as a part of our system.
ϕ_2	Dependency tree distance	Shortest distance between the words of the concept nodes in the dependency tree. We use indicator features for distances of -3 to 3. Positive distance shows if the child word is at the right of the parent in the sentence, and negative otherwise.
	Word distance	Distance between the words of the concept nodes in the sentence.
	Dependency edge	Indicator functions for outgoing edges of the parent and child for the shortest path between them.
	Part of speech tag	Indicator functions for the POS tags of the parent and the child
	Relation type	Indicator functions for unary / binary parent and child nodes.
	Return type	Indicator functions for the return types of the parent and the child nodes. For example, return type of *Equals* is boolean, and that of *Distance* is length.

Table 1: The feature set for our text parsing model. We use Turboparser[39] for POS and syntactic information.

step, we recursively merged proposals that exhibit a low residual when fit to a 1st or a 2nd degree polynomial. We then trained a 2 class CNN resembling VGG-16 [60], with a fourth channel appended to the standard 3 channel RGB input. This fourth channel specifies the location of the diagrammatic element smoothed with a Gaussian kernel of width 5.

Detecting High-level Diagrammatic Elements: Sometimes, we have to further assemble these low level diagrammatic elements (such as lines and arcs) to higher level diagrammatic elements (such as axes, blocks, wedges, pulleys, etc.) For this, we used Harris corner detectors [26] to identify possible locations of intersections of lines. Finally, we merged lines that form higher level diagrammatic elements using combinatorial grouping [43] with a set of manually curated grouping rules for grouping each diagrammatic element.

Detecting Textual Elements: The problem of recognizing text in images is an old one and has led to an entire research area of optical character recognition (OCR) [28]. Most existing OCR systems are trained on character images, scanned documents and scenes and hence, do not work well on diagrams from textbooks. To detect text labelings in diagrams, we used an off-the-shelf OCR system – *Tesseract*[2] as a baseline model. However, since many textual elements are heavily structured (these include elements in vector notation (e.g. $\vec{F}$), greek alphabets (e.g. θ), physical quantities (e.g. 2 m/s)) and are usually longer than a single character, we improved the detection by training a text localizer using a CNN having the same architecture as AlexNet [35]. We used the Chars74K dataset [16], a dataset obtained from vector PDFs of a number of physics textbooks and a set of synthetic renderings of structured textual elements generated by us as training data.

Detecting Label associations: Often, diagram labels such as $\vec{F}_1$, $\vec{F}_2$, $\vec{F}_3$ refer to low-level diagrammatic elements such as the three arrows in Figure 2. As a final step in diagrammatic element recognition, we mapped textual element labels with diagrammatic elements. Given the set of textual elements and the set of diagrammatic elements, we solved a label association problem that maps textual elements with diagrammatic elements. We built a binary logistic regression classifier over pairs of textual elements and low-level diagrammatic elements. The probability that a textual element t_i maps to a low-level diagrammatic element d_j is given by the standard logistic form: $p_{ij} = P(y_{ij}|t_i, d_j; \mathbf{w}) = \left(1 + e^{-\mathbf{w}.\mathbf{f}(t_i, d_j)}\right)^{-1}$. Finally, the textual element t_i is mapped to a single low-level diagrammatic element $\hat{d}_j$ by solving the classic bipartite graph matching ILP: $\max \sum_{i,j} p_{ij} y_{ij} \;\; s.t. \sum_j y_{ij} = 1 \;\; \forall i$. We used a small set of hand engineered features based on distance, shape, size and orientation of elements and labels.

Foreground Detection: Foreground detection [19] is an old computer vision technique which extracts the foreground of an image for further processing. For foreground detection, we classified every pixel in the diagram as a foreground vs background pixel. To achieve this, we built nonparametric kernel density estimates [42, 47] in RGB, texture and entropy spaces. We used these estimates as features for object detection.

Detecting Blobs and Objects: Blob detection [38] is another classical computer vision problem for detecting blobs or regions in an image where some properties such as brightness or color are constant compared to surrounding regions. We tried several object proposal approaches in literature like window classification [62], perceptual grouping [10, 25], cascaded ranking svm [66], objectness [2], selective search [61], global and local search [45] and edge boxes [67]. However, as all these approaches were developed for natural images, we found that these did not work well for diagrams. To tackle this, we used the foreground probability map described above to detect a set of blob segments produced by a set of classifiers with features capturing location, size, central and Hu moments, etc. These segment proposals were then combined with multi-scale combinatorial grouping [43], a recent grouping strategy, to achieve a ranked list of object proposals with corresponding confidence score from the model. Next, these object proposal scores were combined (by learning a linear interpolant on the dev set) with the scores from a discriminatively trained part-based model [20] trained on physics questions for object detection that focus on the detection of manually selected list of objects commonly seen in

[2]https://github.com/tesseract-ocr/tesseract

physics diagrams (blocks, pulleys, etc.) to achieve a ranked list of object proposals and corresponding confidence scores.

Incorporating Question Text for Object Detection: Often, the corresponding question texts provide important cues for detecting these visual elements. For example, the question in Figure 2 mentions the object 'trunk'. While it is unlikely that the object recognition component will correctly recognize the object 'trunk' as 'trunk' doesn't appear again in the training dataset as an object, the mention of the noun phrase 'trunk' in the question text and the context in which it appears is an important cue for identifying that this object is 'trunk'. Hence, we built a text-based object detector that uses logistic regression to classify each noun phrase in the question text as an object or not. We used a small set of manually engineered features for the prediction problem: (a) if the noun phrase is included in a list of objects manually built by us by looking at the dev set, (b) if the noun phrase is an object category in ImageNet, and (c) if the noun phrase is the agent/patient (determined using the Turbo dependency parser[39]) of a small list of actions taking place in our dev set (e.g. pull, run, hit ...).

Mapping Blobs and Objects to Object Labels: Finally, our system maps an object category detected by our diagram-based blob/object detector to an object category detected by the text-based object detector. For this, we built a binary logistic regression classifier over pairs of object categories and set of blobs/objects. The probability that an object category c_i maps to a blob/object o_j is again given by the standard logistic form described earlier. The object category $\hat{c}_i$ is mapped to the most probable blob/object o_j using the bipartite graph matching ILP described before. We used a small set of hand engineered features based on (a) cosine similarity of proposal label of the blob/object and the object category, (b) composition of diagrammatic and textual elements recognized in the image which overlap with the object and context words (we use a context of 2 words on the left and 5 words on the right), and (c) average image similarity (we use mutual information) between top 10 images retrieved on Google by querying for the object category and the blob/object.

Detecting Plot elements: A significant proportion of diagrams are plots (graphs, charts, etc.). Hence, to solve questions with plots, we need to be able to extract information from the plots and associate the information with corresponding legend entries. These plots usually have high variations and often are accompanied with heavy clutter and deformation. We used FigureSeer [58] to parse the plots and generate a structured tabular representation of the plot information.

From Diagram Elements to Literals: Finally, we used the various element detectors along with corresponding detection scores to obtain the formal interpretation for the diagram. We used a set of rules – one for every predicate. The rules decide if the predicate holds for a set of diagram elements which are type consistent with the arguments of the predicate. For example, we have rules for listing objects, relative position of objects, forces acting on objects, force directions, etc. which use the parsed diagram information.

3.3 Reconciling Text and Diagram Parsing

Given the set of literals $L_{text}, L_{diagram}$ and confidence scores given by text and diagram parsing, we wish to technique the best logical formula (i.e. subset of literals) representing the question. We use a technique by [57] who propose a sub-modular objective that scores a subset of literals using the text and diagram parses in their work on geometry question answering. We summarize it below:

The best subset of literals is defined as the subset that has a high affinity with the text t and diagram d and does not suffer from redundancy. Let $\mathcal{A}(L, t, d)$ measure the affinity of literals in L with both the text and the diagram, $C(L, t, d)$ measure the coverage of literals in L compared to the text which discourages redundancies. Let λ be a trade-off parameter between $\mathcal{A}$ and C. Thus, the objective for the best set of literals L^* among set of all possible literals $\mathcal{L}$ can be defined as:

$$L^* = \arg\max_{L \subset \mathcal{L}} \lambda \mathcal{A}(L, t, d) + (1 - \lambda) C(L, t, d)$$

The affinity $\mathcal{A}$ decomposes into text and diagram affinities $\mathcal{A}_{text}$ and $\mathcal{A}_{diagram}$: $\mathcal{A}(L, t, d) = \sum_{l_j \in L} \left[\mathcal{A}_{text}(l_j, t) + \mathcal{A}_{diagram}(l_j, d)\right]$. We substitute scores for literals provided by the text and diagram parsers for text and diagram affinities. The coherence function $C(L, t, d)$ is defined as $N_{covered}(L) - R_{redundant}(L)$. Here $N_{covered}$ is the number of concept nodes used by the literals in L, and $N_{redundant}$ is the number of redundancies among the concept nodes of the literals. We can prove that the objective $\lambda\mathcal{A}(L, t, d) + (1 - \lambda)C(L, t, d)$ is sub-modular by proving that both $\mathcal{A}(L, t, d)$ and $C(L, t, d)$ are sub-modular:

(1) Sub-modularity of $\mathcal{A}(L, t, d)$: Consider a set of literals $L \subset \mathcal{L}$ and an additional literal $l_j \in \mathcal{L} \backslash L$. Let $L' \subset L$ be a subset of L. Our definition of $\mathcal{A}(L, t, d)$ is linear in the literals. Hence, $\mathcal{A}(L' \cup \{l_j\}, t, d) - \mathcal{A}(L', t, d) = \mathcal{A}(L \cup \{l_j\}, t, d) - \mathcal{A}(L, t, d)$. Hence, $\mathcal{A}(L, t, d)$ is sub-modular. □

(2) Sub-modularity of $C(L, t, d)$: Note that the coherence function $C(L, t, d)$ is defined as $N_{covered}(L) - R_{redundant}(L)$. We show that $C(L, t, d)$ is sub-modular by showing that $N_{covered}(L)$ and $-R_{redundant}(L)$ are sub-modular.

Again, we consider a set of literals $L \subset \mathcal{L}$ and an additional literal $l_j \in \mathcal{L} \backslash L$. Let $L' \subset L$ be a subset of L. Also, let K and K' denote the sets of concepts covered by L and L'. Let k_j denote the set of concepts covered by l_j.

(2a) Sub-modularity of $N_{covered}(L)$: Clearly $K' \subset K$. Thus, from set theory $|K \cup k_j| - |K| \leq |K' \cup k_j| - |K'|$. This implies that $N_{covered}(L \cup \{l_j\}) - N_{covered}(L) \leq N_{covered}(L' \cup \{l_j\}) - N_{covered}(L')$. Hence, $N_{covered}(L)$ is sub-modular. □

(2b) Sub-modularity of $-R_{redundant}(L)$: Observe that $R_{redundant}(L \cup \{l_j\}) - R_{redundant}(L) = |K \cap k_j|$. Since $K' \subset K$, $|K' \cap k_j| \leq |K \cap k_j|$. Hence, $R_{redundant}(L' \cup \{l_j\}) - R_{redundant}(L') \leq R_{redundant}(L \cup \{l_j\}) - R_{redundant}(L)$. By negating this statement, we get that $-R_{redundant}(L)$ is sub-modular. □

As the objective is sub-modular, we can come up with a $1 - \frac{1}{e}$ approximation greedy method [33] to maximize the objective. The method maximizes the objective by starting from an empty set of literals and adding the next literal l_j that maximizes the gain of the objective function until the gain becomes negative. Algorithm 1 describes the procedure in detail.

Algorithm 1: Sub-modular objective maximization for reconciling question text and diagram parsing

Data: $(L_{text}, \mathcal{A}_{text})$ and $(L_{diagram}, \mathcal{A}_{diagram}) \leftarrow$ outputs from text and diagram parsing respectively, objective $\mathcal{F} = \lambda\mathcal{A} + (1-\lambda)\mathcal{C}$, set of possible literals $\mathcal{L}$.

Result: $L^* \leftarrow$ Greedy maximization.

Initialization: $L \leftarrow \{\}$;

Do

  Greedy addition: add (L, l_j) s.t. $l_j = \arg\max_{l_j \in \mathcal{L} \setminus L} \mathcal{F}(L \cup \{l_j\}) - \mathcal{F}(L)$

while *gain is positive*;

```
def vector_addition(Vectors vectors):
    result = zero_vector()
    for vector in vectors:
        result = result + vector
    return result

def angle_bw_vectors(Vector vec1, Vector vec2):
    return cos_inv(dot(vec1, vec2)/(norm(vec1)*norm(vec2)))

def project_vector(Vector vec, Direction theta):
    return (vec*cos(theta), vec*sin(theta))

def implicit_g_force(Mass m, Forces forces):
    if not forces.contains((“-mg i + 0 j”)):
        forces.append((“mg i + 0 j”))

def Newton_II_law(Mass m, Forces forces, Accelerations accs):
    net_force = vector_additon(forces)
    net_acceleration = vector_addition(accs)
    return Constraint(net_force = m * net_acceleration)

def conservation_of_momentum(Mass m1, Velocity v1_initial, Mass m2, Velocity v2_initial, Velocity v1_final, Velocity v2_final):
    preconditions = [external_force_on_system() == None]
    if preconditions:
        return Constraint(m1*v1_initial+m2*v2_initial = m1*v1_final+m2*v2_final)
```

Figure 5: Example programs used by our approach.

4 DOMAIN THEORY AS PROGRAMS

Subject knowledge of Newtonian physics is a crucial component in our solver. We presented the domain knowledge to the system in the form of structured programs. Some example programs are shown in Figure 5.

Some of these programs perform basic functions such as vector addition, computing angle between vectors, unit conversion, etc. Others perform more complex functions such as applying Newton's laws of motion or conservation of momentum, etc. A number of axioms denote laws of physics as a mathematical expression. For example, the Newton's second law is expressed simply as $\vec{F}_{net} = m \times \vec{a}$. Here $\vec{F}_{net}$ stands for the vector quantity representing the net force on a body. m stands for the mass of the body and $\vec{a}$ stands for the acceleration of the body. These programs also define a set of preconditions which must be satisfied for it to be executable. When the preconditions are satisfied, the programs define the mathematical expression as a constraint on the output. These constraints are then solved to obtain the answer. We manually wrote a total of 237 such programs. Let $\mathcal{P}$ represent this set of programs. We use this set of programs to answer the physics problems via the following deductive solver.

5 THE DEDUCTIVE SOLVER

Given access to the domain theory, we solve the physics problem by searching for program applications that can lead to the problem solution. We use a forward chaining search procedure exploring various possible program applications. Algorithm 2 describes the procedure.

Algorithm 2: Our Forward Chaining deductive solver

Data: Weighted set of literals L representing the question and Domain knowledge $\mathcal{P}$.

Do

  1. **Match Programs:** Match the pre-conditions of the programs against the set of literals i.e. find all programs $p \in \mathcal{P}$ s.t. the precondition p^{pr} can be unified with some set of literals L.
  2. **Select Program:** Sample a program (randomly uniformly) among the matching programs. Stop if no program can be applied.
  3. **Apply Program:** Apply the chosen program by adding the result to the set of literals/constraint set.

while $\#iterations < N_{upperbound}$;

We score the program applications as a function of the scores of various literals in the program's precondition. The score of a literal is given by the confidence score from the question parser. In case it is a derived literal (derived by an earlier program execution), its score is given by the function value of the program application that derived it. We explored various scoring functions: minimum, arithmetic mean, geometric mean and harmonic mean of all literal scores. We found that taking the harmonic mean of the precondition literals performed the best. Hence, we use harmonic mean of precondition literals as the scoring function in our system. We used an off-the shelf library[3] to solve the constraints introduced by the programs. Then, the following answering interface uses the search results to answer the question.

Handling Various Question and Answer Types: The physics examinations consist of a number of question and answer types. While a majority of questions directly ask about a particular physical quantity, there are a substantial number of questions which do not fit in this paradigm. For example, there are some *which of these are not true*, *select the odd one out*, *match the following* questions. To handle a variety of questions, we build an answering interface. The interface calls the deductive solver described above and answers the question based on the type of the question or the kind of answer sought.

6 EXPERIMENTS

6.1 Datasets

We validated our system on a dataset of physics questions taken from popular pre-university physics textbooks and few AP Physics C: Mechanics courses. We trained our model on physics questions taken from three popular pre-university physics textbooks: *Resnick Halliday Walker*, *D. B. Singh* and *NCERT*. Millions of students in

[3] http://docs.sympy.org/dev/modules/solvers/solvers.html#sympy.solvers.solvers.nsolve

Edge Boxes	*O.S.-Text*	*O.S.*
24.7	62.2	**69.8**

Table 2: Jaccard similarity between detected diagram elements and gold elements for Edge Boxes and two versions of our system's (O.S.) diagram parser: our parser which does not use text information and our full parser.

India study physics from these books every year and these books are available online. We manually identified chapters relevant for Newtonian physics in these textbooks and took all the exercise questions provided in these chapters. This resulted into a dataset of 4941 questions, out of which 1019 had associated diagrams. We partitioned the dataset into a random split of 1000 training, 500 dev and 3441 test questions. We also annotated ground truth logical forms for training and dev question texts and diagrams as described in the supervision subsections earlier. The train set logical forms were used for training various components of our system and the system hyperparameters were tuned on the dev set using grid search. We additionally evaluated our system on section 1 of three AP Physics C Mechanics tests[4]. Section 1 of the AP Physics C Mechanics practice test comprised of 10 questions and the official tests for 1998 and 2012 comprised of 75 and 35 questions respectively.

6.2 Results

We first evaluated the question parsing models individually. For diagram parsing, we computed the Jaccard similarity between the diagram elements detected by various versions of our diagram parser and compared them to gold elements. We considered *Edge Boxes* [67] – the best performing prior computer vision technique explored by us, and two variants of our diagram parser: diagram parser excluding text information and our full diagram parser. Table 2 reports the diagram parsing results on the dev set. Our diagram parser achieved a score of 69.8 which is much better than Edge Boxes. Prior computer vision techniques work on element and blob detection for natural images and do not port well to diagrams. However, our carefully engineered element detector works well. We observed that mapping elements to element labels by incorporating text information contributes to an improvement in the score.

We also studied the importance of each component of our diagram parser. We report the precision, recall, and F1 scores for identifying various diagram elements on the dev set in Table 3. We observed that our diagram parser achieves a very good performance on detecting the various diagram elements (except detecting plot elements). Parsing plots to a tabular representation is difficult and we wish to improve this in future work. In the final column of Table 3, we report precision, recall, and F1 scores for the final diagram parse generated by our diagram parser. Our parser achieved an impressive F1 score of 0.69 which is high given the complexity of the task.

Next, we evaluated our text parser. We compared the parses induced by our models with gold parses on the dev set. Table 4 reports Precision, Recall and F1 scores of the parses induced. For comparison purposes, we built a rule-based parser baseline. A similar baseline was proposed in [57] for their geometry solver. The baseline uses a set of manually designed high-precision rules. Each rule compares the dependency tree of each sentence to pre-defined templates, and if a template pattern is matched, the rule outputs the relation corresponding to that template. Our text-based parser achieved a F1 score of 0.66, a significant improvement over the rule-based parser (0.41).

We further break down this evaluation into the two components of text parsing: concept and relation identification. Table 5 shows the precision, Recall and F1 scores for concept and relation identification. Our parser achieved a high F1 score (0.91) for concept identification and a good F1 score (0.78) for relation identification.

[4]The other sections of the tests are subjective which we leave as future work. More details about the exam and the exam questions are available here: https://apstudent.collegeboard.org/apcourse/ap-physics-c-mechanics

	P	R	F1
Low level Elements	0.89	0.88	0.89
High Level Elements	0.86	0.84	0.85
Textual Elements	0.87	0.84	0.85
Label Associations	0.86	0.83	0.84
Object Detection	0.80	0.75	0.77
Mapping Objects to Object Labels	0.79	0.76	0.78
Plot Elements	0.37	0.23	0.28
Final diagram parse	0.74	0.65	**0.69**

Table 3: Precision, recall, and F1 scores for identifying various diagram elements and also associations between various diagram elements to labels in text. In the final column, we also report the precision, recall, and F1 scores for the final diagram parse generated by our diagram parser.

	P	R	F1
Rule-based	0.82	0.27	0.41
O.S.	0.62	0.72	**0.66**

Table 4: Precision, Recall and F1 scores of parses induced by our text parser compared to a rule based parser.

	P	R	F1
Concept	0.95	0.87	0.91
Relation	0.83	0.74	0.78

Table 5: Precision, Recall and F1 scores of the concept and relation identification components of our text parser.

	T	P	'98	'12
Humans	63	52	44	48
O.S.	**68**	50	42	**54**

Table 6: Score obtained by our system compared to average score obtained by 10 students on our dataset from physics textbooks (T), AP Physics C Mechanics - Section 1 practice test (P) and official tests for two years: 1998 and 2012. The improvement on the textbook dataset is statistically significant (p<0.05 using the two-tailed paired t-test).

	T	P	'98	'12
Humans-w/ diag	61	51	46	50
Humans-w/o diag	64	52	43	47
O.S.-w/ diag	60	42	38	44
O.S.-w/o diag	70	54	45	59

Table 7: Score obtained by our system and the average score obtained by students on questions with and without diagrams.

	T	P	1998	2012
A.M.	54	35	31	36
H.M.	59	39	33	42
Max	63	47	41	50

Table 8: Performance of our system when we use various baselines for reconciling text and diagram parses.

Finally, we report the performance of our overall system on the task of solving Newtonian physics problems. We performed a user study with 10 students[5] who were each asked to solve questions from various datasets. For the AP Physics C exams, each student took the entire test. Whereas, for the textbook dataset, each student was asked to answer a random selection of 100 questions in the test split. We scored the students as well as our model as the percentage of correctly answered questions. We compared the results of our system with the average score achieved by the students on the various datasets. On the textbook questions dataset, our system achieved a score of 68% which is better than the average student score of 63%. On all the out-of-domain three AP Physics exams, our system achieves close to the average student score, superseding it in the 2012 exam.

A key challenge for us was handling diagrams in unison with text. Thus, we also report the performance of our system and humans on questions which have (or do not have) associated diagrams in Table 7. We observed that while the student performance did not change significantly on questions which had diagrams, our system had a drop in performance on questions with diagrams. Thus, diagram parsing remains a key challenge for us and we wish to improve it in future.

We also analyzed the performance of our sub-modular optimizer for reconciling text and diagram parses. For this, we implemented some simple baselines for reconciling text and diagram parses. The first baseline assigns the score of each literal in the reconciled parse as the *mean* of its text parser score (confidence) and its diagram parser score (confidence). We have two variants of this baseline which use the arithmetic mean (A.M.) and the harmonic mean (H.M.), respectively. The second baseline assigns the score of a literal in the reconciled parse as the *maximum* of its text parser score and diagram parser score. Table 8 reports the performance of variants of our system where we use the aforementioned baselines for reconciling text and diagram parses instead of our sub-modular optimization approach. We can observe that the baselines incur a loss in performance over our sub-modular optimization approach. This validates the necessity of our approach.

[5]All the students selected for the user study scored atleast 4 ("well qualified to receive college credit") or 5 ("extremely well qualified to earn college credit") on the 2016 AP Physics C exam.

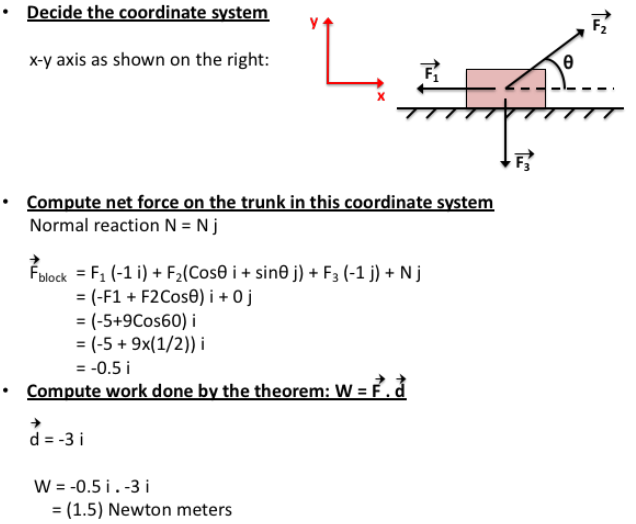

Figure 6: An example explaination for solving our running example in Figure 2. The problem is solved in three steps: first deciding the coordinate system, then computing the net force on the trunk in this coordinate system and finally computing the work done using a Physics theorem.

6.3 Error Analysis

We randomly sampled 100 incorrectly answered dev set questions and manually identified the sources of error for them. Out of the incorrectly answered 31 questions, we found that 20 of the errors are due to failures in text parsing, 16 of the errors are due to failures in diagram parsing. Among them, about 12 of the errors were due to failures in both diagram and text parsing. The rest of errors were due to problems that require more complex reasoning not handled in our system. We provide some additional analysis of our results with ten example correctly answered and ten example incorrectly answered questions in the supplementary[6].

7 EXPLANABILITY

One of the key benefits of our approach is that it provides an easy-to-understand student-friendly deductive solution to Newtonian physics problems. Thus, we claim that it can be useful as an assistive tool for students learning Newtonian physics. To test this hypothesis, we asked the 10 students (same set of students selected for the previous user study) to use our system as a web-based assistive tool. They were each asked to rate if they understood the various steps of the deduction proof or not for a sample of 100 dev questions. A sample explanation for our running example is shown in Figure 6. We found that on an average the participants claimed to 'understand' 86% of the steps of the deduction proof produced by our system. There was a unanimous agreement on 71% of the steps and a high Fleiss's κ value of 0.80. In addition, the

[6]The supplementary is available at http://www.cs.cmu.edu/~mrinmays/kdd_supplementary.pdf

participants claimed to 'understand' the entire proofs 67% of the time. This study lends support to our claims about the potential use of our solver as an assistive tool in education.

8 RELATED WORK

Situated question answering is the problem of answering questions about a constrained environment. There has been renewed interest in situated question answering [17, 34, 51, 57] in the recent years. Situated question answering poses two key challenges: (a) how to interpret the question and (b) how to use the background knowledge about the environment to determine the answer. Situated question answering is part of a larger NLP and Computer Vision effort called Machine comprehension which tests an AI system's ability to understand text (or images) through a series of question-answering tasks. Example machine comprehension tasks include reading comprehensions [11, 44, 46, 50, 52–54, inter alia], science question answering [12, 49, 55, inter alia], algebra word problems [36, 48, inter alia], geometry problems [51, 57] and pre-university entrance exams [5, 23].

Parsing to Programs maps the question text and diagram to a logical interpretation of the question and then uses explicit programs which encode background knowledge to determine the answer. Parsing to Programs can be seen as a natural language interface to expert systems. Expert systems [29] typically reason about knowledge using conventional procedural code. While expert systems have been developed for a large number of applications such as education [40], medicine [27, 63], law [37], sports [14] and forensics [15], they are known to be brittle. Our Parsing to Programs technique addresses this challenge by incorporating a parsing interface on top of a programmatic solving model.

Generating a formal interpretation of questions is a vital ingredient in our system. We achieved this by building a question text parser and a diagram parser and then reconciling the two interpretations. The question text parsing approach builds on three important areas of NLP: semantic parsing [64, 65, inter alia], semantic role labeling [24, inter alia] and relation extraction [6, inter alia]. However, a key challenge in our work was that the overall number of available Newtonian physics questions is quite small compared to typical NLP corpora. Hence, off-the-shelf semantic parsers, semantic role labelers or relation extractors do not work in this domain. We mitigated this problem by careful representation of our logical language and our parsing formalism to this language. The detection and recognition of diagrams is also more challenging than natural images. Hence, to tackle diagrams, we proposed various blob and object detectors along with agglomerative diagram element detection and also incorporated text information. Similar parsing approaches were pursued in [30] for food-web diagrams and in [51, 56] for geometry diagrams question answering. Our approach is influenced by these two pieces of work.

Finally, an important benefit of our approach over any other plausible approaches (e.g. deep learning) is that it provides an explainable solution to the problems in terms of known laws and axioms of physics which can be easily explained to students. This relates to a large body of work on intelligent tutoring systems. A full discussion on intelligent tutoring systems is beyond the scope of this paper but we point the interested reader to [3] for a better perspective.

9 CONCLUSION AND FUTURE WORK

We proposed a formalism called *Parsing to Programs*, combining ideas from parsing of natural language and diagrams with probabilistic programming for situated question answering. We used it to develop a system that can solve pre-university level Newtonian physics problems. Our system achieved close to human performance on a large dataset of physics textbook questions and on section 1 of the AP Physics C exams. Looking towards the future, our system can be a used to assist students learn Newtonian physics on platforms such as MOOCs. Hence, we plan to release it as a desktop and mobile application for public consumption by students in the future. While this paper focused on solving Newtonian physics problems, the approach can be extended to many other problems in fields such as science, engineering and finance.

ACKNOWLEDGMENT

We thank the anonymous reviewers for their comments. We acknowledge the CMLH fellowship to MS and ONR grant N000141712463 for funding our research.

REFERENCES

[1] 2016. AI beats top shogi player in first match of tournament final. (2016). http://www.asahi.com/ajw/articles/AJ201604110029.html
[2] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. 2012. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, 11 (2012), 2189–2202.
[3] John R Anderson, Albert T Corbett, Kenneth R Koedinger, and Ray Pelletier. 1995. Cognitive tutors: Lessons learned. *The journal of the learning sciences* 4, 2 (1995), 167–207.
[4] Noriko H Arai. 2015. The impact of AIâĂŤcan a robot get into the University of Tokyo? *National Science Review* (2015), 135–136.
[5] Noriko H Arai and Takuya Matsuzaki. 2014. The impact of AI on education–Can a robot get into The University of Tokyo?. In *Proc. ICCE*. 1034–1042.
[6] Nguyen Bach and Sameer Badaskar. 2007. *A review of relation extraction*. Technical Report. Language Technologies Institute, CMU.
[7] Samuel Bayer, Laurie Damianos, Christine Doran, Lisa Ferro, Randall Fish, Lynette Hirschman, Inderjeet Mani, Laurel Riek, and Beatrice Oshika. 2005. *Selected Grand Challenges in Cognitive Science*. Technical Report. MITRE.
[8] Jordan Boyd-Graber, Mohit Iyyer, He He, and Hal Daumé III. 2015. Interactive Incremental Question Answering. In *NIPS*.
[9] Murray Campbell, A Joseph Hoane, and Feng-hsiung Hsu. 2002. Deep blue. *Artificial intelligence* 134, 1 (2002), 57–83.
[10] João Carreira, Fuxin Li, and Cristian Sminchisescu. 2012. Object recognition by sequential figure-ground ranking. *International journal of computer vision* 98, 3 (2012), 243–262.
[11] Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858* (2016).
[12] Peter Clark. 2015. Elementary School Science and Math Tests as a Driver for AI:Take the Aristo Challenge!. In *Proceedings of IAAI*.
[13] Peter Clark and Oren Etzioni. 2016. My Computer is an Honor Student - but how Intelligent is it? Standardized Tests as a Measure of AI. In *Proceedings of AI Magazine*.
[14] Anthony C Constantinou, Norman E Fenton, and Martin Neil. 2012. pi-football: A Bayesian network model for forecasting Association Football match outcomes. *Knowledge-Based Systems* 36 (2012), 322–339.
[15] Anthony Costa Constantinou, Barbaros Yet, Norman Fenton, Martin Neil, and William Marsh. 2016. Value of information analysis for interventional and counterfactual Bayesian networks in forensic medical sciences. *Artificial intelligence in medicine* 66 (2016), 41–52.
[16] T. E. de Campos, B. R. Babu, and Manik Varma. 2009. Character recognition in natural images. In *Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal*.

[17] Dina Demner-Fushman and Jimmy Lin. 2006. Situated question answering in the clinical domain: selecting the best drug treatment for diseases. In *Proceedings of the Workshop on Task-Focused Summarization and Question Answering*. Association for Computational Linguistics, 24–31.

[18] Richard O Duda and Peter E Hart. 1972. Use of the Hough transformation to detect lines and curves in pictures. *Commun. ACM* 15, 1 (1972), 11–15.

[19] Ahmed Elgammal, David Harwood, and Larry Davis. 2000. Non-parametric model for background subtraction. In *European conference on computer vision*. Springer, 751–767.

[20] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. 2010. Object detection with discriminatively trained part-based models. *IEEE TPAMI* 32, 9 (2010), 1627–1645.

[21] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. 2010. Building Watson: An overview of the DeepQA project. *AI magazine* 31, 3 (2010), 59–79.

[22] Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A Discriminative Graph-Based Parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 1426–1436.

[23] Akira Fujita, Akihiro Kameda, Ai Kawazoe, and Yusuke Miyao. 2014. Overview of Todai Robot Project and Evaluation Framework of its NLP-based Problem Solving. *World History* 36 (2014), 36.

[24] Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics* 28, 3 (2002), 245–288.

[25] Chunhui Gu, Joseph J Lim, Pablo Arbeláez, and Jitendra Malik. 2009. Recognition using regions. In *Computer Vision and Pattern Recognition (CVPR), 2009*. IEEE, 1030–1037.

[26] C. Harris and M. Stephens. 1988. A Combined Corner and Edge Detection. In *Proceedings of The Fourth Alvey Vision Conference*. 147–151.

[27] Alan M Hofmeister, R Brad Althouse, Marilyn Likins, Daniel P Morgan, et al. 1994. SMH. PAL: An expert system for identifying treatment procedures for students with sever disabilities. *Exceptional children* 61, 2 (1994), 174.

[28] S Impedovo, L Ottaviano, and S Occhinegro. 1991. Optical character recognition – a survey. *International Journal of Pattern Recognition and Artificial Intelligence* 5, 01n02 (1991), 1–24.

[29] Peter Jackson. 1986. *Introduction to expert systems*. Addison-Wesley Pub. Co., Reading, MA.

[30] Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Min Joon Seo, Hannaneh Hajishirzi, and Ali Farhadi. 2016. A Diagram is Worth a Dozen Images. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*. 235–251.

[31] Dan Klein and Christopher D Manning. 2004. Parsing and hypergraphs. In *New developments in parsing technology*. Springer, 351–372.

[32] Iasonas Kokkinos. 2010. Highly accurate boundary detection and grouping. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2520–2527.

[33] Andreas Krause and Daniel Golovin. 2012. Submodular Function Maximization. (2012).

[34] Jayant Krishnamurthy, Oyvind Tafjord, and Aniruddha Kembhavi. 2016. Semantic Parsing to Probabilistic Programs for Situated Question Answering. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, Jian Su, Xavier Carreras, and Kevin Duh (Eds.). The Association for Computational Linguistics, 160–170.

[35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[36] Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

[37] Philip Leith. 2010. The rise and fall of the legal expert system. *European Journal of Law and Texhnology* 1 (march 2010).

[38] Tony Lindeberg. 1993. Detecting salient blob-like image structures and their scales with a scale-space primal sketch: a method for focus-of-attention. *International Journal of Computer Vision* 11, 3 (1993), 283–318.

[39] André FT Martins, Noah A Smith, and Eric P Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*. Association for Computational Linguistics, 342–350.

[40] N Nwigbo Stella and Agbo Okechuku Chuks. 2011. Expert system: a catalyst in educational development in Nigeria. (2011).

[41] Nobuyuki Otsu. 1975. A threshold selection method from gray-level histograms. *Automatica* 11, 285-296 (1975), 23–27.

[42] Emanuel Parzen. 1962. On estimation of a probability density function and mode. *The annals of mathematical statistics* 33, 3 (1962), 1065–1076.

[43] J Pont-Tuset, P Arbelaez, J Barron, F Marques, and J Malik. 2016. Multiscale Combinatorial Grouping for Image Segmentation and Object Proposal Generation. *IEEE TPAMI* (2016).

[44] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* (2016).

[45] Pekka Rantalankila, Juho Kannala, and Esa Rahtu. 2014. Generating object segmentation proposals using global and local search. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2417–2424.

[46] Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text.. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*.

[47] Murray Rosenblatt et al. 1956. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics* 27, 3 (1956), 832–837.

[48] Subhro Roy and Dan Roth. 2015. Solving General Arithmetic Word Problems. In *Proceedings of EMNLP*.

[49] Mrinmaya Sachan, Avinava Dubey, and Eric P. Xing. 2016. Science Question Answering using Instructional Materials. *CoRR* abs/1602.04375 (2016). http://arxiv.org/abs/1602.04375

[50] Mrinmaya Sachan, Avinava Dubey, Eric P Xing, and Matthew Richardson. 2015. Learning Answer-Entailing Structures for Machine Comprehension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

[51] Mrinmaya Sachan, Kumar Dubey, and Eric Xing. 2017. From Textbooks to Knowledge: A Case Study in Harvesting Axiomatic Knowledge from Textbooks to Solve Geometry Problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 773–784.

[52] Mrinmaya Sachan and Eric Xing. 2016. Easy questions first? a case study on curriculum learning for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 453–463.

[53] Mrinmaya Sachan and Eric Xing. 2016. Machine comprehension using rich semantic representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Vol. 2. 486–492.

[54] Mrinmaya Sachan and Eric P. Xing. 2018. Self-Training for Jointly Learning to Ask and Answer Questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*. 629–640.

[55] Carissa Schoenick, Peter Clark, Oyvind Tafjord, Peter D. Turney, and Oren Etzioni. 2016. Moving Beyond the Turing Test with the Allen AI Science Challenge. *CoRR* abs/1604.04315 (2016). http://arxiv.org/abs/1604.04315

[56] Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, and Oren Etzioni. 2014. Diagram Understanding in Geometry Questions. In *Proceedings of AAAI*.

[57] Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving geometry problems: combining text and diagram interpretation. In *Proceedings of EMNLP*.

[58] Noah Siegel, Zachary Horvitz, Roie Levin, Santosh Kumar Divvala, and Ali Farhadi. 2016. FigureSeer: Parsing Result-Figures in Research Papers. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VII*. 664–680.

[59] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.

[60] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[61] Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. 2013. Selective search for object recognition. *International journal of computer vision* 104, 2 (2013), 154–171.

[62] Paul Viola and Michael J Jones. 2004. Robust real-time face detection. *International journal of computer vision* 57, 2 (2004), 137–154.

[63] Linda K Woolery and Jerzy Grzymala-Busse. 1994. Machine learning for an expert system to predict preterm birth risk. *Journal of the American Medical Informatics Association* 1, 6 (1994), 439–446.

[64] John M. Zelle and Raymond J. Mooney. 1993. Learning Semantic Grammars with Constructive Inductive Logic Programming. In *Proceedings of the 11th National Conference on Artificial Intelligence. Washington, DC, USA, July 11-15, 1993*. 817–822.

[65] John M Zelle and Raymond J Mooney. 1996. Learning to Parse Database Queries Using Inductive Logic Programming. In *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*.

[66] Ziming Zhang, Jonathan Warrell, and Philip HS Torr. 2011. Proposal generation for object detection using cascaded ranking svms. In *Computer Vision and Pattern Recognition (CVPR), 2011*. IEEE, 1497–1504.

[67] C Lawrence Zitnick and Piotr Dollár. 2014. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision*. Springer, 391–405.