A Meaning-based Statistical English Math Word Problem Solver

Chao-Chun Liang, Yu-Shiang Wong, Yi-Chung Lin and Keh-Yih Su
Institute of Information Science, Academia Sinica, Taiwan
{ccliang, yushiangwtw, lyc, kysu}@iis.sinica.edu.tw

Abstract

We introduce MeSys, a meaning-based approach to solving English math word problems (MWP) via understanding and reasoning in this paper. It first analyzes the text, transforms both body and question parts into their corresponding logic forms, and then performs inference on them. The associated context of each quantity is represented with proposed role-tags (e.g., nsubj, verb, etc.), which provides the flexibility for annotating an extracted math quantity with its associated context information (i.e., the physical meaning of this quantity). Statistical models are proposed to select the operator and operands. A noisy dataset is designed to assess if a solver solves MWP mainly via understanding or pattern matching. Experimental results show that our approach outperforms existing systems on both benchmark datasets and the noisy dataset, which demonstrates that the proposed approach more understands the meaning of each quantity in the text.

1 Introduction

The math word problem (MWP) (see Figure 2(a)) is frequently chosen to study natural language understanding and simulate human problem solving (Bakman, 2007; Hosseini et al., 2014; Liang et al., 2016) for the following reasons: (1) the answer to the MWP cannot be simply extracted by performing keyword/pattern matching. It thus shows the merit of understanding and inference. (2) An MWP usually possesses less complicated syntax and requires less amount of domain knowledge, so the researchers can focus on the task of understanding and reasoning. (3) The body part of MWP that provides the given information for solving the problem consists of only a few sentences. The understanding and reasoning procedures could be checked more efficiently. (4) The MWP solver has its own applications such as Computer Math Tutor.

According to the approaches used to identify entities, quantities, and to select operations and operands, previous MWP solvers can be classified into: (1) Rule-based approaches (Mukherjee and Garain, 2008; Hosseini et al., 2014), which make all related decisions based on a set of rules; (2) Purely statistics-based approaches (Kushman et al., 2014; Roy et al., 2015; Zhou et al., 2015), in which all related decisions are done via a statistical classifier; (3) DNN-based approaches (Ling et al, 2017; Wang et al, 2017), which map the given text into the corresponding math operation/equation via a DNN; and (4) Mixed approaches, which identify entities and quantities with rules, yet, decide operands and operations via statistical/DNN classifiers. This category can be further divided into two subtypes: (a) Without understanding (Roy and Roth, 2015; Koncel-Kedziorski et al., 2015; Huang et al, 2017; Shrivastava et al., 2017), which does not check the entity-attribute consistency between each quantity and the target of the given question; and (b) With understanding (Lin et al., 2015; Mitra and Baral, 2016), which also checks the entity-attribute consistency while solving the problem.

However, a widely covered rule-set is difficult to construct for the rule-based approach. Also, it is awkward in resolving ambiguity problem. In contrast, the performance of purely statistics-based approaches deteriorates significantly when the MWP includes either irrelevant statistics-based approaches and information gaps (Hosseini et al., 2014), as it is solved without first understanding the meaning.

For the category (3a), since the physical meaning is only implicitly utilized and the result is not generated via inference, it would be difficult to explain how the answer is obtained in a human

1 It is a survey paper which reviews most of the rule-based approaches before 2008.
comprehensible way. Therefore, the categories (2) and (3a) belong to the less favored direct translation approach\(^2\) (Pape, 2004).

In contrast, the approaches of (3b) can avoid the problems mentioned above. However, among them, Mitra and Baral (2016) merely handled Addition and Subtraction. Only the meaning-based framework proposed by Lin et al. (2015) can handle general MWPs via understanding and reasoning. Therefore, it is possible to explain how the answer is obtained in a human comprehensible way (Huang et al., 2015). However, although their design looks promising, only a few Chinese MWPs had been tested and performance was not evaluated. Accordingly, it is hard to make a fair comparison between their approach and other state-of-the-art methods. In addition, in their prototype system, the desired operands of arithmetic operations are identified with predefined lexical-syntactic patterns and ad-hoc rules. Reusing the patterns/rules designed for Chinese in another language is thus difficult even if it is possible.

In this paper, we adopt the framework proposed by Lin et al. (2015) to solve English MWPs (for its potential in solving difficult/complex MWPs and providing more human comprehensible explanations). Additionally, we make the following contributions: (1) A new statistical model is proposed to select operands for arithmetic operations, and its model parameters can be automatically learnt via weakly supervised learning (Artzi and Zettlemoyer, 2013). (2) A new informative and robust feature-set is proposed to select the desired arithmetic operation. (3) We show the proposed approach significantly outperforms other existing systems on the common benchmark datasets reported in the literature. (4) A noisy dataset with more irrelevant quantities in MWPs is created and released. It could be used to check if an approach really understands what a given MWP looks for. (5) An experiment is conducted to compare various approaches on this new dataset. The superior performance of our system demonstrates that the proposed meaning-based approach has good potential in handling difficult/complex MWPs.

2 System Description

The adopted meaning-based framework (Lin et al., 2015) is a pipeline with following four phases: (1) Language Analysis, (2) Solution Type Identification, (3) Logic Form Transformation and (4) Logic Inference. Our system adopts the Stanford CoreNLP suite (Manning et al., 2014) as the language analysis module. The other three modules are briefly described below. Last, we adopt the weakly supervised learning (Artzi and Zettlemoyer, 2013; Kushman et al., 2014) to automatically learn the model parameters without manually annotating each MWP with the adopted solution type and selected operands benchmark.

2.1 Solution Type Identification (STI)

After language analysis, each MWP is assigned with a specific solution type (such as Addition, Multiplication, etc.) which indicates the stereotype math operation pattern that should be adopted to solve this problem. We classify the English MWPs released by Hosseini et al. (2014) and Roy and Roth (2015) into 6 different types: Addition, Subtraction, Multiplication, Division, Sum and TVQ-F\(^3\). An SVM (Chang and Lin, 2011) is used to identify the solution type with 26 features. Most of them are derived from some important properties associated with each quantity.

In addition to the properties Entity\(^4\) and Verb (Hosseini et al., 2014) associated with the quantity, we also introduce a new property Time which encodes the tense and aspect of a verb into an integer to specify a point in the timeline. We assign 2, 4, and 6 to the tenses Past, Present and Future, respectively, and then adjust it with the aspect-values -1, 0 and 1 for Perfect, Simple, and Progressive, respectively.

Another property Anchor is associated with the unknown quantity asked in the question sentence. If the subject of the question sentence is a noun phrase (e.g., “how many apples does John have?”), Anchor is the subject (i.e., John). If the subject is an expletive nominal (e.g. “how many apples are there in the box?”), then Anchor is the associated nominal modifier nmod (i.e., “box”). Otherwise, Anchor is set to “Unknown”.

Inspired by (Hosseini et al., 2014), we transform Verb to Verb-Class (VC) which is positive, negative or static. A verb is positive/negative if it increases/decreases the associated quantity of the

\(^2\) According to (Pape, 2004), the meaning-based approach of solving MWPs achieves the best performance among various behaviors adopted by middle school children.

\(^3\) TVQ-F means to get the final state of a Time-Variant-Quantity that involves both Addition and Subtraction.

\(^4\) In our works, the term “Entity” also includes the unit of the quantity (e.g., “cup of coffee”).
subject. For example, in the sentence “Tom borrowed 3 dollars from Mike”, the verb is positive because the money of subject “Tom” increases.

However, a positive verb does not always imply the Addition operation. If the question is “How much money does Mike have now?”, the operation should be Subtraction. Two new properties Anchor-Role (AR) and Action (A) are thus proposed: AR indicates the role that Anchor associated with qi, and is set to nsubj/obj/nmod/φ. Ai is determined by following rules: (1) Ai=positive if (VCi, ARi) is either (positive, nsubj) or (negative, obj/nmod). (2) Ai=negative if (VCi, ARi) is either (negative, nsubj) or (positive, obj/nmod). (3) Otherwise, Ai=VCi.

To rule out the noisy quantities introduced by irrelevant information, we further associate each known quantity with the property Relevance (R) according to the unknown quantity asked in the question sentence. Let qi denote the i-th known quantity, Ei denote the entity of qi, Xi denote the property X of qi, qU denote the unknown quantity asked, and XU denote the property X of qU. Ri is specified with following rules: (1) Rj=2 (Directly-Related) if either {Anchor is Unknown & Ei entails EU} or {Anchor is not Unknown & ARi=φ & Ei entails Ei}. (2) Rj=1 (Indirectly-Related) if there is a qi which maps5 to qj and Ri=2 (i.e., qi is Directly-Related). (3) Rj=0 (Unrelated) otherwise.

The solution type is identified by an SVM based on 26 binary features. Let the symbols p, n, s, A, E, R, T, V, Sb, So and wQ stand for positive, negative, stative, Action, Entity, Relevance, Time, Verb, “a body sentence”, “the question sentence” and “a word in question sentence” respectively. Also, let R(x) be the indicator function to check if x is true. The 26 features are briefly described as follows:

5 That is, qj is linked to a directly-related quantity qi under an expression such as “2 pencils weigh 30 grams”.

Figure 1: An example of logic form transformation

(a) Pack 100 candies into 5 boxes.
(b) (c) gmap(n1,q1,q2)&verb(n1,pack)
(d) quan(q1,f,candy)=100&verb(q1,pack)&nmod(q1,n2)
quan(q2,box)=5&verb(q2,pack)&nsubj(q2,n1)

(1) VCi=p; (2) ∃Rj=2 s.t. Aij=p; (3) ∃Rj=2 s.t. Aij=n;
(4) ∃Rj=2 s.t. Aij=s; (5) ∑q(Rj=2)>2;
(6) ∑q(Rj=2 & Ai∈{p,n})=2;
(7) ∃Rj=2 s.t. Aip & Ti<Tj;
(8) ∃Rj=2 s.t. Ais & Ti<Tj;
(9) ∃Rj=2 s.t. Ai=s & Ti=max Ti;
(10) ∃Rj=2 s.t. Ai=s & Ti<Tj;
(11) Ti≥max Ti; (12) Ti=min Ti;
(13) ∀Rj=2, Vj are the same; (14) ∀Rj=2 s.t. Ti=Tj;
(15) ∀Rj=2, Tj are the same;
(16) ∃Rj=2, ∃Rj=1 s.t. qi maps to qj & qi>qj;
(17) ∃Rj=2, ∃Rj=1 s.t. qi maps to qj & qi is associated with a word “each/every/per/an”;
(18) ∃Rj=2, ∃Rj=1 s.t. qi maps to qj & qi is associated with a word “each/every/per/an”;
(19) ∃qj, qi s.t. Rj=Ri=2 & Vi=Vj = Vi;
(20) wQ ∈ {total, in, altogether, sum};
(21) wQ ∈ {more, than} or wQ s.t. wQ-POS=RBR;
(22) wQ = "left"; (23) qj appears in S0;
(24) “the rest ∨ Ei,” appears in S0 (V for any verb);
(25) “each NN” appears in S0 (NN for any noun);
(26) Anchori is Unknown/nmod & VCi = s.

2.2 Logic Form Transformation (LFT)

The results of language analysis are transformed into a logic form, which is expressed with the first-order logic (FOL) formalism (Russell and Norvig, 2009). Figure 1 shows how to transform the sentence (a) “Pack 100 candies into 5 boxes,” into the corresponding logic form (d). First, the dependency tree (b) is transformed into the semantic representation tree (c) adopted by Lin et al., (2015). Afterwards, according to the procedure proposed in Lin et al., (2015), the domain-dependent logic expressions are generated in (d).

The domain-dependent logic expressions are related to crucial generic math facts, such as quantities and relations between quantities. The FOL function quan(quand, unit6, entity)=number is for
(a) A sandwich is priced at $0.75. A pudding is priced at $0.25. Tim bought 2 sandwiches and 4 puddings. Mary bought 2 puddings. How much should Tim pay?

(b) ...price(sandwich,0.75)&price(pudding,0.25)... quan(q1,#,sandwich)=2&verb(q1,buy)&nsubj(q1,Tim)... quan(q2,#,pudding)=4&verb(q2,buy)&nsubj(q2,Tim)... quan(q3,#,pudding)=2&verb(q3,buy)&nsubj(q3,Mary)... ASK Sum(?q,?dollar,#,verb(?q,pay)&nsubj(?q,Tim))

(c) quan(?q,?u,?o) & verb(?q,pay)&nsubj(?q,Tim) → quan($q,dollar,#)=quan(?q,?u,?o)×?p & verb($q,pay) & nsubj($q,Tim)

(d) quan(q4,dollar,#)=1.5&verb(q4,pay)&nsubj(q4,Tim)... quan(q5,dollar,#)=1&verb(q5,pay)&nsubj(q5,Tim)... quan(q6,dollar,#)=0.5&verb(q6,pay)&nsubj(q6,Mary)

Figure 2: A logic inference example

2.3 Logic Inference

The logic inference module adopts the inference engine from Lin et al., (2015). Figure 2 shows how it uses inference rules to derive new facts from the initial facts directly provided from the description. The MWP (a) provides some facts (b) generated from the LFT module. An inference rule (c), which implements the common sense description. The MWP (a) provides some facts (b) from the initial facts directly provided from the engine from Lin et al., (2015). Figure 2 shows how it uses inference rules to derive new facts (d). The facts associated with q6 can be interpreted as “Mary paid 0.5 dollar for two puddings”.

The inference engine (IE) also provides 5 utility functions, including Addition, Subtraction, Multiplication and Division, and Sum. The first four utilities all return a value by performing the named math operation on its two input arguments. On the other hand, Sum(function,condition) returns the sum of the values of FOL function instances which can be unified with the first argument (i.e., function) and satisfy the second argument (i.e., condition). For example, according to the last line in Figure 2(b), three newly derived quantity facts q4, q5 and q6 (in 2(d)) can be unified with the first argument quan(?q,dollar,#) in 2(c), but only q4 and q5 satisfy the second argument verb(?q,pay)&nsubj(?q,Tim). As a result, the answer 2.5 is returned by taking sum on the values of the quantity facts quan(q4,dollar,#) and quan(q5,dollar,#).

2.4 Probabilistic Operand Selection

The most error-prone part in the LFT module is instantiating the utility function of math operation especially if many irrelevant quantity facts appear in the given MWP. Figure 3 shows the LFT module needs to select two quantity facts (among 4) for Addition. Please note that the question quantity Q_0, transformed from “how many flowers”, is not a candidate for operand selection.

Lin et al., (2015) used predefined lexicosyntactic patterns and ad-hoc rules to instantiate utility functions. However, their rule-based approach fails when the MWP involves more quantifiers.

(a) Tim bought 2 roses and 3 lilies. Mary bought 4 roses and 5 lilies. How many flowers did Tim buy?

(b) quan(q1,#,rose)=2&verb(q1,buy)&nsubj(q1,Tim)... quan(q2,#,lily)=3&verb(q2,buy)&nsubj(q2,Tim)... quan(q3,#,rose)=4&verb(q3,buy)&nsubj(q3,Mary)... quan(q4,#,lily)=5&verb(q4,buy)&nsubj(q4,Mary)... quan(q5,#,flower)&verb(q5,buy)&nsubj(q5,Tim)...
ties. Therefore, we propose a statistical model to select operands for the utility functions Addition, Subtraction, Multiplication and Division. The operand selection procedure can be regarded as finding the most likely configuration \((q^n, r)\), where \(q^n = q_1, \ldots, q_n\) is a sequence of random indicators which denote if the corresponding quantity will be selected as an operand, and \(r\) is a tri-state variable to represent the relation between the values of two operands (i.e., \(r = -1, 0\) or \(1\); which denote that the first operand is less than, equal to, or greater than the second operand, respectively). Given a solution type \(s\), the MWP logic expressions \(L\) and the \(n\) quantities \(q^n\) = \(q_1, \ldots, q_n\) in \(L\). The procedure is formulated as:

\[
P(r, q^n|L, s) = P(r|s) \times P(q^n|L, \Phi(q_i, L, s)),
\]

(1)

The last factor in Eq (1) is further derived as:

\[
P(q^n|L, s) \approx \prod_{i=1}^{n} P(q_i|L, s) \approx \prod_{i=1}^{n} P(o_i|\Phi(q_i, L, s)),
\]

(2)

where \(\Phi()\) is a feature extraction function to map \(q_i\) and its context into a feature vector. Here, the probabilistic factor \(P(q_i|\Phi(q_i, L, s))\) is obtained via an SVM classifier (Chang and Lin, 2011).

\(\Phi()\) extracts 25 features (24 of them are binary) for \(q_i\). The following 11 of them are independent on the question in the MWP:

1. Four features to indicate if \(s\) is Addition, Subtraction, Multiplication or Division.
2. A feature to indicate if \(q_i\) is within a qmap(…).
3. A feature to indicate if \(q_i = 1\).
4. Five features to indicate if \(n < 2, n = 2, n = 3, n = 4\) or \(n > 4\); where \(n\) is the number of quantities in Eq (1).

\(\Phi()\) also extracts features by matching the logic expressions of \(q_0\) to check the role-tag consistencies between \(q_i\) and \(q_0\). Another fourteen features are extracted with three indicator functions \(I_m(), I_e(), I_3()\) and one tri-state function \(T_m()\) as follows:

\[
\begin{align*}
I_m(q_i, q_0, nsubj) & \quad I_e(q_i, q_0, entity), \\
I_m(q_i, q_0, verb) & \quad I_e(q_i, q_0, verb), \\
I_3(q_0, nsubj) & \quad T_m(q_i, q_0, nsubj), \\
I_3(q_0, modifier) & \quad I_m(q_i, q_0, modifier), \\
I_3(q_0, place) & \quad I_m(q_i, q_0, place), \\
I_3(q_0, temporal) & \quad I_m(q_i, q_0, temporal), \\
I_3(q_0, xcomp) & \quad I_m(q_i, q_0, xcomp)
\end{align*}
\]

where the indicator functions \(I_m(x, y, z)\) checks if the \(z\) of \(x\) matches the \(z\) of \(y\), \(I_e(x, y, z)\) checks if the \(z\) of \(x\) entails the \(z\) of \(y\) and \(I_3(y, z)\) checks if the \(z\) of \(y\) exists. \(T_m(q_i, q_0, nsubj)\) returns “exact-match” (if \(nsubj\) of \(q_i\) matches \(nsubj\) of \(q_0\)), “quasi-match” (if \(nsubj\) of \(q_0\) does not exist or is a plural pronoun), and “unmatch”.

\(I_e()\) uses WordNet to check the entailment relation. The entity, verb and \(nsubj\) of a quantity are determined according to the logic expressions. The modifier, place, temporal and \(xcomp\) of a quantity are extracted from the dependency tree with some lexicosyntactic patterns. For example, the \(modifier\) and \(place\) of the quantity in the sentence “There are 30 red flowers in the garden.” are “red” and “garden” respectively. The temporal and \(xcomp\) of a quantity are extracted according to the dependency relations “tmod” (i.e., temporal \(modifier\)) and “\(xcomp\)” (i.e., open clausal complement), respectively.

3 Datasets for Performance Evaluation

The AI2 dataset provided by Hosseini et al. (2014) and the IL dataset released by Roy and Roth (2015) are adopted to compare our approach with other state-of-the-art methods. The AI2 dataset has 395 MWPs on addition and subtraction, with 121 MWPs containing irrelevant information (Hosseini et al., 2014). It is the most popular one for comparing different approaches. On the other hand, the IL dataset consists of 562 elementary MWPs which can be solved by one of the four arithmetic operations (i.e., +, −, ×, and ÷) without any irrelevant quantity. It is the first publicly available dataset for comparing performances that covers all four arithmetic operations.

However, the difficulty of solving an MWP depends not only on the number of arithmetic operations required, but also on how many irrelevant quantities inside, and even on how the quantities are described. One way to test if a proposed approach solves the MWPs with understanding is to check whether it is robust to those irrelevant quantities. Therefore, it is desirable to have a big enough dataset that contains irrelevant quantities which are created under different situations (e.g., confusing with an irrelevant agent, entity, or modifier, etc.) and allow us to probe the system weakness from different angles. We thus create a new dataset with more irrelevant quantities. But before we do that, we need to know how difficult the task of solving the given MWPs is. Therefore, we first
propose a way to measure how easy that a system solves the problem by simply guessing.

3.1 Perplexity-flavor Measure

We propose to adopt the Perplexity to measure the task difficulty, which evaluates how likely a solver will get the correct answer by guessing. Every MWP in the datasets can be associated with a solution expression template, such as “□ + □” or “□ − □”, where the symbol □ represents a slot to hold a quantity. The solution can be obtained by placing correct quantities at appropriate slots. A random baseline is to solve an MWP by guessing. It first selects a solution expression template according to the prior distribution of the templates and then places quantities into the selected template according to the uniform distribution.

The expected accuracy of the random baseline on solving an MWP is a trivial combination and permutation exercise. For example, the expected accuracy of solving an MWP associated with “□ + □” template is \( p_{□+□} \times \binom{n}{2}^{-1} \), where the factor \( p_{□+□} \) denotes the prior probability of the template “□ + □” and \( n \) is the total number of quantities (including irrelevant ones) in the MWP. On the other hand, expected accuracy of solving an MWP associated with “□ − □” template is \( p_{□-□} \times \binom{n}{2}^{-1} \). Let \( A_i \) denote the expected accuracy of solving the \( i \)-th MWP in a dataset. The accuracy of the random baseline on the dataset of size \( N \) is then computed as

\[
A = \frac{1}{N} \sum_{i=1}^{N} A_i
\]

The word “Accuracy” comprises the opposite sense of the word “Perplexity” (i.e., in the sense of how hard a prediction problem is). The lower the Accuracy is, the higher the Perplexity is. Therefore, we transform the Accuracy measure into a Perplexity-Flavor measure (PP) via the formula:

\[
PP = 2^{-\log_2 A}
\]

For instance, the Perplexity-Flavor measures of AI2 and IL datasets are 4.46 and 8.32 respectively.

3.2 Noisy Dataset

Human Math/Science tests have been considered more suitable for judging AI progress than Turing test (Clark and Etzioni, 2016). In our task, solving MWPs is mainly regarded as a test for intelligence (not just for creating a Math Solver package). A noisy dataset is thus created to assess if a solver solves the MWPs mainly via understanding or via mechanical/statistical pattern matching. If a system solves an MWP mainly via pattern matching, it would have difficulty in solving a similar MWP augmented from the original one with some irrelevant quantities. Therefore, we first create a noisy dataset by selecting some MWPs that can be correctly solved, and then augmenting each of them with an additional noisy sentence which involves an irrelevant quantity. This dataset is created to examine if the solver knows that this newly added quantity is irrelevant.

Figure 4 shows how we add noise into an MWP template. (a.1) is created by associating an irrelevant quantity to a new subject (i.e., Mary). Here the ellipse symbol “…” denotes unchanged text. (a.2) is obtained by associating an irrelevant quantity to a new entity (i.e., books). In addition, we also change modifiers (such as yellow, red, …) to add new noisy sentence (not shown here). Since the noisy dataset is not designed to assess the lexicon coverage rate of a solver, we reuse the words in the original dataset as much as possible while adding new subjects, entities and modifiers.

136 MWPs that both Illinois Math Solver (Roy and Roth, 2016) and our system can correctly solve are selected from the AI2 and IL datasets. This subset is denoted as OSS (Original Sub-Set). Afterwards, based on the 136 MWPs of OSS, we create a noisy dataset of 396 MWPs by adding irrelevant quantities. This noisy dataset is named as NDS. Table 1 lists the size of MWPs, Perplexity-flavor Measure.

<table>
<thead>
<tr>
<th># MWPs</th>
<th>OSS</th>
<th>NDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perplexity (PP)</td>
<td>7.42</td>
<td>18.83</td>
</tr>
<tr>
<td># Quantities</td>
<td>2.64</td>
<td>3.62</td>
</tr>
</tbody>
</table>

Table 1: Perplexity measures of OSS and NDS test (Clark and Etzioni, 2016).
ties (PP), and the average numbers of quantities in each MWP of these two datasets.

4 Experimental Results and Discussion

We compare our approach with (Roy and Roth, 2015) because they achieved the state-of-the-art performance on the IL dataset, which consists of MWPs on all four arithmetic operations. In their approach, each quantity in the MWP was associated with a quantity schema whose attributes are extracted from the context of the quantity. Based on the attributes, several statistical classifiers were used to select operands and determine the operator. They also reported the performances on the AI2 dataset to compare their approach with those of others (e.g., Kushman et al., 2014), which is a purely statistical approach that aligns the text with various pre-extracted equation templates.

To compare the performance of the statistical method with the DNN approach, we only implement a Bi-directional RNN-based Solution Type Identifier (as our original statistical Operand Selector is relatively much better). It consists of a word embedding layer (both body and question parts), and a bidirectional GRU layer as an encoder. We apply the attention mechanism to scan all hidden state sequence of body by the last hidden state of question to pay more attention to those more important (i.e., more similar between the body and the question) words. Lastly, it outputs the solution type by a softmax function. We train it for 100 epochs, with mini-batch-size = 128 and learning-rate = 0.001; the number of nodes in the hidden layer is 200, and the drop-out rate is 0.713.

We follow the same n-fold cross-validation evaluation setting adopted in (Roy and Roth, 2015) exactly. Therefore, various performances could be directly compared. Table 2 lists the accuracies of different systems in solving the MWPs of various datasets. The last two rows are extracted from (Roy and Roth, 2015). The results show that our performances of the statistical approach significantly outperform that of our DNN approach and other two systems on every dataset.

The performances of STI and LFT modules are listed in Table 3. As described in section 2, the benchmark for both solution type and the operand selection benchmark are automatically determined

13 Since the data-set is not large enough for splitting a development set, we choose those hyper parameters based on the test set in coarse grain. Therefore, the DNN performance we show here might be a bit optimistic.

14 In evaluating the performances on OSS and NDS datasets, our system is trained on the folds 2-5 of the IL dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>STI (Statistical)</th>
<th>STI (DNN)</th>
<th>LFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI2</td>
<td>81.5</td>
<td>74.5</td>
<td>92.1</td>
</tr>
<tr>
<td>IL</td>
<td>81.0</td>
<td>68.8</td>
<td>94.8</td>
</tr>
</tbody>
</table>

Table 2: Performances of various approaches

<table>
<thead>
<tr>
<th>Dataset</th>
<th>STI (DNN)</th>
<th>LFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI2</td>
<td>74.5</td>
<td>92.1</td>
</tr>
<tr>
<td>IL</td>
<td>68.8</td>
<td>94.8</td>
</tr>
</tbody>
</table>

Table 3: Performances of different STIs and LFT

<table>
<thead>
<tr>
<th>Dataset</th>
<th>STI (DNN)</th>
<th>LFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSS</td>
<td>81.4</td>
<td>92</td>
</tr>
<tr>
<td>OSS'</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>NDS</td>
<td>73.7</td>
<td>74.5</td>
</tr>
<tr>
<td>NDS'</td>
<td>81.4</td>
<td>75.4</td>
</tr>
</tbody>
</table>

Table 4: Performances on the OSS and NDS by weakly supervised learning. The first and second rows of Table 3 show the solution type accuracies of our statistical and DNN approaches. The third row shows the operand selection accuracy obtained by giving the solution type benchmark. Basically, LFT accuracies are from 92% to 95%, and the system accuracies are dominated by STI.

The left-hand side of Table 4 shows the performances on the OSS and NDS datasets. Recall that OSS is created by selecting some MWPs which both Illinois Math Solver (Roy and Roth, 2016) and our system14 can correctly solve. Therefore, both systems have 100% accuracy in solving the OSS dataset. However, these two systems behave very differently while solving the noisy dataset. The much higher accuracy of our system on the noisy dataset shows that our meaning-based approach understands the meaning of each quantity more. As a result, it is less confused with the irrelevant quantities.

One MWP in the noisy dataset that confuses Illinois Math Solver (IMS) is “Tom has 9 yellow balloons. Sara has 8 yellow balloons. Bob has 5 yellow flowers. How many yellow balloons do they have in total?”, where the underlined sentence is the added noisy sentence. The solver sums all quantities and gives the wrong answer 22, which reveals that IMS cannot understand that the quantity “5 yellow flowers” is irrelevant to the question “How many yellow balloons?”. On the contrary, our system avoids this mistake.

Although the meaning of each quantity is explicitly checked in our LFT module, our system
still cannot correctly solve all MWPs in NDS. The error analysis reveals that the top-4 error sources are STI, LFT, CoreNLP and incorrect problem construction (for 27%, 27%, 18%, 18%), which indicates that our STI and LFT still cannot completely prevent the damage caused from the noisy sentences (and more consistency check for quantity meaning should be done). The remaining errors are due to incorrect quantity extraction, lacking common-sense or not knowing entailment relationship between two entities.

A similar experiment is performed to check if the DNN approach will be affected by the noisy information more. We first select 124 MWPs (denoted as OSS') from OSS that can be correctly solved both in our statistical and DNN approaches and then filter out 350 derived MWPs (denotes as NDS') from NDS. The right-hand side of Table 4 displays the performance of DNN approach drops more than the statistical approach in the noisy dataset, indicating that our statistical approach is less sensitive to the irrelevant quantities and thus able to capture more meaningful information.

5 Related Work

To the best of our knowledge, MWP solvers proposed before 2014 all adopted the rule-based approach. Mukherjee and Garain (2008) had given a good survey for all related approaches before 2008. Afterwards, Ma et al. (2010) proposed a MSWPAS system to simulate human arithmetic multi-step addition and subtraction behavior without evaluation. Besides, Liguda and Pfeiffer (2012) proposed a model based on augmented semantic networks, and claimed that it could solve multi-step MWPs and complex equation systems and was more robust to irrelevant information (also no evaluation).

Recently, Hosseini et al. (2014) proposed a Container-Entity based approach, which solved the MWP with a sequence of state transition. And Kushman et al. (2014) proposed the first statistical approach, which heuristically extracts some algebraic templates from labeled equations, and then aligns them with the given sentence. Since no semantic analysis is conducted, the performance is quite limited.

In more recent researches (Roy and Roth, 2015; Koncel-Kedziorz ski et al., 2015), quantities in an MWP were associated with attributes extracted from their contexts. Based on the attributes, several statistical classifiers were used to select operands and determine operators to solve multi-step MWPs. Since the physical meaning of each quantity is not explicitly considered in getting the answer, the reasoning process cannot be explained in a human comprehensible way. Besides, Shi et al. (2015) attacked the number word problem, which only deal with numbers, with a semantic parser. Mitra and Baral (2016) mapped MWPs into three types of problems, including Part-Whole, Change and Comparison. Each problem was associated with a generic formula. They used a log-linear model to determine how to instantiate the formula with quantities and solve the only one Unknown variable. They achieved the best performance on the A12 dataset. However, their approach cannot handle Multiplication or Division related MWPs. Recently, DNN-based approaches (Ling et al, 2017; Wang et al, 2017) have emerged. However, they only attacked algebraic word problems, and required a very large training-set.

Our proposed approach mainly differs from those previous approaches in combining the statistical framework with logic inference, and also in adopting the meaning-based statistical approach for selecting the desired operands.

6 Conclusion

A meaning-based logic form represented with role-tags (e.g., nsubj, verb, etc.) is first proposed to associate the extracted math quantity with its physical meaning, which then can be used to identify the desired operands and filter out irrelevant quantities. Afterwards, a statistical framework is proposed to perform understanding and reasoning based on those logic expressions. We further compare the performance with a typical DNN approach, the results show the proposed approach is still better. We will try to integrate domain concepts into the DNN approach to improve the learning efficiency in the future.

The main contributions of our work are: (1) Adopting a meaning-based approach to solve English math word problems and showing its superiority over other state-of-the art systems on common datasets. (2) Proposing a statistical model to select operands by explicitly checking the meanings of quantities against the meaning of the question sentence. (3) Designing a noisy dataset to test if a system solves the problems by understanding. (4) Proposing a perplexity-flavor measure to assess the complexity of a dataset.
References


Peter Clark and Oren Etzioni. 2016. My Computer is an Honor Student - but how Intelligent is it? Standardized Tests as a Measure of AI. AI Magazine, pages 5–12.


Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL Demonstrations*.


Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about quantities in natural language. Trans-


