Learning to Solve Arithmetic Word Problems with Verb Categorization

Mohammad Javad Hosseini[1], **Hannaneh Hajishirzi**[1], **Oren Etzioni**[2], **and Nate Kushman**[3]

[1]{hosseini, hannaneh}@washington.edu, [2]OrenE@allenai.org, [3]nkushman@csail.mit.edu
[1]University of Washington, [2]Allen Institute for AI, [3]Massachusetts Institute of Technology

Abstract

This paper presents a novel approach to learning to solve simple arithmetic word problems. Our system, ARIS, analyzes each of the sentences in the problem statement to identify the relevant variables and their values. ARIS then maps this information into an equation that represents the problem, and enables its (trivial) solution as shown in Figure 1. The paper analyzes the arithmetic-word problems "genre", identifying seven categories of verbs used in such problems. ARIS learns to categorize verbs with 81.2% accuracy, and is able to solve 77.7% of the problems in a corpus of standard primary school test questions. We report the first learning results on this task without reliance on predefined templates and make our data publicly available.[1]

1 Introduction

Designing algorithms to automatically solve math and science problems is a long-standing AI challenge (Feigenbaum and Feldman, 1963). For NLP, mathematical word problems are particularly attractive because the text is concise and relatively straightforward, while the semantics reduces to simple equations.

Arithmetic word problems begin by describing a partial world state, followed by simple updates or elaborations and end with a quantitative question. For a child, the language understanding part is trivial, but the reasoning may be challenging; for our system, the opposite is true. ARIS needs to make sense of multiple sentences, as shown in Figure 2, without *a priori* restrictions on the syntax or vocabulary used to describe the problem. Figure 1 shows an example where ARIS is asked to infer how many kittens Joan received based on facts and constraints expressed in the text, and represented by the state diagram and corresponding equation. While the equation is trivial, the text could have involved assembling toy aircraft, collecting coins, eating cookies, or just about any activity involving changes in the quantities of discrete objects.

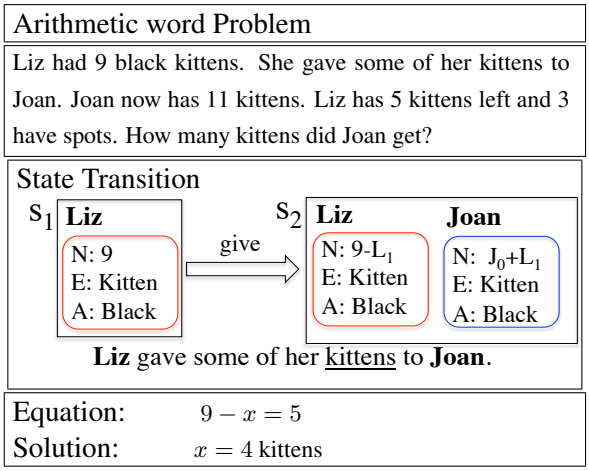

Figure 1: Example problem and solution.

This paper investigates the task of learning to solve such problems by mapping the verbs in the problem text into categories that describe their impact on the world state. While the verbs category is crucial (e.g., what happens if "give" is replaced by "receive" in Figure 1?), some elements of the problem are irrelevant. For instance, the fact that three kittens have spots is immaterial to the solution. Thus, ARIS has to determine what information is relevant to solving the problem.

To abstract from the problem text, ARIS maps the text to a state representation which consists of

[1]Our data is available at `https://www.cs.washington.edu/nlp/arithmetic`.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 523–533,
October 25-29, 2014, Doha, Qatar.

a set of entities, their containers, attributes, quantities, and relations. A problem text is split into fragments where each fragment corresponds to an observation or an update of the quantity of an entity in one or two containers. For example in Figure 1, the sentence "Liz has 5 kittens left and 3 have spots" has two fragments of "Liz has 5 kittens left" and "3 have spots".

The verb in each sentence is associated with one or two containers, and ARIS has to classify each verb in a sentence into one of seven categories that describe the impact of the verb on the containers (Table 1). ARIS learns this classifier based on training data as described in section 4.2.

To evaluate ARIS, we compiled a corpus of about 400 arithmetic (addition and subtraction) word problems and utilized cross validation to both train ARIS and evaluate its performance over this corpus. We compare its performance to the template-based learning method developed independently and concurrently by Kushman et al. (2014). We find that our approach is much more robust to domain diversity between the training and test sets.

Our contributions are three-fold: (a) We present ARIS, a novel, fully automated method that learns to solve arithmetic word problems; (b) We introduce a method to automatically categorize verbs for sentences from simple, easy-to-obtain training data; our results refine verb senses in WordNet (Miller, 1995) for arithmetic word problems; (c) We introduce a corpus of arithmetic word problems, and report on a series of experiments showing high efficacy in solving addition and subtraction problems based on verb categorization.

2 Related Work

Understanding semantics of a natural language text has been the focus of many researchers in natural language processing (NLP). Recent work focus on learning to align text with meaning representations in specific, controlled domains. A few methods (Zettlemoyer and Collins, 2005; Ge and Mooney, 2006) use an expensive supervision in the form of manually annotated formal representations for every sentence in the training data. More recent work (Eisenstein et al., 2009; Kate and Mooney, 2007; Goldwasser and Roth, 2011; Poon and Domingos, 2009; Goldwasser et al., 2011; Kushman and Barzilay, 2013) reduce the amount of required supervision in mapping sentences to meaning representations while taking advantage of special properties of the domains. Our method, on the other hand, requires small, easy-to-obtain training data in the form of verb categories that are shared among many different problem types.

Our work is also closely related to the grounded language acquisition research (Snyder and Barzilay, 2007; Branavan et al., 2009; Branavan et al., 2012; Vogel and Jurafsky, 2010; Chen et al., 2010; Hajishirzi et al., 2011; Chambers and Jurafsky, 2009; Liang et al., 2009; Bordes et al., 2010) where the goal is to align a text into underlying entities and events of an environment. These methods interact with an environment to obtain supervision from the real events and entities in the environment. Our method, on the other hand, grounds the problem into world state transitions by learning to predict verb categories in sentences. In addition, our method combines the representations of individual sentences into a coherent whole to form the equations. This is in contrast with the previous work that study each sentence in isolation from the other sentences.

Previous work on studying math word and logic problems uses manually aligned meaning representations or domain knowledge where the semantics for all the words is provided (Lev, 2007; Lev et al., 2004). Most recently, Kushman et al. (2014) introduced an algorithm that learns to align algebra problems to equations through the use of templates. This method applies to broad range of math problems, including multiplication, division, and simultaneous equations, while ARIS only handles arithmetic problems (addition and subtraction). However, our empirical results show that for the problems it handles, ARIS is much more robust to diversity in the problem types between the training and test data.

3 Arithmetic Problem Representation

We address solving arithmetic word problems that include addition and subtraction. A problem text is split into fragments where each fragment is represented as a transition between two world states in which the quantities of entities are updated or observed (Figure 2). We refer to these fragments as sentences. We represent the world state as a tuple $\langle E, C, R \rangle$ consisting of entities E, containers C, and relations R among entities, containers, attributes, and quantities.

Entities: An *entity* is a mention in the text corre-

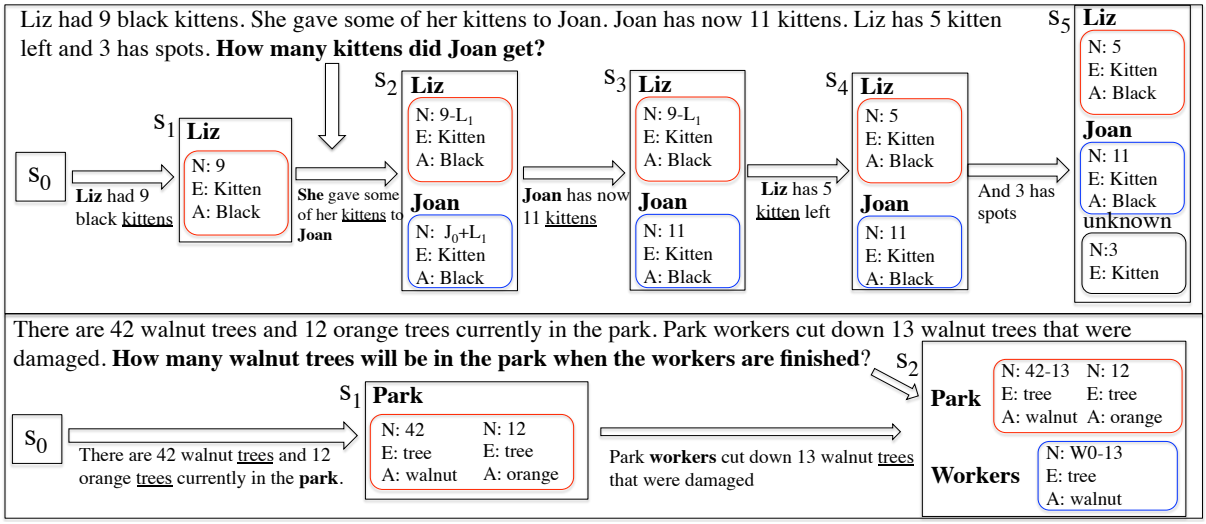

Figure 2: A figure sketching different steps of our method — a sequence of states.

sponding to an object whose quantity is observed or is changing throughout the problem. For instance, `kitten` and `tree` are entities in Figure 2. In addition, every entity has *attributes* that modify the entity. For instance, `black` is an attribute of `kittens`, and `walnut` is an attribute of `tree` (more details on attributes in section 4.1). Relations describing attributes are invariant to the state changes. For instance kittens stay `black` throughout the problem of Figure 1.

Containers: A *container* is a mention in the text representing a set of entities. For instance, `Liz`, `Joan`, `park`, and `workers` are containers in Figure 2. Containers usually correspond to the *person* possessing entities or a *location* containing entities. For example, in the sentence "There are 43 blue marbles in the basket. John found 32 marbles.", `basket` and `John` are containers of `marbles`.

Quantities: Containers include entities with their corresponding quantities in a particular world state. Quantities can be known numbers (e.g. 9), unknown variables (e.g. L_1), or numerical expressions over unknown quantities and numbers (e.g. $9-L_1$). For instance, in state 2 of Figure 2, the numerical expression corresponding to `Liz` is $9-L_1$ and corresponding to `Joan` is $J_0 + L_1$, where J_0 is a variable representing the number of `kittens` that `Joan` has started with.

Hereinafter, we will refer to a generic entity as e, container as c, number as num, attribute as a. We represent the relation between a container, an entity, and a number in the form of a quantity expression `N(c,e)`. Figure 2 shows the quantity relations in different world states.

Category	Example
Observation	There **were** 28 bales of hay in the *barn*.
Positive	*Joan* **went** to 4 football games this year.
Negative	*John* **lost** 3 of the violet balloons.
Positive Transfer	*Mike's dad* **borrowed** 7 nickels from *Mike*.
Negative Transfer	*Jason* **placed** 131 erasers in the *drawer*.
Construct	*Karen* **added** 1/4 of a cup of walnuts to a *batch* of trail mix.
Destroy	The *rabbits* **ate** 4 of *Dan's* potatoes.

Table 1: Examples for different verb categories in sentences. Entities are underlined; *containers* are italic, and **verbs** are bolded.

State transitions: Sentences depict progression of the world state (Figure 2) in the form of observations of updates of quantities. We assume that every sentence w consists of a verb v, an entity e, a quantity num (might be unknown), one or two containers c_1, c_2, and attributes a. The presence of the second container, c_2, will be dictated by the category of the verb, as we discuss below. Sentences abstract transitions $(s_t \rightarrow s_{t+1})$ between states in the form of an algebraic operation of addition or subtraction. For every sentence, we model the state transition according to the verb category and containers in the sentence. There are three verb categories for sentences with one container: *Observation*: the quantity is initialized in the container, *Positive*: the quantity is increased in the container, and *Negative*: the quantity is decreased in the container. Moreover, there are four categories for sentences with two containers: *Pos-*

itive transfer: the quantity is transferred from the second container to the first one, *Negative transfer*: the quantity is transferred from the first container to the second one, *Construct*: the quantity is increased for both containers, and *Destroy*: the quantity is decreased for both containers.

Figure 2 shows how the state transitions are determined by the verb categories. The sentence "Liz has 9 black kittens" initializes the quantity of `kittens` in the container `Liz` to `9`. In addition, the sentence "She gave some of her kittens to Joan." shows the negative transfer of L_1 kittens from `Liz` to `Joan` represented as `N(Liz,kitten)=9-`L_1 and `N(Joan,kitten)=`$J_0 + L_1$.

Given a math word problem, ARIS grounds the world state into entities (e.g., `kitten`), containers (e.g., `Liz`), attributes (e.g., `black`), and quantities (e.g., `9`) (Section 4.1). In addition, ARIS learns state transitions by classifying verb categories in sentences (Section 4.2). Finally, from the world state and transitions, it generates an arithmetic equation which can be solved to generate the numeric answer to the word problem.

4 Our Method

In this section we describe how ARIS maps an arithmetic word problem into an equation (Figure 2). ARIS consists of three main steps (Figure 3): (1) grounding the problem into entities and containers, (2) training a model to classify verb categories in sentences, and (3) solving the problem by updating the world states with the learned verb categories and forming equations.

4.1 Grounding into Entities and Containers

ARIS automatically identifies entities, attributes, containers, and quantities corresponding to every sentence fragment (details in Figure 3 step 1). For every problem, this module returns a sequence of sentence fragments $\langle w_1, \ldots, w_T, w_x \rangle$ where every w_t consists of a verb v_t, an entity e_t, its quantity num_t, its attributes a_t, and up to two containers c_{t_1}, c_{t_2}. w_x corresponds to the question sentence inquiring about an unknown entity. ARIS applies the Stanford dependency parser, named entity recognizer and coreference resolution system to the problem text (de Marneffe et al., 2006; Finkel et al., 2005; Raghunathan et al., 2010). It uses the predicted coreference relationships to replace pronouns (including possessive pronouns) with their coreferenent links. The named entity recognition output is used to identify numbers and people.

Entities: Entities are references to some object whose quantity is observed or changing throughout the problem. So to determine the set of entities, we define h as the set of noun types which have a dependent number (in the dependency parse) somewhere in the problem text. The set of entities is then defined as all noun phrases which are headed by a noun type in h. For instance `kitten` in the first sentence of Figure 1 is an entity because it is modified by the number `9`, while `kitten` in the second sentence of Figure 1 is an entity because `kitten` was modified by a number in the first sentence. Every number in the text is associated with one entity. Numbers which are dependents of a noun are associated with its entity. Bare numbers (not dependent on a noun) are associated with the previous entity in the text. The entity in the last sentence is identified as the question entity e_x . Finally, ARIS splits the problem text into $T + 1$ sentence fragments $\langle w_1, \ldots w_T, w_x \rangle$ such that each fragment contains a single entity and it's containers. For simplicity we refer to these fragments as a sentences.

Containers: Each entity is associated with one or two container noun phrases using the algorithm described in in Figure 3 step 1c. As we saw earlier with numbers, arithmetic problems often include sentences with missing information. For example in Figure 2, the second container in the the sentence "Park workers had to cut down 13 walnut trees that were damaged." is not explicitly mentioned. To handle this missing information, we use the *circumscription* assumption (McCarthy, 1980). The circumscription assumption formalizes the commonsense assumption that things are as expected unless otherwise specified. In this setting, we assume that the set of containers are fixed in a problem. Thus if the container(s) for a given entity cannot be identified they are set to the container(s) for the previous entity with the same head word. For example in Figure 2 we know from the previous sentence that `trees` were in the `park`. Therefore, we assume that the unmentioned container is the `park`.

Attributes: ARIS selects attributes A as modifiers for every entity from the dependency parser (details in Figure 3 step 1a). For example `black` is an attribute of the entity `kitten` and is an adjective modifier in the parser. These attributes are

1. **Grounding into entities and containers: for every problem p in dataset** (Section 4.1)
   (a) $\langle e_1, \ldots, e_T, e_x \rangle_p \leftarrow$ extract all entities and the question entity
      i. Extract all numbers and noun phrases (NP).
      ii. $h \leftarrow$ all noun types which appear with a number as a dependant (in the dependency parse tree) somewhere in the problem text.
      iii. $e_t \leftarrow$ all NPs which are headed by a noun type in h.
      iv. $num_t \leftarrow$ the dependant number of e_t if one exists. Bare numbers (not directly dependant on any noun phrase) are associated with the previous entity in the text. All other num_t are set to unknown.
      v. $e_x \leftarrow$ the last identified entity.
      vi. $a_t \leftarrow$ adjective and noun modifiers of e_t. Update implicit attributes using the previously observed attributes.
      vii. $v_t \leftarrow$ the verb with the shortest path to e_t in the dependency parse tree.
   (b) $\langle w_1, \ldots, w_T, w_x \rangle_p \leftarrow$ split the problem text into fragments based on the entities and verbs
   (c) $\langle c_{t_1}, c_{t_2}, \ldots, c_{T_1}, c_{T_2}, c_x \rangle_p \leftarrow$ the list of containers for each entity
      i. $c_{t_1} \leftarrow$ the subject of w_t.
      If w_t contains *There is/are*, c_{t_1} is the first adverb of place to the verb.
      ii. $c_{t_2} \leftarrow$ An NP that is direct object of the verb. If not found, c_{t_2} is the object of the first adverbial phrase of the verb.
      iii. Circumscription assumption: When c_{t_1} or c_{t_2} are not found, they are set to the previous containers.

2. **Training for sentence categorization** (Section 4.2)
   (a) $instances_1, instances_2 \leftarrow \emptyset$
   (b) for every sentence $w_t \in \langle w_1, \ldots, w_T, w_x \rangle_p$ in the training set:
      i. $features_t \leftarrow$ extract features (similarity based, WordNet based, structural) (Section 4.2.1)
      ii. $l_{t_1}, l_{t_2} \leftarrow$ determine labels for containers c_{t_1} and c_{t_2} based on the verb category of w_t.
      iii. append $\langle features_t, l_{t,1} \rangle$, $\langle features_t, l_{t,2} \rangle$ to $instances_1, instances_2$.
   (c) $M_1, M_2 \leftarrow$ train two SVMs for $instances_1, instances_2$

3. **Solving: for every problem p in the test set** (Section 4.3)
   (a) **Identifying verb categories in sentences**
      i. for every sentence $w_t \in \langle w_1, \ldots, w_T, w_x \rangle_p$:
         A. $features_t \leftarrow$ extract features (similarity based, WordNet based, structural).
         B. $l_{t_1}, l_{t_2} \leftarrow$ classify w_t for both containers c_{t_1} and c_{t_2} using models M_1, M_2.
   (b) **State progression**: Form $\langle s_0, \ldots, s_T \rangle$ (Section 4.3.1)
      i. $s_0 \leftarrow null$.
      ii. for $t \in \langle 1, \ldots, T \rangle$: $s_t \leftarrow progress(s_{t-1}, w_t)$.
         A. if $e_t = e_x$ and $a_t = a_x$:
            if w_t is an observation: $N_t(c_{t_1}, e_t) = num_t$.
            else: update $N_t(c_{t_1}, e_t)$ and $N_t(c_{t_2}, e_t)$ given verb categories l_{t_1}, l_{t_2}.
         B. copy $N_{t-1}(c, e)$ to $N_t(c, e)$ for all other (c, e) pairs.
   (c) **Forming equations and solution** (Section 4.3.2)
      i. Mark each w_t that matches with w_x if:
      a) c_{t_1} matches with c_x and verb categories are equal or verbs are similar.
      b) c_{t_2} matches with c_x and the verbs are in opposite categories.
      ii. $x \leftarrow$ the unknown quantity if w_x matches with a sentence introducing an unknown number
      iii. If the question asks about an unknown variable x or a start variable (w_x contains "begin" or "start"):
      For some container c, find two states s_t (quantity expression contains x) and s_{t+1} (quantity is a known number). Then, form an equation for x: $N_t(c, e_x) = N_{t+1}(c, e_x)$.
      iv. else: form equation as $x = N_t(c_x, e_x)$.
      v. Solve the equation and return the absolute value of x.

Figure 3: ARIS: a method for solving arithmetic word problems.

used to prune the irrelevant information in progressing world states.

Arithmetic problems usually include sentences with no attributes for the entities. For example, the attribute `black` has not been explicitly mentioned for the `kitten` in the second sentence. In particular, ARIS updates an implicit attribute using the previously observed attribute. For example, in "Joan went to 4 football games this year. She went to 9 games last year.", ARIS assigns `football` as an attribute of the `game` in both sentences.

4.2 Training for Verb Categories

This step involves training a model to identify verb categories for sentences. This entails predicting one label (increasing, decreasing) for each (verb, container) pair in the sentence. Each possible setting of these binary labels corresponds to one of the seven verb categories discussed earlier. For example, if c_1 is increasing and c_2 is decreasing this is a *positive transfer* verb.

Our dataset includes word problems from different domains (more details in Section 5.2). Each verb in our dataset is labeled with one of the 7 cat-

egories from Table 1.

For training, we compile a list of sentences from all the problems in the dataset and split sentences into training and test sets in two settings. In the first setting no instance from the same domain appears in the training and test sets in order to study the robustness of our method to new problem types. In the second setting no verb is repeated in the training and test sets in order to study how well our method predicts categories of unseen verbs.

For every sentence w_t in the problems, we build two data instances, (w_t, c_1) and (w_t, c_2), where c_1 and c_2 are containers extracted from the sentence. For every instance in the training data, we assign training labels using the verb categories of the sentences instead of labeling every sentence individually. The verb can be increasing or decreasing corresponding to every container in the sentence. For positive (negative) and construction (destruction) verbs, both instances are labeled positive (negative). For transfer positive (negative) verbs, the first instance is labeled positive (negative) and the second instance is labeled negative (positive). For observation verbs, both instances are labeled positive. We assume that the observation verbs are known (total of 5 verbs). Finally, we train Support Vector Machines given the extracted features and training labels explained above (Figure 3 step 2). In the following, we describe the features used for training.

4.2.1 Features

There are three sets of features: similarity based, Wordnet-based, and structural features. The first two sets of features focus on the verb and the third set focuses on the dependency structure of the sentence. All of our features are unlexicalized. This allows ARIS to handle verbs in the test questions which are completely different from those seen in the training data.

Similarity-based Features: For every instance (w, c), the feature vector includes similarity between the verb of the sentence w and a list of seed verbs. The list of seed verbs is automatically selected from a set $\mathcal{V}$ containing the 2000 most common English verbs using ℓ_1 regularized feature selection technique. We select a small set of seed verbs to avoid dominating the other feature types (structural and WordNet-based features).

The goal is to automatically select verbs from $\mathcal{V}$ that are most discriminative for each of the 7 verb categories in Table 1. We define 7 classification tasks: "Is a verb a member of each category?" Then, we select the three most representative verbs for each category. To do so, we randomly select a set of 65 verbs $\mathcal{V}_l$, from all the verbs in our dataset (118 in total) and manually annotate the verb categories. For every classification task, the feature vector $\mathbf{X}$ includes the similarity scores (Equation 1) between the verb v and all the verbs in the $\mathcal{V}$. We train an ℓ_1 regularized regression model (Park and Hastie, 2007) over the feature vector $\mathbf{X}$ to learn each category individually. The number of original (similarity based) features in $\mathbf{X}$ is relatively large, but ℓ_1 regularization provides a sparse weight vector. ARIS then selects the three most common verbs (without replacement) among the features (verbs) with non-zero weights. This accounts for 21 total seed verbs to be used for the main classification task. We find that in practice using this selection technique leads to better performance than using either all the verbs in $\mathcal{V}$ or using just the 65 randomly selected verbs.

Our method computes the similarity between two verbs v_1 and v_2 from the similarity between all the senses (from WordNet) of these verbs (Equation 1). We compute the similarity between two senses using linear similarity (Lin, 1998). The similarity between two synsets sv_1 and sv_2 are penalized according to the order of each sense for the corresponding verb. Intuitively, if a synset appears earlier in the set of synsets of a verb, it is more likely to be considered as the correct meaning. Therefore, later occurrences of a synset should result in reduced similarity scores. The similarity between two verbs v_1 and v_2 is the maximum similarity between two synsets of the verbs:

$$\text{sim}(v_1, v_2) = \max_{sv:\text{synsets}(v)} \frac{\text{lin-sim}(sv_1, sv_2)}{log(p_1 + p_2)} \quad (1)$$

where sv_1, sv_2 are two synsets, p_1, p_2 are the position of each synset match, and lin-sim is the linear similarity. Our experiments show better performance using linear similarity compared to other common similarity metrics (e.g., WordNet path similarity and Resnik similarity (Resnik, 1995)).

WordNet-based Features: We use WordNet verb categories in the feature vector. For each part of speech in WordNet, the synsets are organized into different categories. There are 15 categories for verbs. Some examples in-

clude "verb.communication", "verb.possession", and "verb.creation". In addition, WordNet includes the frequency measure $f_{c_{sv}}$ indicating how often the sense sv has appeared in a reference corpus. For each category i, we define the feature f_i as the ratio of the frequency of the sense sv_i over the total frequency of the verb i.e., $f_i = f_{c_{sv_i}}/f_{c_v}$.

Structural Features: For structural features, we use the dependency relations between the verb and the sentence elements since they can be a good proxy of the sentence structure. ARIS uses a binary vector including 35 dependency relations between the verb and other elements. For example, in the sentence "Joan picked 2 apples from the apple tree", the dependency between ('picked' and 'tree') and ('picked' and 'apples') are depicted as 'prep-from' and 'dobj' relations in the dependency parser, respectively. In addition, we include the length of the path in the dependency parse from the entity to the verb.

4.3 Solving the Problem

So far, ARIS grounds every problem into entities, containers, and attributes, and learns verb categories in sentences. Solving the problem consists of two main steps: (1) progressing states based on verb categories in sentences and (2) forming the equation.

4.3.1 State Progression with Verb Categories

This step (Figure 3 step 3b) involves forming states $\langle s_1, \ldots, s_T \rangle$ by updating quantities in every container using learned verb categories (Figure 3 step 3a). ARIS initializes s_0 to an empty state. It then iteratively updates the state s_t by progressing the state s_{t-1} given the sentence w_t with the verb v, entity e, number num, and containers c_1 and c_2.

For a given sentence t, ARIS attempts to match e_t and c_t to entities and categories in s_{t-1}. An entity/category is matched if has the same head word and same set of attributes as an existing entity/category. If an entity or category cannot be matching to one in s_{t-1}, then a new one is created in s_t.

The progress subroutine prunes the irrelevant sentences by checking if the entity e and its attributes a agree with the question entity e_x and its attributes a_x in the question. For example both `game` entities agree with the question entity in the problem "Joan went to 4 football games this year. She went to 9 games last year. How many football games did Joan go?". The first entity has an explicit `football` attribute, and the second entity has been assigned the same attribute (Section 4.1). Even if the question asks about `games` without mentioning `football`, the two sentences will match the question. Note that the second sentence would have not been matched if there was an explicit mention of the 'basketball game' in the second sentence.

For the matched entities, ARIS initializes or updates the values of the containers c_1, c_2 in the state s_t. ARIS uses the learned verb categories in sentences (Section 4.2) to update the values of containers. For an observation sentence w_t, the value of c_1 in the state s_t is assigned to the observed quantity num. For other sentence types, if the container c does not match to a container the previous state, its value is initialized with a start variable C_0. For example, the container `Joan` is initialized with J_0 at the state s_1 (Figure 2). Otherwise, the values of c_1 and c_2 are updated according to the verb category in the sentence. For instance, if the verb category in the sentence is a positive transfer then $N_t(c_1, e) = N_{t-1}(c_1, e) - num$ and $N_t(c_2, e) = N_{t-1}(c_2, e) + num$ where $N_t(c, e)$ represents the quantity of e in the container c at state s_t (Figure 2).

4.3.2 Forming Equations and Solution

The question entity e_x can match either to an entity in the final state, or to some unknown generated during the state progression. Concretely, the question sentence w_x asks about the quantity x of the entity e_x in a container c_x at a particular state s_u or a transition after the sentence w_u (Figure 3 step 3c).

To determine if e_x matches to an unknown variable, we define a matching subroutine between the question sentence w_x and every sentence w_t to check entities, containers, and verbs (Figure 3 step 3(c)i). We consider two cases. 1) When w_x contains the words "begin", or "start", the unknown variable is about the initial value of an entity, and it is set to the start variable of the container c_x (Figure 3 step 3(c)iii). For example, in "Bob had balloons. He gave 9 to his friends. He now has 4 balloons. How many balloons did he have to start with?", the unknown variable is set to the start variable B_0. 2) When the question verb is not one of the defined set of observation verbs, ARIS attempts to match e_x with an unknown introduced by one of the state transitions (Figure 3

step 3(c)iii). For example, the second sentence in Figure 1 introduces an unknown variable over `kittens`. The matching subroutine matches this entity with the question entity since the question container, i.e. `Joan`, matches with the second container and verb categories are complementary.

In order to solve for the unknown variable x, ARIS searches through consecutive states s_t and s_{t+1}, where in s_t, the quantity of e_x for a container c is an expression over x, and in s_{t+1}, the quantity is a known number for a container matched to c. It then forms an equation by comparing the quantities for containers matched between the two states. In the previous example, the equation will be $B_0 - 9 = 4$ by comparing states s_2 and s_3, where the numerical expression over `balloons` is $B_0 - 9$ in the state s_2, and the quantity is a known number in the state s_3.

When neither of the two above cases apply, ARIS matches e_x to an entity in the final state, s_T and returns its quantity, (Figure 3 step 3(c)iv). In the `football` example of the previous section, the equation will be $x = N_t(c_x, e_x)$, where $N_t(c_x, e_x)$ is the quantity in the final state.

Finally, the equation will be solved for the unknown variable x and the absolute value of the unknown variable is returned.

5 Experiments

To experimentally evaluate our method we build a dataset of arithmetic word problems along with their correct solutions. We test our method on the accuracy of solving arithmetic word problems and identifying verb categories in sentences.

5.1 Experimental Setup

Datasets: We compiled three diverse datasets MA1, MA2, IXL (Table 2) of Arithmetic word problems on addition and subtraction for third, fourth, and fifth graders. These datasets have similar problem types, but have different characteristics. Problem types include combinations of additions, subtractions, one unknown equations, and U.S. money word problems. Problems in MA2 include more irrelevant information compared to the other two datasets, and IXL includes more information gaps. In total, they include 395 problems, 13,632 words, 118 verbs, and 1,483 sentences.

Tasks and Baselines: We evaluate ARIS on two tasks: 1) solving arithmetic word problems in the three datasets and 2) classifying verb categories in sentences. We use the percentage of correct answers to the problems as the evaluation metric for the first task and accuracy as the evaluation metric for the second task. We use Weka's SVM (Witten et al., 1999) with default parameters for classification which is trained with verb categories in sentences (as described in Section 4.2).

	Source	#Tests	Avg.# Sentences
MA1	math-aids.com	134	3.5
IXL	ixl.com	140	3.36
MA2	math-aids.com	121	4.48

Table 2: Properties of the datasets.

	MA1	IXL	MA2	Total
3-fold Cross validation				
ARIS	83.6	**75.0**	**74.4**	**77.7**
ARIS$_2$	83.9	75.4$^+$	69.8$^+$	76.5$^+$
KAZB	**89.6**	51.1	51.2	64.0
Majority	45.5	71.4	23.7	48.9
Gold sentence categorization				
Gold ARIS	94.0	77.1	81.0	84.0

Table 3: Accuracy of solving arithmetic word problems in three datasets MA1, IXL, and MA2. This table compares our method, ARIS, ARIS$_2$ with the state-of-the-art KAZB. All methods are trained on two (out of three) datasets and tested on the other one. ARIS$_2$ is trained when no verb is repeated in the training and test sets. Gold ARIS uses gold verb categories. The improvement of ARIS (boldfaced) and ARIS$_2$ (denoted by $^+$) are significant over KAZB and the majority baseline with $p < 0.05$.

For the first task, we compare ARIS with KAZB (Kushman et al., 2014), majority baseline, ARIS$_2$, and Gold ARIS. KAZB requires training data in the form of equation systems and numerical answers to the problems. The majority baseline classifies every instance as increasing. In ARIS$_2$ (a variant of ARIS) the system is trained in a way that no verb is repeated in the training and test sets. Gold ARIS uses the ground-truth sentence categories instead of predicted ones. For the second task, we compare ARIS with a baseline that uses WordNet verb senses.

5.2 Results

We evaluate ARIS in solving arithmetic word problems in the three datasets and then evaluate its ability in classifying verb categories in sentences.

5.2.1 Solving Arithmetic Problems

Table 3 shows the accuracy of ARIS in solving problems in each dataset (when trained on the other two datasets).Table 3 shows that ARIS

significantly outperforms KAZB and the majority baseline. As expected, ARIS shows a larger gain on the two more complex datasets MA2 and IXL; our method shows promising results in dealing with irrelevant information (dataset MA2) and information gaps (dataset IXL). This is because ARIS learns to classify verb categories in sentences and does not require observing similar patterns/templates in the training data. Therefore, ARIS is more robust to differences between the training and test datasets and can generalize across different dataset types. As discussed in the experimental setup, the datasets have mathematically similar problems, but differ in the natural language properties such as in the sentence length and irrelevant information (Table 2).

Table 3 also shows that the sentence categorization is performed with high accuracy even if the problem types and also the verbs are different. In particular, there are a total of 118 verbs among which 64 verbs belong to MA datasets and 54 are new to IXL. To further study this, we train our method ARIS_2 in which no verb can be repeated in the training and test sets. ARIS_2 still significantly outperforms KAZB. In addition, we observe only a slight change in accuracy between ARIS and ARIS_2.

To further understand our method, we study the effect of verb categorization in sentences in solving problems. Table 3 shows the results of Gold ARIS in solving arithmetic word problems with gold sentence categorizations. In addition, comparing ARIS with Gold ARIS suggests that our method is able to reliably identify verb categories in sentences.

We also perform an experiment where we pool all of the problems in the three datasets and randomly choose 3 folds for the data (instead of putting each original dataset into it's own fold). We compare our method with KAZBin this scenario. In this setting, our method's accuracy is 79.5% while KAZB's accuracy is 81.8%. As expected, our method's performance has not changed significantly from the previous setting, while KAZB's performance significantly improves because of the reduced diversity between the training and test sets in this scenario.

5.2.2 Sentence Categorization

Table 4 compares accuracy scores of sentence categorization for our method with different features, a baseline that uses WordNet verb senses, and the majority baseline that assigns every (verb, container) pair as increasing. Similar to ARIS_2, we randomly split verbs into three equal folds and assign the corresponding sentences to each fold. No verb is shared between training and test sets. We then directly evaluate the accuracy of the SVM's verb categorization (explained in Section 4.2). This table shows that ARIS performs well in classifying sentence categories even with new verbs in the test set. This suggests that our method can generalize well to predict verb categories for unseen verbs.

Table 4 also details the performance of four variants of our method that ablate various features of ARIS. The table shows that similarity, contextual, and WordNet features are all important to the performance of ARIS in verb categorization, whereas the WordNet features are less important for solving the problems. In addition, it shows that similarity features play more important roles. We also performed another experiment to study the effect of the proposed feature selection method for similarity-based features. The accuracy of ARIS in classifying sentence categories is 69.7% when we use all the verbs in $\mathcal{V}$ in the similarity feature vector. This shows that our feature selection algorithm for selecting seed verbs is important towards categorizing verbs.

Finally, Table 4 shows that our method significantly outperforms the baseline that only uses WordNet verb sense. An interesting observation is that the majority baseline in fact outperforms WordNet verb senses in verb categorization, but is significantly worse in solving arithmetic word problems. In addition, we evaluate the accuracy of predicting only verb categories by assigning the verb label according to the majority of its labels in the sentence categories. The accuracy of verb categories is 78.2% confirming that ARIS is able to successfully categorize verbs.

5.2.3 Error Analysis

We analyzed all 63 errors of Gold ARIS and present our findings in Table 5. There are five major classes of errors. In the first category, some information is not mentioned explicitly and should be entailed. For example, 'washing cars' is the source of 'making money'. Despite the improvements that come from ARIS, a large portion of the errors can still be attributed to irrelevant information. For example, 'short' is not a 'toy'. The third category refers to errors that require knowledge

	Categorization	Solution
ARIS	**81.2**$^+$	**76.5**$^+$
No similarity features	68.8	65.4
No WordNet features	75.3	78.0$^+$
No structural features	75.5	72.4$^+$
Baseline (WordNet)	67.8	68.4
Majority Baseline	73.4	48.9

Table 4: Ablation study and baseline comparisons: this table reports the accuracy of verb categorization in sentences and solutions for ARIS with ablating features. It also provides comparisons to WordNet and majority baselines. The improvement of ARIS (boldfaced) and ablations denoted by $^+$ are statistically significant over the baselines (with $p < 0.05$) for both tasks.

Error type	Example
Entailment, Implicit Action (26%)	Last week Tom had $74. He **washed cars** over the weekend and now has $86. How much money did he make washing cars?
Irrelevant Information (19%)	Tom bought a skateboard for $9.46, and spent $9.56 on marbles. Tom also spent $14.50 on **shorts**. In total, how much did Tom spend on **toys**?
Set Completion (13%)	Sara's school **played** 12 games this year. They **won** 4 games. How many games did they **lose**?
Parsing Issues (21%)	Sally had 27 Pokemon cards. Dan gave **her** 41 new Pokemon cards. How many Pokemon cards does Sally have now?
Others (21%)	In March it rained 0.81 inches. It rained 0.35 inches **less** in April than in March. How much did it rain in April?

Table 5: Examples of different error categories and relative frequencies. The cause of error is **bolded**.

about set completions. For example, the 'played' games can be split into 'win' and 'lost' games. Finally, parsing and coreference mistakes are another source of errors for ARIS.

6 Discussions and Conclusion

In this paper we introduce ARIS, a method for solving arithmetic word problems. ARIS learns to predict verb categories in sentences using syntactic and (shallow) semantic features from small, easy-to-obtain training data. ARIS grounds the world state into entities, sets, quantities, attributes, and their relations and takes advantage of the circumscription assumption and successfully fills in the information gaps. Finally, ARIS makes use of attributes and discards irrelevant information in the problems. Together these provide a new representation and a learning algorithm for solving arithmetic word problems.

This paper is one step toward building a system that can solve any math and logic word problem. Our empirical evaluations show that our method outperforms a template-based learning method (developed recently by Kushman et al. (2014)) on solving addition and subtraction problems with diversity between the training and test sets. In particular, our method generalizes better to data from different domains because ARIS only relies on learning verb categories which alleviates the need for equation templates for arithmetic problems. In this paper, we have focused on addition and subtraction problems. However, KAZB can deal with more general types of problems such as multiplication, division, and simultaneous equations.

We have observed a complementary behavior between our method and that of Kushman et al. This suggests a hybrid approach that can benefit from the strengths of both methods while being applicable to more general problems while robust to the errors specific to each. In addition, we plan to focus on incrementally collecting domain knowledge to deal with missing information gaps. Another possible direction is to improve parsing and coreference resolution.

Acknowledgments

The research was supported by the Allen Institute for AI, and grants from the NSF (IIS-1352249) and UW-RRF (65-2775). We thank Ben Hixon and the anonymous reviewers for helpful comments and the feedback on the work.

References

Antoine Bordes, Nicolas Usunier, and Jason Weston. 2010. Label ranking under ambiguous supervision for learning semantic correspondences. In *Proc. International Conference on Machine Learning (ICML)*.

SRK Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proc. of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing of the AFNLP (ACL-AFNLP)*.

SRK Branavan, Nate Kushman, Tao Lei, and Regina Barzilay. 2012. Learning high-level planning from text. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proc. of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing of the AFNLP (ACL-AFNLP)*.

David Chen, Joohyun Kim, and Raymond Mooney. 2010. Training a multilingual sportscaster: Using perceptual context to learn language. *Journal of Artificial Intelligence Research*, 37.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proc. Language Resources and Evaluation Conference (LREC)*.

Jacob Eisenstein, James Clarke, Dan Goldwasser, and Dan Roth. 2009. Reading to learn: Constructing features from semantic abstracts. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Edward A. Feigenbaum and Julian Feldman, editors. 1963. *Computers and Thought*. McGraw Hill, New York.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ruifang Ge and Raymond J. Mooney. 2006. Discriminative reranking for semantic parsing. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Dan Goldwasser and Dan Roth. 2011. Learning from natural instructions. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

Dan Goldwasser, Roi Reichart, James Clarke, and Dan Roth. 2011. Confidence driven unsupervised semantic parsing. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Hannaneh Hajishirzi, Julia Hockenmaier, Erik T. Mueller, and Eyal Amir. 2011. Reasoning about robocup soccer narratives. In *Proc. Conference on Uncertainty in Artificial Intelligence (UAI)*.

Rohit J. Kate and Raymond J. Mooney. 2007. Learning language semantics from ambiguous supervision. In *Proc. Conference of the Association for the Advancement of Artificial Intelligence (AAAI)*.

Nate Kushman and Regina Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Proceeding of the Annual Meeting of the North American Chapter of the Association for Computational Linguistics*.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Iddo Lev, Bill MacCartney, Christopher D. Manning, , and Roger Levy. 2004. Solving logic puzzles: From robust processing to precise semantics. In *Workshop on Text Meaning and Interpretation at Association for Computational Linguistics (ACL)*.

Iddo Lev. 2007. *Packed Computation of Exact Meaning Representations*. Ph.D. thesis, CS, Stanford University.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proc. of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing of the AFNLP (ACL-AFNLP)*.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proc. International Conference on Machine Learning (ICML)*.

John McCarthy. 1980. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38.

Mee Young Park and Trevor Hastie. 2007. L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *International joint conference on Artificial intelligence (IJCAI)*.

Benjamin Snyder and Regina Barzilay. 2007. Database-text alignment via structured multilabel classification. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

Adam Vogel and Daniel Jurafsky. 2010. Learning to follow navigational directions. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ian H Witten, Eibe Frank, Leonard E Trigg, Mark A Hall, Geoffrey Holmes, and Sally Jo Cunningham. 1999. Weka: Practical machine learning tools and techniques with java implementations.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. Conference on Uncertainty in Artificial Intelligence (UAI)*.