Learning to Automatically Solve Logic Grid Puzzles

Arindam Mitra
SCIDSE
Arizona State University
amitra7@asu.edu

Chitta Baral
SCIDSE
Arizona State University
chitta@asu.edu

Abstract

Logic grid puzzle is a genre of logic puzzles in which we are given (in a natural language) a scenario, the object to be deduced and certain clues. The reader has to figure out the solution using the clues provided and some generic domain constraints. In this paper, we present a system, LOGICIA, that takes a logic grid puzzle and the set of elements in the puzzle and tries to solve it by translating it to the knowledge representation and reasoning language of Answer Set Programming (ASP) and then using an ASP solver. The translation to ASP involves extraction of entities and their relations from the clues. For that we use a novel learning based approach which uses varied supervision, including the entities present in a clue and the expected representation of a clue in ASP. Our system, LOGICIA, learns to automatically translate a clue with 81.11% accuracy and is able to solve 71% of the problems of a corpus. This is the first learning system that can solve logic grid puzzles described in natural language in a fully automated manner. The code and the data will be made publicly available at http://bioai.lab.asu.edu/logicgridpuzzles.

1 Introduction

Understanding natural language to solve problems be it algebraic word problems (Kushman et al., 2014; Hosseini et al., 2014) or questions from biology texts (Berant et al., 2014; Kim et al., 2011), has attracted a lot of research interest over the past few decades. For NLP, these problems are of particular interest as they are concise, yet rich in information. In this paper, we attempt to solve another problem of this kind, known as Logic Grid Puzzle. Problem.1 shows an example of the same. Puzzle problems in the same spirit as the previously mentioned science problems, do not restrict the vocabulary; they use everyday language and have diverse background stories. The puzzle problems, however, are unique in their requirement of high precision understanding of the text. For a puzzle problem, the solution is never in the text and requires involved reasoning. Moreover, one needs to correctly understand each of the given clues to successfully solve a problem. Another interesting property is that only a small core of the world knowledge, noticeably spatial, temporal and knowledge related to numbers, is crucial to solve these problems.

PROBLEM .1 A LOGIC GRID PUZZLE

> Waterford Spa had a full appointment calendar booked today. Help Janice figure out the schedule by matching each masseuse to her client, and determine the total price for each.

1. Hannah paid more than Teri's client.
2. Freda paid 20 dollars more than Lynda's client.
3. Hannah paid 10 dollars less than Nancy's client.
4. Nancy's client, Hannah and Ginger were all different clients.
5. Hannah was either the person who paid $180 or Lynda's client.

Clients: *Aimee, Ginger, Freda, Hannah*.
Prices: *$150, $160, $170, $180*.
Masseuses: *Lynda, Nancy, Tery, Whitney*.

A logic grid puzzle contains a set of categories and an equal number of elements in each category.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1035–1045,
Lisbon, Portugal, 17-21 September 2015.

And the goal is to find out which elements are linked together based on a series of given clues. Each element is used only once. Each puzzle has a unique solution and can be solved using logical reasoning. A logic grid puzzle is called a (n, m)-puzzle if it contains n categories and each category has m elements. For the example in Problem.1, there are three categories, namely *clients, prices, masseuses* and each category has four elements which are shown in the respective columns. A total of five clues are given in free text and the goal is to find the members of the four tuples, where each tuple shall contain exactly one element from each category such that all the members in a tuple are linked together.

To solve such a puzzle problem, it is crucial to understand the clues (for example, "Hannah paid more than Teri's client."). Each clue talks about a set of entities (for example, "Hannah", "client", "Terry") and their relations ("a greater-than relation between Hannah and the client of Terry on the basis of payment"). Our system, LOGICIA, learns to discover these entities and the underlying semantics of the relations that exist between them. Once the relations are discovered, a pair of Answer Set Programming (ASP) (Baral, 2003) rules are created. The reasoning module takes these ASP rules as input and finds a group configuration that satisfies all the clues. LOGICIA has "knowledge" about a fixed set of predicates which models different relations that hold between entities in a puzzle world. Clues in the puzzle text that are converted into ASP rules, use these predicates as building blocks. In this research, our goal is to build a system which can automatically do this conversion and then reason over it to find the solution. The set of predicates that the reasoning model is aware of is not sufficient to represent all logic grid puzzles. The family of logic grid puzzles is broad and contains variety of clues. Our future work involves dealing with such a diverse set of relations. In this work we assume that the relations in Table 1 are sufficient to represent the clues. Following are some examples of clues that cannot be modeled using the predicates in Table 1.

- *Esther's brother's* seat is at one end of the block of seven.
- The *writer* of *Lifetime Ambition* has a *first name* with more letters than that of the *tennis star*.
- *Edward* was *two places* behind *Salim* in one of the lines, *both* being in odd-numbered positions.
- *Performers* who finished in the top *three* places, in no particular order, are *Tanya* , the *person* who performed the *fox trot*, and the *one* who performed the *waltz*.

The rest of the paper is organized as follows: in section 2, we describe the representation of a puzzle problem in ASP and delineate how it helps in reasoning; in section 3, we present our novel method for learning to automatically translate a logic problem described in natural language to its ASP counterpart. In section 4, we describe the related works. In section 5, we discuss the detailed experimental evaluation of our system. Finally, section 6 concludes our paper.

2 Puzzle Representation

Answer Set Programming (ASP) (Baral, 2003; Lifschitz, 1999; Gelfond and Lifschitz, 1991) has been used to represent a puzzle and reason over it. This choice is facilitated by the two important reasons: 1) non-monotonic reasoning may occur in a puzzle (Nagy and Allwein, 2004) and 2) ASP constructs greatly simplify the reasoning module, as we will see in this section. We now briefly describe a part of ASP. Our discussion is informal. For a detailed account of the language, readers are referred to (Baral, 2003).

2.1 Answer Set Programming

An answer set program is a collection of rules of the form,

$$L_0 \mid ... \mid L_k \text{ :- } L_{k+1}, ..., L_m, \textbf{not } L_{m+1}, ..., \textbf{not } L_n$$

where each of the L_i's is a literal in the sense of a classical logic. Intuitively, the above rule means that if $L_{k+1}, ..., L_m$ are to be true and if $L_{m+1}, ..., L_n$ can be safely assumed to be false then at least one of $L_0, ..., L_k$ must be true. The left-hand side of an ASP rule is called the *head* and the right-hand side is called the *body*. A rule with no *head* is often referred to as a $constraint$. A rule with empty *body* is referred to as a $fact$ and written as,

$$L_0 \mid L_1 \mid ... \mid L_k.$$

Example

fly(X) :- bird(X), not ab(X).

The above program represents the knowledge that "Most birds fly". If we add the following rule ($fact$) to the program,

bird(penguin).

the answer set of the program will contain the belief that penguins can fly, $\{bird(penguin), fly(penguin)\}$. However, adding one more fact, '$ab(penguin)$.', to convey that the penguin is an abnormal bird, will change the belief that the penguin can fly and correspondingly the answer set, $\{bird(penguin), ab(penguin)\}$, will not contain the fact, $fly(penguin)$.

Choice Rule

$$m\ \{p(X) : q(X)\}\ n : -L_1, ..., L_k, ..., \textbf{not}\ L_n.$$

Rules of this type allow inclusion in the program's answer sets of arbitrary collections S of atoms of the form $p(t)$ such that, $m \leq \mid S \mid \leq n$ and if $p(t) \in S$ then $q(t)$ belongs to the corresponding answer set.

2.2 Representing Puzzle Entities

A (m, n)-puzzle problem contains m categories and n elements in each category. The term 'puzzle entity' is used to refer to any of them. Each category is assigned an unique index, denoted by the predicate $cindex/1$ (the number after the '/' denotes the arity of the predicate). The predicate $etype/2$ captures this association. Each element is represented, by the $element/2$ predicate which connects a category index to its element. The predicate $eindex/1$, denotes the tuple indices. The following blocks of code shows the representation of the entities for the puzzle in Problem.1.

```
cindex(1...3).
eindex(1...4).

etype(1,clients).
etype(2,prices).
etype(3,masseuses).

element(1,aimee;;1,ginger).
element(1,freda;;1,hannah).
element(2,150;;2,160).
element(2,170;;2,180).
element(3,lynda;;3,nancy).
element(3,teri;;3,whitney).
```

2.3 Representing Solution

Solution to a logic grid puzzle is a set of tuples containing related elements. The $tuple/3$ predicate captures this tuple membership information of the elements. For example, the fact, $tuple(2, 1, aimee)$, states that the element $aimee$ from the category with index 1 is in the tuple 2. The rel/m predicate captures all the elements in a tuple for a (m, n)-puzzle and is defined using the $tuple/3$ predicate.

2.4 Domain Constraints

In the proposed approach, the logic grid puzzle problem is solved as a constraint satisfaction problem. Given a puzzle problem the goal is to enumerate over all possible configurations of $tuple/3$, and select the one which does not violate the constraints specified in the clues. However, 1) each tuple in a logic grid puzzle will contain exactly one element from each category and 2) an element will belong to exactly one tuple. These constraints come from the specification of a puzzle problem and will hold irrespective of the problem instance. Following blocks of code show an elegant representation of these domain constraints in ASP along with the enumeration.

```
%enumerate over the tuple
%assignments with constraint#1
1 {
    tuple(G,Cat,Elem):
    element(Cat,Elem)
} 1 :- cindex(Cat),
       eindex(G).

%domain constraint#2
:-tuple(G1,Cat,Elem),
  tuple(G2,Cat,Elem),
  G1!=G2.
```

2.5 Representing clues

Each clue describes some entities and the relations that hold between them. In its simplest form, the relations will suggest if the entities are linked together or not. However, the underlying semantics of such relations can be deep such as the one in clue 5 of Problem.1. There are different ways to express the same relation that holds between entities. For example, in Problem.1, the *possessive* relation has been used to express the linking between clients and masseuses; and the word *paid*

expresses the linking between the clients and the prices. Depending on the puzzles the phrases that are used to express the relations will vary and it is crucial to identify their underlying semantics to solve the problems in systematic way.

In the current version, the reasoning module has knowledge of a selected set of relations and the translation module tries to represent the clue as a conjunction of these relations. All these relations and their underlying meanings are described in table 1. In this subsection, we describe the representation of a clue in terms of these relations in ASP and show how it is used by the reasoning module. In the next section, we present our approach to automate this translation.

Let us consider the clues and their representation from Problem.1:

[1] Hannah paid more than Teri's client.

```
clue1 :-
  greaterThan(hannah,1,X,1,2),
  sameTuple(X,1,teri,3).
:- not clue1.
```

The first rule $clue1$ evaluates to true (will be in the answer set) if the element from category 1 with value $hannah$ is linked to some element from category 2 which has a higher value than the element from its own category which is linked to an element from category 1 which is linked to $teri$ from category 3. Since the desired solution must satisfy the relations described in the clue, the second ASP rule is added. A rule of this form that does not have a $head$ is known as a $constraint$ and the program must satisfy it to have an answer set. As the reasoning module enumerates over all possible configurations, in some cases the $clue1$ will not hold and subsequently those branches will be pruned. Similar constraints will be added for all clues. In the below, we show some more examples. A configuration which satisfies all the clue constraints and the domain constraints described in the previous section, will be accepted as the solution to the puzzle.

[2] Nancy's client, Hannah and Ginger were all different clients.

```
clue4 :-
  diffTuple(hannah,1,ginger,1),
  diffTuple(hannah,1,X,1),
  diffTuple(X,1,ginger,1),
  sameTuple(X,1,nancy,3).
:- not clue4.
```

[3] Hannah was either the person who paid $180 or Lynda's client.

```
clue5 :-
   eitherOr(hannah,1,X,1,Y,1),
   sameTuple(X,1,180,2).
   sameTuple(Y,1,lynda,3).
:- clue5.
```

3 Learning Translation

To automate the translation of a clue to the pair of ASP rules, the translation module needs to identify the entities that are present in the clue, their category and their value; and the underlying interpretations of all the relations that hold between them. Once all the relation instances $\{R_1(arg_1, ..., arg_{p_1})$,..., $R_q(arg_1, ..., arg_{p_q})\}$, in the clue are identified, the ASP representation of the clue is generated in the following way:

$$clue : -R_1(arg_1..., arg_{p_1}), ..., R_q(arg_1..., arg_{p_q})$$

The entity classification problem for logic grid puzzles poses several challenges. First, the existence of a wide variety in the set of entities. Entities can be names of objects, time related to some event, numbers, dates, currency, some form of ID etc. And it is not necessary that the entities in puzzles are nouns. It can be verbs, adjectives etc. Second and of paramount important, the "category" of a puzzle "element" is specific to a puzzle problem. Same element may have different category in different problems. Also, a constituent in a clue which refers to an entity in a particular problem may not refer to an entity in another problem. We formalize this problem in this section and propose one approach to solve the problem. Next, we discuss the method that is used to extract relations from clues. To the best of our knowledge, this type of entity classification problem has never been studied before.

Relation	Interpretation
sameTuple(E1, C1, E2, C2)	States that two elements, (C1,E1) and (C2,E2) are in the same tuple. The dictionary also contains the negation of it, *diffTuple*(E1, C1, E2, C2).
referrent(E1, C1, E2, C2)	States that the elements are identical.
posDiff(E1, C1, E2, C2, N1, NC1)	If (C1,E1) is related to (NC1,X1) and (E2,C2) is related to (NC1,X2), then difference(X1,X2)=N1. Similarly the dictionary contains the predicate *negDiff*.
greaterThan(E1, C1, E2, C2, NC1)	Similar to *posDiff* however the difference(X1,X2) $>$ 0. The dictionary also contains its opposite predicate *lessThan*.
members(E1, C1, E2, C2,..., EN, CN)	All the elements are distinct and do not share a tuple.
eitherOr(E1, C1, E2, C2,..., EN, CN)	The first element is related to one of the last $N-1$ elements. The last $N-1$ elements are assumed to be different unless contradicts with other beliefs.
referrent22(E1, C1, E2, C2, E3, C3, E4, C4)	The first two elements are different and referring to the last two elements.

Table 1: Describes the various relations that are part of the reasoning module.

3.1 Entity Classification

The entity classification problem is defined as follows:

Problem description Given m categories C_1, ..., C_m and a text T, each category C_i, $1 \leq i \leq m$, contains a collection of elements E_i and an optional textual description d_i. The goal is to find the class information of all the constituents in the text T. Each category contributes two classes, where one of them represents the category itself and the other represents an instance of that category. Also, a constituent may not refer to any category or any instance of it, in that case the class of that constituent is $null$. So, there are a total $2m+1$ classes and a constituent will take one value from them.

Example In the puzzle of Problem.1, there are 3 categories with, C_1 = {Aimee, Freda, Ginger, Hannah}, C_2 = {\$150, \$160, \$170, \$180}, C_3 = {Lynda, Nancy, Terry, Whiteney} and d_1 = "clients", d_2 = "prices", d_3 = "masseuses". The text T, is the concatenation of all clues. In the last clue, there are a total 5 entities, namely "Hannah", "person", "\$180", "Lydia","client" and the corresponding classes are "Instance of C_1", "Instance of C_1", "Instance of C_2", "Instance of C_3" and "Instance of C_1" respectively. The remaining constituents in that clue have the class value $null$. The constituent "clients" in the fourth clue refers to the category C_1.

Our approach We model the Entity Classification problem as a decoding query on Pairwise Markov Network (Koller and Friedman, 2009; Kindermann et al., 1980; Zhang et al., 2001). A pairwise Markov network over a graph $\mathcal{H}$, is associated with a set of node potentials $\{\phi(X_i) : i = 1, ..., n\}$ and a set of edge potentials $\{\phi(X_i, X_j) : (X_i, X_j) \in \mathcal{H}\}$. Each node $X_i \in \mathcal{H}$, represents a random variable. Here, each X_i can take value from the set $\{1...2m+1\}$, denoting the class of the corresponding constituent in the text T.

In our implementation, the node potential captures the chances of that node to be classified as one of the possible categories without being affected by the given text T. And the edge potentials captures hints from the context in T for classification. After constructing the pairwise Markov network, a decoding query is issued to obtain the configuration that maximizes the joint probability distribution of the pairwise Markov network in consideration. The proposed approach is inspired by the following two observations: 1) to find the class of a constituent one needs some background knowledge; 2) however, background knowledge is not sufficient on its own, one also needs to understand the text to properly identify the class of each constituent. For example, let us consider the word "person" in clue 5 of Problem.1. Just skimming through the categories, one can discover that the word "person" is very unlikely to be a instance of the category "prices", which is from her knowledge about those constituents. However a proper

disambiguation may face an issue here as there are two different categories of human beings. To properly classify the word "person" it is necessary to go through the text.

The following paragraphs describe the construction of the grah $\mathcal{H}$, and the algorithm that is used in the computation of associated set of node potentials and edge potentials.

Construction of the graph While constructing the graph, we assign a label, L, to each edge in $\mathcal{H}$ which will be used in the edge potential computation. Let $\mathcal{D}_\mathcal{G}$ denotes the dependency graph of the text T obtained from the Stanford dependency parser (Chen and Manning, 2014) and $dep(v_1, v_2)$ denotes the grammatical relation between $(v_1, v_2) \in \mathcal{D}_\mathcal{G}$. Then the graph, $\mathcal{H}$, is constructed as follows:

1. Create a node in $\mathcal{H}$ for each constituent w_j in T if $w_j \in \mathcal{D}_\mathcal{G}$.
2. Add an edge (X_i, X_j) to $\mathcal{H}$ if the corresponding edge $(w_p, w_q) \in \mathcal{D}_\mathcal{G}$. $L(X_i, X_j) := dep(w_p, w_q)$.
3. Add an edge between a pair of nodes (X_i, X_j) if the corresponding words are synonyms. $L(X_i, X_j) := synonymy$.
4. Create a node for each element and category specified in the puzzle and add an edge from them to others if the corresponding string descriptions are 'same'. In this case, the edges are labeled as $exact\_match$.
5. If $(X_i, X_j) \in \mathcal{H}$ and $L(X_i, X_j) = exact\_match$ and both of them are referring to a verb, then add more edges (X_i', X_j') to $\mathcal{H}$ with label $spatial\_symmetry$, where $L(X_i, X_i') = L(X_j, X_j')$.

Determining Node potentials For each element in the m category, a set of naive regular-expression based taggers are used to detect it's type (For example, "am-pm time"). Each element type maps to a WordNet (Miller, 1995) representative (For example, "time_unit#n"). For each constituent w a similarity score, $sim(w,c)$, is calculated to each class $c \in \{1...2m+1\}$, in the following way:

•**Class c is denoting instance of some category C_i** Similarity scores are computed between the textual description of the constituent to both the WordNet representative of E_i and the textual description d_i using the HSO WordNet similarity algorithm (Hirst and St-Onge, 1998). The similarity score, $sim(w,c)$, is chosen to be the maximum of them.

•**Class c is denoting a category C_i :** $sim(w,c)$ is assigned the value of HSO Similarity between the textual description and d_i.

•**Class c is $null$:** In this case similarity is calculated using the following formula:

$$sim(w, null) = MAX_{HSO} - \max_{c \neq null} sim(w, c)$$

where MAX_{HSO} denotes the maximum similarity score returned by HSO algorithm, which is 16.

Node potential for each node $X_i \in \mathcal{H}$, corresponding to the constituent w_j, are then calculated by,

$$\phi(X_i = c) = 1 + sim(w_j, c), \forall c$$

Determining Edge potentials For each edge in the graph H, the edge potential, $\phi(X_i, X_j)$ is calculated using the following formula,

$$\phi(X_i = c_1, X_j = c_2) =$$
$$1 + \begin{cases} P(X_i = X_j | L(X_i, X_j)), & \text{if } c_1 = c_2 \\ P(X_i \neq X_j | L(X_i, X_j)), & \text{otherwise} \end{cases}$$

In the training phase, each entity in a clue is tagged with its respective class. The probability values are then calculated from the training dataset using simple count.

3.2 Learning To Extract Relations

The goal here is to identify all the relations $R(arg_1, ..., arg_p)$ that are present in a clue, where each relation belongs to the logical vocabulary described in Table 1 . This problem is known as *Complex relation extraction* (McDonald et al., 2005; Bach and Badaskar, 2007; Fundel et al., 2007; Zhou et al., 2014). The common approach for solving the *Complex relation extraction* problem is to first find the relation between each pair of entities and then discover the complex relations from binary ones using the definition of each relation.

Figure 1 depicts the scenario. The goal is to identify the relation $possDiff(E1, E2, E3)$, where $E1, E2, E3$ are constituents having a non-$null$ class value. However instead of identifying $posDiff(E1, E2, E3)$ directly, first the relation

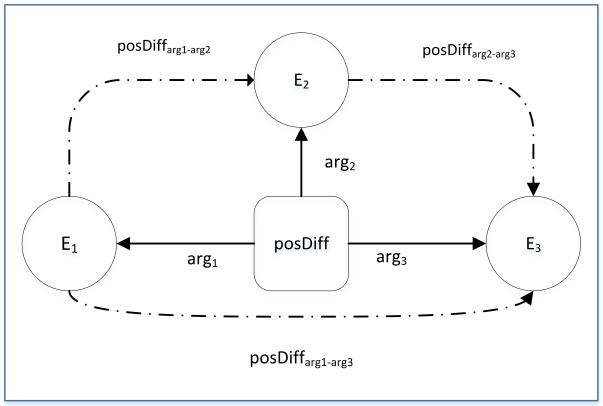

Figure 1: Binary representation of the relation $possDiff$

between each pair of entities will be identified. If the relations $\{posDiff_{arg_1-arg_2}(E_1, E_2)$, $posDiff_{arg_2-arg_3}(E_2, E_3)$, $posDiff_{arg_1-arg_3}(E_1,E_3)$ $\}$ are identified, the extraction module will infer that $posDiff(E1, E2, E3)$ holds. In a similar manner, a set of total 39 binary relations are created for all the relations described in Table 1.

In the training phase, all the relations and their respective arguments in each clue are given. Using this supervision, we have built a Maximum Entropy based model (Berger et al., 1996; Della Pietra et al., 1997) to classify the relation between a pair of entities present in a clue. Maximum entropy classifier has been successfully applied in many natural language processing applications (Charniak, 2000; Chieu and Ng, 2002; Ratnaparkhi and others, 1996) and allows the inclusion of various sources of information without necessarily assuming any independence between the features. In this model, the conditional probability distribution is given by:

$$P(c|d) = \frac{\prod_{j=1...K} e^{\lambda_i f_i(d,c)}}{\sum_{c' \in C} \prod_{j=1...K} e^{\lambda_i f_i(d,c')}} \quad (1)$$

where the denominator is the normalization term and the parameter λ_i correspond to the weight for the feature f_i. Features in Maximum Entropy model are functions from context and classes to the set of real numbers. A detailed description of the model or parameter estimation method used - Generalized Iterative Scaling, can be found at (Darroch and Ratcliff, 1972).

Table 2 describes the features that are used in the classification task. Here, $path(E_1, E_2)$ denotes all the words that occur in the path(s) connecting E_1 and $E2$ in the dependency graph of the clue.

Feature Set
Class of E_1 and E_2
All the grammatical relations between the words in $path(E_1, E_2)$
All the adjectives and adverbs in $path(E_1, E_2)$.
POS tags of all the words in $path(E_1, E_2)$
TypeMatched = [[class of E_1 = class of E_2]]
IsE_1Numeric = [[class of E_1 is Numeric]]
IsE_2Numeric = [[class of E_2 is Numeric]]
All the words that appears in the following grammatical relations $advmod$, $amod$, cop, det with the words in $path(E_1, E_2)$.
hasNegativeWord = [[$\exists$w $\in$ $path(E_1, E_2)$ s.t. w has a neg relation starting with it.]]

Table 2: Features used in the classification task

The relation between each pair of entities in a clue is the one which maximizes the conditional probability in equation (1).

3.2.1 Missing Entity

In the case of comparative relations in Table 1, such as $greaterThan$, the basis of the comparison can be hidden. For example, in clue 1 of the example problem, the two entities, "Hannah" and "client" have been compared on the basis of "price", however there is no constituent in the clue which refers to an element from that category. The basis of comparison is hidden in this case and is implied by the word "paid". In the current implementation, the translation module does not handle this case. For puzzles that contain only one category consisting of numeric elements, the translation module goes with the obvious choice. This is part of our future work.

4 Related Work

There has been a significant amount of work on the representation of puzzle problems in a formal language (Gelfond and Kahl, 2014; Baral, 2003; Celik et al., 2009). However, there has not been any work that can automatically solve a logic grid puzzle. The latest work (Baral and Dzifcak, 2012) on this problem, assumes that the entities in a clue are given and the authors manually simplify the sentences for translation. Furthermore their representation of logic grid puzzles does not consider the category of a variable in the formal representation

i.e. uses $element/1$ and $tuple/2$ predicates and thus cannot solve puzzles containing more than one numeric categories.

In the same work (Baral and Dzifcak, 2012), the authors propose to use a semantic parser to do the translation. This method works well for simple sentences such as "Donna dale does not have green fleece" however it faces several challenges while dealing with real world puzzle sentences. The difficulty arises due to the restrictions enforced in the translation models used by the existing semantic parsers. Traditional semantic parsers (Vo et al., 2015; Zettlemoyer and Collins, 2005) assign meanings to each word in a dictionary and combine the meaning of the words to characterize the complete sentence. A phrase structure grammar formalism such as Combinatory Categorial Grammar (Steedman and Baldridge, 2011; Vo et al., 2015; Zettlemoyer and Collins, 2005), Context Free Grammar (Aho and Ullman, 1972; Wong and Mooney, 2006), is normally used to obtain the way words combine with each other. In the training phase, the semantic parser learns the meanings of words given a corpus of <sentence, meaning> pairs and stores them in a dictionary. During translation, the semantic parser uses those learned meanings to obtain the meaning of the sentence. Firstly, for the puzzle problems the meaning of the words changes drastically depending on the puzzle. A word may be an entity in one puzzle, but, in a different problem it might not be an entity or might belong to a different category altogether. Thus a learned dictionary may not be useful while translating clues in a new puzzle. Secondly, in puzzles relations are normally expressed by phrases. For example, in the clue "The person who played at Eden Gardens played for India", the phrases "played at" and "played for" are used to express two different relations. Thus, using a model that assigns meaning to each word may not be suitable here. Finally, it is difficult to identify the participants of a relation with a parse tree generated following a phrase structure grammar. For example, consider the parse tree of the clue "The person who trekked for 8 miles started at Bull Creek". Even though, the relation "started at" takes the word 'person' and 'Bull Creek' as its input, it receives the entire phrase "the person who trekked for 8 miles" as its argument along with the other input 'Bull Creek'.

The entity classification problem studied in this research shares many similarity with Named Entity Recognition (Nadeau and Sekine, 2007; Zhou and Su, 2002) and the Word Sense disambiguation (Stevenson and Wilks, 2003; Sanderson, 1994) task. However, our work has a major difference; in the entity classification problem, the class of an entity varies with the problem and does not belong to a known closed set, whereas for the other two problems the possible classes are pre-specified.

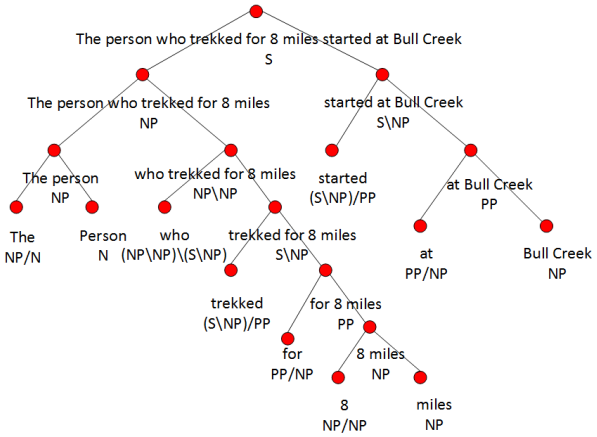

Figure 2: Parse tree of an example sentence in Combinatory categorial grammar

5 Experimental Evaluation

Dataset To evaluate our method we have built a dataset of logic grid puzzles along with their correct solutions. A total of 150 problems are collected from `logic-puzzles.org`. Out of them 100 problems are fully annotated with the entities and the relations information. The remaining 50 puzzles do not have any annotation except their solution. The set of annotated puzzles contain a total of 467 clues, 5687 words, 1681 entities and 862 relations. The set of 50 puzzles contain a total of 222 clues with 2604 words.

Tasks We evaluate LOGICIA on three tasks: 1) puzzle solving; 2) entity classification; and 3) relation extraction. We use the percentage of correct answers as the evaluation metric for all the three tasks. In case of a logic grid puzzle solving, an answer is considered correct if it exactly matches the solution of that puzzle.

Training-Testing Out of the 100 annotated puzzle problems 50 are used as training samples and remaining 50 puzzles are used in testing. The set of 50 unannotated puzzles are used solely for the task of testing puzzle solving.

	Entity classification	Binary relation classification with annotation		Relation extraction with annotation		Solution
		Yes	No	Yes	No	
Total	1766	960		450		50
Correct	1502	922	854	410	365	37
Percentage	85.05%	96.04%	88.95%	90.90%	81.11%	74%

Table 3: Accuracy on 50 annotated puzzle problems in the Test set.

Results Table 3 & 4 shows the efficacy of our approach in solving logic grid puzzles with the selected set of relations. LOGICIA is able to classify the constituents with 85.05% accuracy and is able to solve 71 problems out of the 100 test puzzles. It should be noted that puzzle problems requires precise understanding of the text and to obtain the correct solution of a puzzle problem all the entities and their relations in the puzzle need to be identified. Columns 2 and 3 in Table 3 compares the performance on relation extraction when it is used in conjunction with the entity classification and when it directly uses the annotated entity.

Puzzle Solving		
Total	Correct	Percentage
50	34	68%

Table 4: Accuracy on unannotated 50 test puzzle problems.

Error Analysis The errors in entity classification falls into two major categories. In the first category, more knowledge of similarity is needed than what is currently obtained from the WordNet. Consider for example, the categories are "class number" and "class size" and the constituent is "20 students". Even though the constituent is closer to "class size", standard WordNet based similarity methods are unable to provide such information. In the second category, the WordNet similarity of the constituent to one of the classes is quite high due to their position in the WordNet hierarchy; however with respect to the particular problem the constituent is not an entity. The relation extraction task performs fairly well, however the binary relation classification task does not jointly consider the relation between all the entities and because of that if one of the necessary binary relation of a complex relation is misclassified, the extraction of the entire relation gets affected.

6 Conclusion & Future Work

This paper presents a novel approach for solving logic grid puzzle. To the best of our knowledge, this is a novel work with respect to the fact that that it can automatically solve a given logic grid puzzle.

There are several advantages of our approach. The inclusion of knowledge in terms of a vocabulary of relations makes it scalable. For puzzles which make use of a different set of constraints, such as "Lynda *sat on an even numbered position*", can be easily integrated into the vocabulary and the system can then be trained to identify those relations for new puzzles. Also, the proposed approach separates the representation from reasoning. The translation module only identifies the relation and their arguments; it is not aware of the meaning of those relations. The reasoning module, on the other hand, knows the definition of each relation and subsequently prunes those possibilities when relations appearing in a clue does not hold. This separation of representation from reasoning allows the system to deal with the complex relations that appear in a clue.

There are a few practical and theoretical issues which need to be addressed. One of those is updating the logical vocabulary in a scalable manner. Logic grid puzzle is a wide family of puzzles and it will require more knowledge of relations than what is currently available. Another challenge that needs to be addressed is the computation of similarity between complex concepts such as "size of class" and "20 students". Also, the case of "missing entity" (3.2) needs to be modeled properly. This work is the first step towards further understanding these important issues.

Acknowledgements

We thank NSF for the DataNet Federation Consortium grant OCI-0940841 and ONR for their grant N00014-13-1-0334 for partially supporting this research.

References

Alfred V Aho and Jeffrey D Ullman. 1972. *The theory of parsing, translation, and compiling*. Prentice-Hall, Inc.

Nguyen Bach and Sameer Badaskar. 2007. A review of relation extraction. *Literature review for Language and Statistics II*.

Chitta Baral and Juraj Dzifcak. 2012. Solving puzzles described in english by automated translation to answer set programming and learning how to do that translation. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*.

Chitta Baral. 2003. *Knowledge representation, reasoning and declarative problem solving*. Cambridge university press.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Brad Huang, Christopher D Manning, Abby Vander Linden, Brittany Harding, and Peter Clark. 2014. Modeling biological processes for reading comprehension. In *Proc. EMNLP*.

Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71.

Mehmet Celik, Halit Erdogan, Fırat Tahaoglu, Tansel Uras, and Esra Erdem. 2009. Comparing asp and cp on four grid puzzles. In *Proceedings of the Sixteenth RCRA International Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion (RCRA09). CEUR Workshop Proceedings*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139. Association for Computational Linguistics.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750.

Hai Leong Chieu and Hwee Tou Ng. 2002. Named entity recognition: a maximum entropy approach using global information. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

John N Darroch and Douglas Ratcliff. 1972. Generalized iterative scaling for log-linear models. *The annals of mathematical statistics*, pages 1470–1480.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. 1997. Inducing features of random fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(4):380–393.

Katrin Fundel, Robert Küffner, and Ralf Zimmer. 2007. Relexrelation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371.

Michael Gelfond and Yulia Kahl. 2014. *Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach*. Cambridge University Press.

Michael Gelfond and Vladimir Lifschitz. 1991. Classical negation in logic programs and disjunctive databases. *New generation computing*, 9(3-4):365–385.

Graeme Hirst and David St-Onge. 1998. Lexical chains as representations of context for the detection and correction of malapropisms. *WordNet: An electronic lexical database*, 305:305–332.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jin-Dong Kim, Sampo Pyysalo, Tomoko Ohta, Robert Bossy, Ngan Nguyen, and Jun'ichi Tsujii. 2011. Overview of bionlp shared task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 1–6. Association for Computational Linguistics.

Ross Kindermann, James Laurie Snell, et al. 1980. *Markov random fields and their applications*, volume 1. American Mathematical Society Providence, RI.

Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. *ACL (1)*, pages 271–281.

Vladimir Lifschitz. 1999. Answer set planning. In *Logic Programming and Nonmonotonic Reasoning*, pages 373–374. Springer.

Ryan McDonald, Fernando Pereira, Seth Kulick, Scott Winters, Yang Jin, and Pete White. 2005. Simple algorithms for complex relation extraction with applications to biomedical ie. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 491–498. Association for Computational Linguistics.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Benedek Nagy and Gerard Allwein. 2004. Diagrams and non-monotonicity in puzzles. In *Diagrammatic Representation and Inference*, pages 82–96. Springer.

Adwait Ratnaparkhi et al. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the conference on empirical methods in natural language processing*, volume 1, pages 133–142. Philadelphia, PA.

Mark Sanderson. 1994. Word sense disambiguation and information retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 142–151. Springer-Verlag New York, Inc.

Mark Steedman and Jason Baldridge. 2011. Combinatory categorial grammar. *Non-Transformational Syntax: Formal and Explicit Models of Grammar. Wiley-Blackwell*.

Mark Stevenson and Yorick Wilks. 2003. Word sense disambiguation. *The Oxford Handbook of Comp. Linguistics*, pages 249–265.

Nguyen Ha Vo, Arindam Mitra, and Chitta Baral. 2015. The NL2KR platform for building natural language translation systems. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 899–908.

Yuk Wah Wong and Raymond J Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 439–446. Association for Computational Linguistics.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. In *UAI*, pages 658–666. AUAI Press.

Yongyue Zhang, Michael Brady, and Stephen Smith. 2001. Segmentation of brain mr images through a hidden markov random field model and the expectation-maximization algorithm. *Medical Imaging, IEEE Transactions on*, 20(1):45–57.

GuoDong Zhou and Jian Su. 2002. Named entity recognition using an hmm-based chunk tagger. In *proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 473–480. Association for Computational Linguistics.

Deyu Zhou, Dayou Zhong, and Yulan He. 2014. Biomedical relation extraction: From binary to complex. *Computational and mathematical methods in medicine*, 2014.