Computing Meaning: an Application of Natural Language Processing in Grid Puzzles

Introduction

Even though understanding the meaning of most words is effortless to the native speakers of a language, quantifying this meaning is surprisingly difficult. This meaning of language is investigated in many areas of Linguistics but is most central to Semantics, which studies how complex meaning can emerge by combining the meaning of simpler expressions. In addition to semantics, the syntactic function of words can also offer insight into their meaning (McKoon and Ratcliff, 2003). Accordingly, the current project investigates meaning by considering both semantics and syntax. Gaining a better understanding of the meaning of a language would not only benefit linguistics, but also technology, as natural language processing applications become increasingly more prevalent.

Inferring semantic information is a notoriously difficult task in natural language processing. In particular, tasks such as performing inference in web search engines and information retrieval warrant an automated construction of meaning representation. Providing these representations is one of the goals of computational semantics. These representations are commonly based on logic, because similarly to language, logic statements can be easily combined to form more complex representations. The most common types of logic that are used in semantic representations are Propositional Logic and Predicate Logic. In the current investigation, we pursue a third choice - Montague semantics.

Propositional Logic

Propositional logic consists of atomic propositions and basic logic operators, such as negation, conjunction, and disjunction, that can be used to generating more complicated
expressions. While propositional logic can account for some of the generative nature of natural language, it fails to represent common meaning between expressions. This problem occurs because every new expression is represented with different atomic proposition constants. Hoping to generate a model that accounts for more variability in representation, we turn to predicate logic.

**Predicate Logic**

Similarly to propositional logic, predicate formulas consist of basic units and logic rules that can combine them. However, predicate logic has more structured propositions, or "predicates", that can be used to relate up to $n$ variables. In addition, predicate logic can represent quantification through its universal ($\forall$) and existential ($\exists$) quantifiers. These additional features of predicate logic allow for a more natural expression of language meaning. However, we would like to take full advantage of syntactical information, which can only be done by using a higher-order logic.

**Montague Semantics**

The type of higher-order logic used in the current project was proposed by Richard Montague in the 1970s. In addition to providing a way to systematically relate syntax to semantics, Montague's approach is especially useful because it is based on lambda calculus, which has an analog in functional programming. For instance, the sentence "John sings." can be expressed as follows using Montague semantics:

$$\lambda P [P(\text{John})](\text{sing})$$

(1)

Here, the verb "sing" serves as a predicate and this formulation can also be expressed in a functional programming language, such as Haskell, by applying the formulation of the noun phrase (NP) to that of the verb phrase (VP), as follows:
In order to use Montague semantics, we need to first parse the language bits into syntactic categories. The parser used in the current project is the Stanford probabilistic context-free grammar (PCFG) parser (Klein and Manning, 2003).

**Syntactic parsing**

The parser used in this project was trained on hand-parsed sentences and produces the most likely parses of sentences according to its training. While the Stanford PCFG parser outperforms many other models, it is not without flaws. Particularly, two problematic areas that are common across parsers are imperative sentences and binding of prepositional phrases (PP). Unfortunately, both occur frequently in the domain of grid puzzles, which is the subject of the current study. The following example illustrates both an inaccurate imperative parse and an inaccurate binding of a sentence from an instruction to a grid puzzle known as KenKen.

We observe a few semantic errors in this parse, such as neglecting to recognize that “of a cage” is a whole prepositional phrase and that it relates to “each square” instead of “with a number”. While the Stanford parse may exhibit some inaccurate results, it is generally at least more reliable than other parsers and was thus selected for use in this project.
Grid puzzle domain

In this project, we attempt to gain a better understanding of language meaning by restricting our domain to instructions of grid puzzles in English. The puzzle framework will enable us to focus on key words in the puzzle instructions and attempt to extract their meanings based on the assumed context. By focusing on a small subset of English words, this constraint removes a large portion of the ambiguity encountered in natural language. For instance, distinguishing between certain homonyms, such as the verb and noun corresponding to "row", would not be necessary. In addition, the remaining morphological ambiguities, such as the verb
and noun corresponding to "place", are differentiated by using the syntactic information from the parse.

The overall goal of this investigation and its future extensions is to extract sufficient meaning from the grid puzzle instructions to specify sufficient constraints that can be used to solve the grid puzzle. In the current study, we provided semantic representations for some words that commonly occur in grid puzzles.

**Current approach**

We first identified specific grid puzzles and instructions. Puzzles must be solvable and be presented on a finite grid. Some of the puzzles from which we sampled instructions are shown in Figure 2.

![Figure 2. Grid puzzle examples from left to right: KenKen, Sudoku, and Nonograms.](image)

All instructions were written in paragraph form in English and pertained to a grid puzzle. An example instruction for a KenKen puzzle reads:

*Fill in each square of a cage with a number. The numbers in a cage must combine—in any order, using only that cage’s math operation—to form that cage’s target number. You*
may not repeat a number in any row or column. You can repeat a number within a cage, as long as those repeated numbers are not in the same row or column.

Next, we calculated the frequencies of occurrence of words in these instructions to gain a better understanding of the word domain. The 40 words that occurred most frequently can be seen in Table 1.

<table>
<thead>
<tr>
<th>the</th>
<th>29</th>
<th>grid</th>
<th>7</th>
<th>are</th>
<th>5</th>
<th>jigsaw</th>
<th>4</th>
<th>on</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>of</td>
<td>15</td>
<td>7</td>
<td>four</td>
<td>5</td>
<td>no</td>
<td>4</td>
<td>this</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>in</td>
<td>13</td>
<td>to</td>
<td>7</td>
<td>is</td>
<td>5</td>
<td>all</td>
<td>3</td>
<td>will</td>
<td>3</td>
</tr>
<tr>
<td>a</td>
<td>9</td>
<td>chips</td>
<td>6</td>
<td>so</td>
<td>5</td>
<td>box</td>
<td>3</td>
<td>diagonal(s/ly)</td>
<td>3</td>
</tr>
<tr>
<td>that</td>
<td>9</td>
<td>column</td>
<td>6</td>
<td>digit(s)</td>
<td>5</td>
<td>either</td>
<td>3</td>
<td>puzzle(s)</td>
<td>3</td>
</tr>
<tr>
<td>square(s)</td>
<td>9</td>
<td>each</td>
<td>6</td>
<td>black</td>
<td>4</td>
<td>given</td>
<td>3</td>
<td>two</td>
<td>2</td>
</tr>
<tr>
<td>and</td>
<td>8</td>
<td>row</td>
<td>6</td>
<td>clues</td>
<td>4</td>
<td>have</td>
<td>3</td>
<td>same</td>
<td>2</td>
</tr>
<tr>
<td>number(s)</td>
<td>8</td>
<td>with</td>
<td>6</td>
<td>fill</td>
<td>4</td>
<td>must</td>
<td>3</td>
<td>appear</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3. Word frequencies for the top 40 most common words in sample of grid puzzle instructions.

We then parsed the instructions with the Stanford PCFG parser in order to incorporate syntactic information into the semantic representations. A number of parses required substantial editing because of the aforementioned issues with imperative sentences and PP binding.

Next, we constructed semantic representations for some sentence fragments. An example of one such representation for the following sentence is illustrated below.

\[
\text{No consecutive digits touch.}
\]

\[\lambda(p) \forall x, y \ (|x - y| = 1) \rightarrow \neg p(x, y)\]
In (4), \( p \) refers to the predicate "touch", while \( x \) and \( y \) represent the digits. This representation can be expanded further by including a semantic representation for "touch" in the following way:

\[
\forall x, y \in \text{digits} \ (|x - y| = 1) \rightarrow \\
\forall a, b \in \text{grid cells} \ (\neg \text{contains}(a, x) \lor \neg \text{contains}(b, y) \lor \neg \text{adjacent}(a, b))
\]

In this representation we assume that we have an internal representation of the predicates \( \text{contains} \) and \( \text{adjacent} \). We finally constructed the computational semantic representations of these fragments using Haskell, as it is able to handle both lambda expressions and quantification. As an example, below is the output for (3), formatted for better readability.

\begin{verbatim}
A[x4,x5]  (conj[ digit[x4,x5], abs_diff[x4,x5] == 1 ] =>
A[x6,x7]  (cell[x6,x7] =>
            disj [~contains[x6,x4],~contains[x7,x5],~adjacent[x6,x7]]
          )
          )
\end{verbatim}

Here, \( A \) corresponds to the \( \forall \) operator, \( \text{conj} \) to the intersection, and \( \text{disj} \) to the union. As we can see, the computational semantic representation for this sentence matches well with the high-order logic.

Similarly, the following output emerges if we extend the sentence as follows:

\textit{No consecutive digits touch horizontally, or vertically, or diagonally.}
Further directions

The current investigation is part of a larger project to formulate constraints that can be useful to solve the grid puzzles based on semantic information. In this project, we explored the semantic representation of a subset of words that frequently occur in instructions of grid puzzles. To be able to represent a large number of grid puzzles, it will be necessary to extend the represented vocabulary. Another extension of the current study is to implement logical inference to increase the readability of the output and to decrease any redundancy in the semantic representation.

Visual module

In addition, instruction are not the only pieces of information that factor into the puzzle constraints. Visual input is also necessary because, as we can see in Figure 2, some puzzles have information embedded in the grid. This information cannot be extracted purely from the
instructions. A visual module that would be helpful in formulating the problem constraints would detect any numbers, colors, and outlines for each grid cell, among other characteristics. It is also essential to detect any regions in the grid that may not be explicit from the instructions. For instance, the heavily outlined boxes make for great examples for such regions. The visual module would also verify some of the extracted semantic information. If the semantic representation determines that there is a region with some kind of markings, this region must have been detected by the visual module.

**Improving the syntactic parsing**

To improve the results further, it may be beneficial to develop an algorithm that improves the Stanford PCFG parses. As discussed previously, the Stanford parser has trouble with imperative sentences and PP binding, which created some inaccurate parses. While the Stanford parser provided for a good starting point, an approach to a better parsing algorithm may be to accept the parse trees as locally valid and explore moving local branches to different nodes in the tree. There may be a penalty for moving parts of the original parse tree and a reward for improved semantic coherence. This algorithm may avoid the combinatorial explosion in possible number of new parses by considering that the sentences must describe a solvable grid puzzle.

**Formulating constraints and solving the puzzles**

Lastly, in order to be able to solve the puzzle, it will be necessary to extract the actual constraints. Once these constraints are specified, the puzzle will become a constraint satisfaction problem.
Conclusion

Using Montague semantics and a syntactic parse from the Stanford probabilistic context-free grammar parser, we were able to create a number of accurate representations of instruction sentences. While more work is needed to solve the puzzles, these results are encouraging and warrant further investigation.
References
