Natural Language Processing Assignment 2 – CSC384

Farhang Farid
Student Number: 991619068

Introduction

The purpose of this assignment was to implement a natural language processing system and to construct a chart parser in Prolog that will understand and respond to natural English queries about the domain of inventors and inventions. The application constructs a parse tree from a knowledge base and grammar.

Lexicon Design

The main issue that I had to consider while designing the lexicon was the fact that lexicon had to include all the possible words that the user might have included in his/her questions. The reason for this is the program uses a predicate in the interface file to Screen the words in the users question and match them with those in the lexicon. The words that can not be matched will be removed from the list of the words in the question and this could cause the parser not operate correctly. AS a result of this issue I have added all the words in knowledge base to my lexicon as well as some additional words that are discussed below.

1- actor/inventor: The name of the inventors is mapped to the inventor atom.

2- env: the location where the inventor has invented something is referred to env.

3- year: all the years of the inventions are mapped to the year atom.

4- object: inventions are mapped to object atoms

5- misc: this includes w clauses (who, what, where, when, whom), and words like person, something, someone.

Grammar Design

In order to come up with an efficient grammar I have generalized many of the grammatical forms of the questions that can be asked from the system. Additionally, I have also used nested lists to simplify my grammar. An example of this technique would be:

```
gram_rule(wClauseList, [wClause]).
gram_rule(wClauseList, [wClause, and, wClauseList]).
```

Where a wClauseList consists of several wClauses. This s Specific example solves the situation where user includes 2 or more W clauses in his/her example.

E.g. where and when did Thomas Edison invent light bulbs?

In general every grammatical sentence is form by combining 2 or more of the following atoms.

1- wClause: one or more wClauses combined with an and sign.

2- passAct: passive action consist of the term “was” followed by and optional objectphrase (invention) and a past tense verb.

3- actAct Active act consist of the term “did” followed by the inventorPhrase(inventor) and a present verb.

4- relList: A nested list referring to the location and the year where and invention was invented

5- objectPhrase : consist of a list of one or more objects.

The above mentioned atoms by them selves consist of smaller inner atoms.

Parser Design

I have used prolog’s inner back tracking facility to write highly optimized parser. The parser extracts the words and phrases from the question string according to grammar and lexicon rules and constructs a parse Tree. The Parsing algorithm works in a recursive manner as follows:

Given a list and two indexes x and y in the list the parse algorithm recursively checks the items of the list between these two indexes, if the phrase between the two indexes matches the format of one of the rules of the grammar the expansion tree of that phrase within the grammar rule is added as a child node to the parse tree otherwise if a word matches an exact rule in the lexicon that the expansion of that rule in the lexicon is added to its (expansion tree parent) as a leaf node. The algorithm recursively traverses through the list until it has considered all the combinations of matching words/phrases possible. Sub trees of the parse can be extracted using the index of the word in the list of question words.

Additional Features

Not only I have tried to come up with a comprehensive grammar, but I have also tried to consider the situations which were not part of the specifications. One of this additional features is the capability to answer question that contain several W clauses. These questions are recognized and parsed with in my application and the system is able to answer such questions as: “Where and when did Thomas Edison invent the light bulb?”

Additional Design Considerations

One of the other enhancements that I have made to the parsing engine is related to the role of the plural objects, the previous system required that the plural form of all the inventions be added to the lexicon in order for the system to correctly parse them. However I have improved this design by introducing two small predicates in events and lexicon files to avoid the introduction of several new rules. The lexicon rule simply creates a list of all the objects in the lexicon + the letter s and adds them to the knowledge base and the rule in the event frame removes the s from the end of the words with the object role so the engine can match these words with those existing in the knowledge base.

Analysis

To do any sort of analysis for this assignment I needed to compare my coverage against a standard and although the only standard that I have was the grammatical format of the sentences that I came up with I still tried to come up with an estimation of the type of the sentences that my system is currently able to parse and respond to:

1- Sentences starting with W clause (which, where, when, who, what) in passive, active, present and path formats.

   e.g. where did Edison invent the light bulb

   where was the light bulb invented

   who invented the light bulb

   who is the inventor of the light bulb

2- Multiple objects(inventions) in a sentence

3- Multiple w clauses in a sentence

4- Sentences ending with whom.

5- Sentences including something, someone, somewhere

   e.g. where did someone invent the light bulb

6- Multiple relList in a sentence (in loc, in time)

   e.g. what did Thomas Invent in 1879 in Menlo park

I have put in a lot of effort to make sure that my program is able to respond to most type of grammatically correct/incorrect questions, however it is clear that it’s simply impossible to do so in limited amount of time. As an additional feature to the program given more time, I would like to extend my grammar as well as improving the engine such that its not completely dependent on the lexicon and it can automatically locate the goals (only) of a question and answer to them.