PROJECT REPORT

Computational Linguistics CSE 567

Authors

Akshay Chopra 50248989
Shubham Pandey 50245725

Project Description

- We have created a Natural Language understanding System that interfaces with a 'smart' refrigerator and allows users to text queries or instructions to the fridge.
- For the system, we have created a vocabulary with lemmas from different part of speech (like nouns, pronouns, determiners, intransitive verbs, transitive verbs, adjectives etc).
- Grammar has been defined and is used by the parser.
- Parser has been created for parsing the sentence given to the system and annotating the lexical items with the corresponding syntactic and semantic representations. Parser output is fed to the model checker.
- Finally, a model checker has been created which takes the output of the parser as the input and performs an evaluation on it and responds accordingly to the user.

NOTE: The parser does not parse annotations like period (.), comma (,) or question mark (?) etc. Please enter the text in the chat without any annotations
Like: 1. who ate the chicken
2. The fridge contains fruits

Modules of the Project.pl file

Parser: Modified Shift Reduce Parser. It is based on parser_sem_sr.txt file shared in class with an added case to handle 3 POS tags at a time. This was necessary for rules like – 'ynq-> be np vp' which require combining 3 tags to form one.

Lemmas: Over 100+ Lemmas have been created for the project from 10 different part of speech.

Lexicons: Lexicons have been defined for various lemmas. These are necessary to combine and form syntactically correct sentences or questions as per the rules defined in the .pl file.

Morphological Parsing: We handled inflections of words by using simple facts using a predicate 'suffix' to mention allowed suffixes. We extract the base form of the verb using simple atom_concat rules.

Rules: Rules are responsible for merging two or more parts of speech in order to generate a meaningful sentence or question. Any ill-formed sentence is not accepted by these rules.

WordNet: Hand-built mini-WordNet fragment has been created. Hypernyms have been added for various nouns like "**apple, orange, banana** is a **type** of **fruit"**, **"chicken, ham, sausage** is a **type** of **meat"** etc. A new predicate '**isa**' has been used for achieving this task.

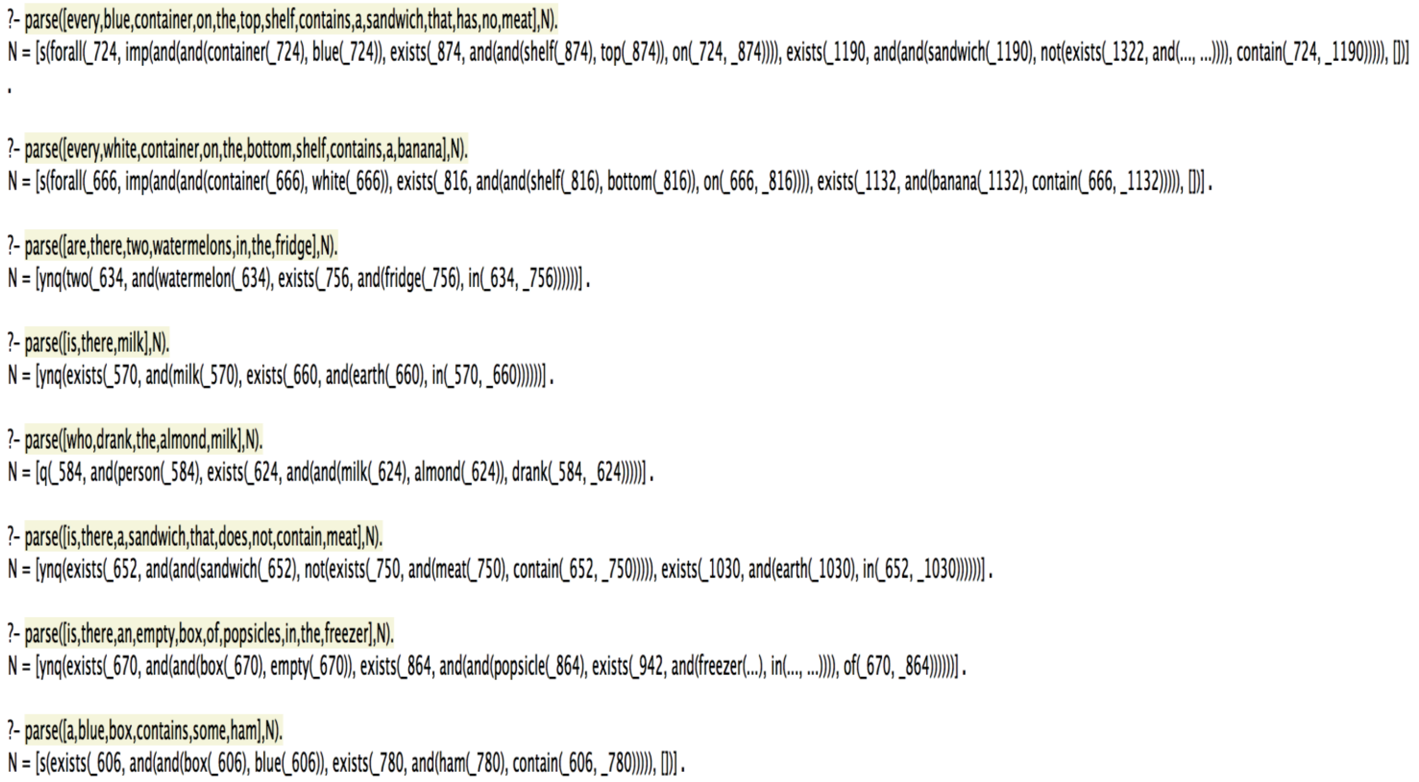
Model Checker: We have used the base code from sat.txt provided in class with some changes of our own. Model has been created containing values relevant to the examples given in the pdf plus the examples that we have added by ourselves. Model checker, sat, f and other predicates have been modified according to the project requirement. We added a new rule for f in order to handle hypernyms.

Also, in order to get all the properties of an answer, we used a findall approach inside a new predicate ‘checker’ created for exclusively content questions. There are 6 modelchecker rules - 3 positive and 3 negative cases for all 3 types of input - (sentence, yn question, content question). We added 10 new sat rules to handle numerals in a sentence. Eg – ‘are there two eggs inside the blue box’. In this example, user asks for two eggs, so we use a findall approach for every egg in the blue box and compare the length of the list obtained as the answer with 2. If greater, then the system responds with ‘yes’.

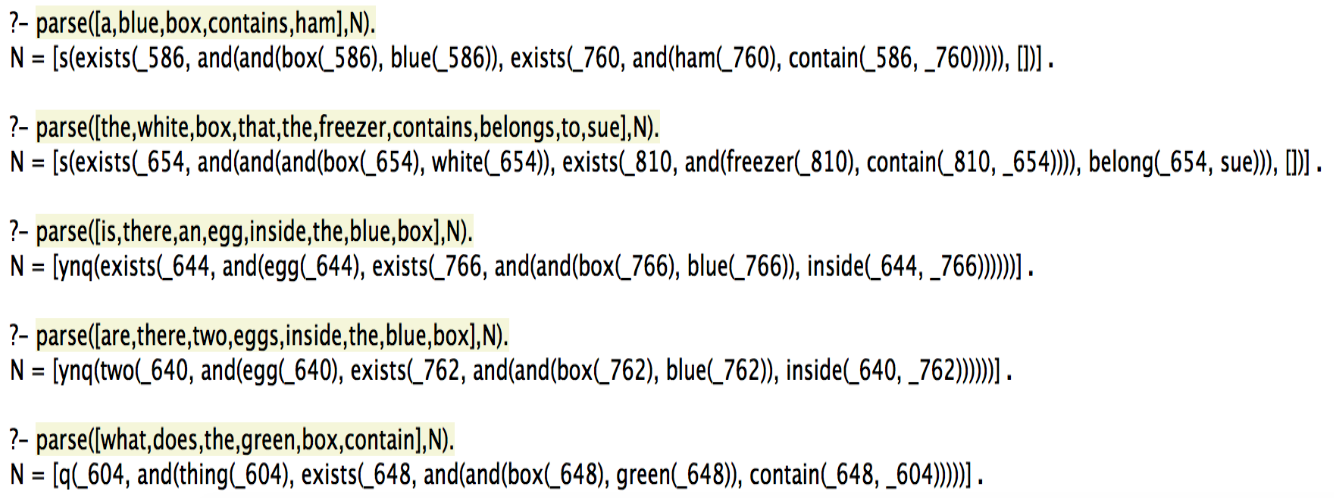
Running the Examples already given in the Project PDF

- Every blue container on the top shelf contains a sandwich that has no meat
- Every white container on the bottom shelf contains a banana
- Are there two watermelons in the fridge
- Is there milk
- Who drank the almond milk
- Is there a sandwich that does not contain meat
- Is there an empty box of popsicles in the freezer
- A blue box contains some ham
- A blue box contains ham
- The white box that the freezer contains belongs to sue
- Is there an egg inside the blue box
- Are there two eggs inside the blue box
- What does the green box contain

Parser Output for the above examples



```
?- parse([every,blue,container,on,the,top,shelf,contains,a,sandwich,that,has,no,meat],N).
N = [s(forall(_724, imp(and(and(container(_724), blue(_724)), exists(_874, and(and(shelf(_874), top(_874)), on(_724, _874)))), exists(_1190, and(and(sandwich(_1190), not(exists(_1322, and(..., ...)))), contain(_724, _1190))))), [])]
.

?- parse([every,white,container,on,the,bottom,shelf,contains,a,banana],N).
N = [s(forall(_666, imp(and(and(container(_666), white(_666)), exists(_816, and(and(shelf(_816), bottom(_816)), on(_666, _816)))), exists(_1132, and(banana(_1132), contain(_666, _1132))))), [])] .

?- parse([are,there,two,watermelons,in,the,fridge],N).
N = [ynq(two(_634, and(watermelon(_634), exists(_756, and(fridge(_756), in(_634, _756))))))] .

?- parse([is,there,milk],N).
N = [ynq(exists(_570, and(milk(_570), exists(_660, and(earth(_660), in(_570, _660))))))] .

?- parse([who,drank,the,almond,milk],N).
N = [q(_584, and(person(_584), exists(_624, and(and(milk(_624), almond(_624)), drank(_584, _624)))))] .

?- parse([is,there,a,sandwich,that,does,not,contain,meat],N).
N = [ynq(exists(_652, and(and(sandwich(_652), not(exists(_750, and(meat(_750), contain(_652, _750))))), exists(_1030, and(earth(_1030), in(_652, _1030))))))] .

?- parse([is,there,an,empty,box,of,popsicles,in,the,freezer],N).
N = [ynq(exists(_670, and(and(box(_670), empty(_670)), exists(_864, and(and(popsicle(_864), exists(_942, and(freezer(...), in(..., ...)))), of(_670, _864))))))] .

?- parse([a,blue,box,contains,some,ham],N).
N = [s(exists(_606, and(and(box(_606), blue(_606)), exists(_780, and(ham(_780), contain(_606, _780))))), [])] .
```



```
?- parse([a,blue,box,contains,ham],N).
N = [s(exists(_586, and(and(box(_586), blue(_586)), exists(_760, and(ham(_760), contain(_586, _760))))), [])] .

?- parse([the,white,box,that,the,freezer,contains,belongs,to,sue],N).
N = [s(exists(_654, and(and(and(box(_654), white(_654)), exists(_810, and(freezer(_810), contain(_810, _654)))), belong(_654, sue))), [])] .

?- parse([is,there,an,egg,inside,the,blue,box],N).
N = [ynq(exists(_644, and(egg(_644), exists(_766, and(and(box(_766), blue(_766)), inside(_644, _766))))))] .

?- parse([are,there,two,eggs,inside,the,blue,box],N).
N = [ynq(two(_640, and(egg(_640), exists(_762, and(and(box(_762), blue(_762)), inside(_640, _762))))))] .

?- parse([what,does,the,green,box,contain],N).
N = [q(_604, and(thing(_604), exists(_648, and(and(box(_648), green(_648)), contain(_648, _604)))))] .
```

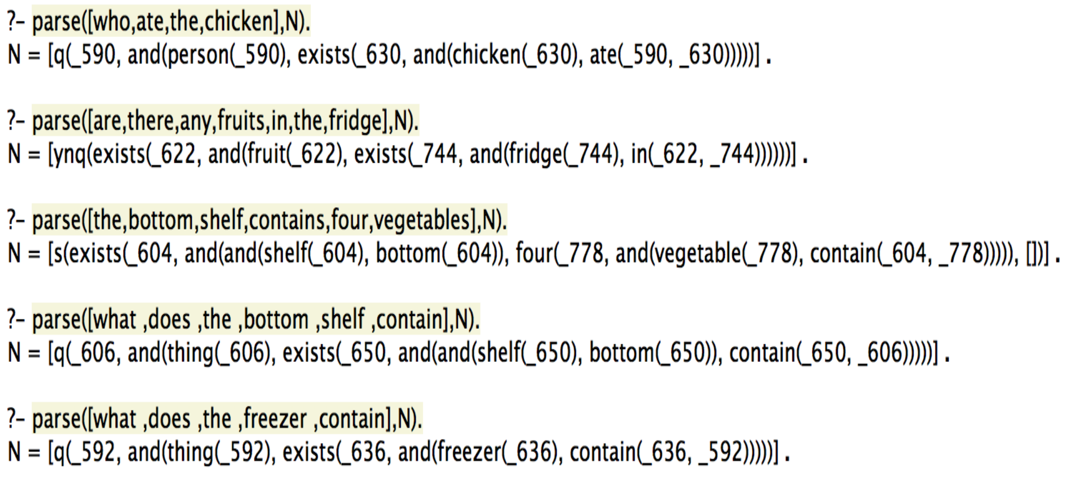
Chat result for the above examples

```
?- chat.
|: Every blue container on the top shelf contains a sandwich that has no meat

That is correct

|: Every white container on the bottom shelf contains a banana

That is not correct

|: Are there two watermelons in the fridge


yes

|: Is there milk

yes

|: Who drank the almond milk

[sam]

|: Is there a sandwich that does not contain meat

yes

|: Is there an empty box of popsicles in the freezer

yes

|: A blue box contains some ham

That is correct

|: A blue box contains ham

That is correct

|: The white box that the freezer contains belongs to sue

That is correct

|: Is there an egg inside the blue box

no

|: Are there two eggs inside the blue box

no

|: What does the green box contain

[ham,ham,red box,popsicle]

|: bye.
> bye!
```

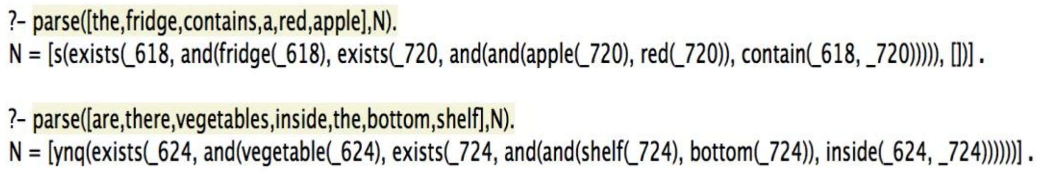
Running the Self Made Examples

- Who ate the chicken
- Are there any fruits in the fridge
- The bottom shelf contains four vegetables
- What does the bottom shelf contain
- What does the freezer contain
- The fridge contains a red apple
- Are there vegetables inside the bottom shelf

Parser output for the above examples



```
?- parse([who,ate,the,chicken],N).
N = [q(_590, and(person(_590), exists(_630, and(chicken(_630), ate(_590, _630)))))] .

?- parse([are,there,any,fruits,in,the,fridge],N).
N = [ynq(exists(_622, and(fruit(_622), exists(_744, and(fridge(_744), in(_622, _744))))))] .

?- parse([the,bottom,shelf,contains,four,vegetables],N).
N = [s(exists(_604, and(and(shelf(_604), bottom(_604)), four(_778, and(vegetable(_778), contain(_604, _778))))), [])] .

?- parse([what ,does ,the ,bottom ,shelf ,contain],N).
N = [q(_606, and(thing(_606), exists(_650, and(and(shelf(_650), bottom(_650)), contain(_650, _606)))))] .

?- parse([what ,does ,the ,freezer ,contain],N).
N = [q(_592, and(thing(_592), exists(_636, and(freezer(_636), contain(_636, _592)))))] .
```



```
?- parse([the,fridge,contains,a,red,apple],N).
N = [s(exists(_618, and(fridge(_618), exists(_720, and(and(apple(_720), red(_720)), contain(_618, _720))))), [])] .

?- parse([are,there,vegetables,inside,the,bottom,shelf],N).
N = [ynq(exists(_624, and(vegetable(_624), exists(_724, and(and(shelf(_724), bottom(_724)), inside(_624, _724))))))] .
```

Chat Result for the above examples

```
?- chat.
|: Who ate the chicken

[sue]

|: Are there any fruits in the fridge

yes

|: The bottom shelf contains four vegetables


That is correct

|: What does the bottom shelf contain

[tomato,onion,carrot,spinach]

|: What does the freezer contain

[popsicle,icecream,white empty box]

|: The fridge contains a red apple

That is correct

|: Are there any vegetables inside the bottom shelf

yes

|: bye.
> bye!
true.
```