Maciej Girek: 652037379, mgirek2
Manuel Adrianzen: 674487564, madria3
CS 421 Natural Language Processing
UIC Spring 2019

Term Project: Part 1

We chose to use Stanford Core NLP Parser along with Python NLTK library because we were able to use it along with Python language which we prefer over java due to its readiness, being more familiar with this language, and being able to write computationally powerful code with fewer lines. Besides that, we spend few days researching libraries and technologies that would be best suited for this task and it again occurred to us that Stanford Libraries are best equipped whit tools needed to complete part 1 of this project. Here are some websites that helped us make this decision:

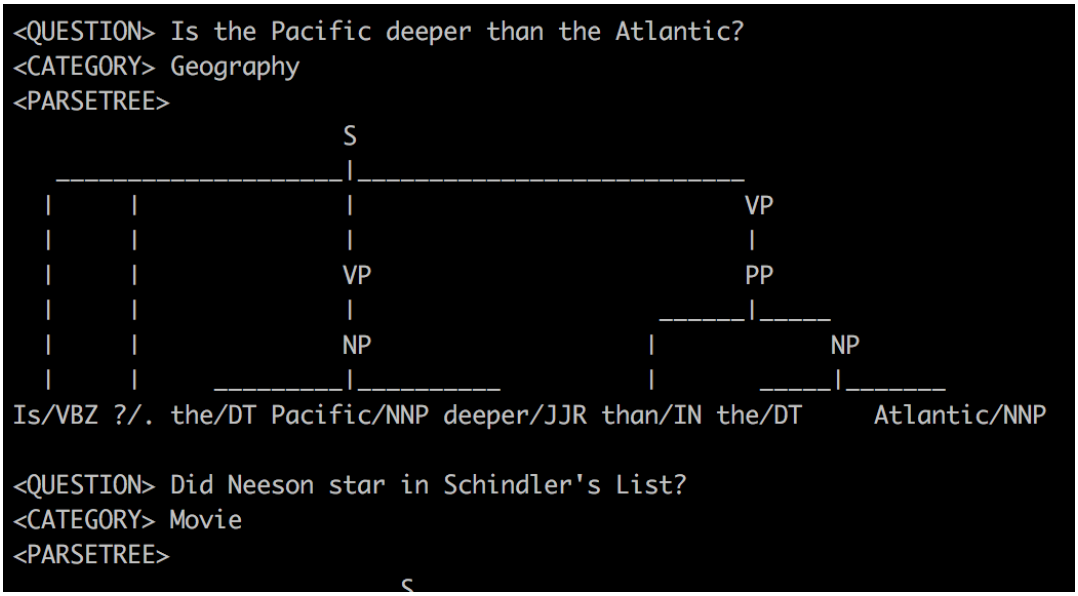
1. https://towardsdatascience.com/5-heroic-tools-for-natural-language-processing-7f3c1f8fc9f0

2. https://www.researchgate.net/post/What_is_the_best_natural_language_tool_to_recognize_the_Part_of_Speech

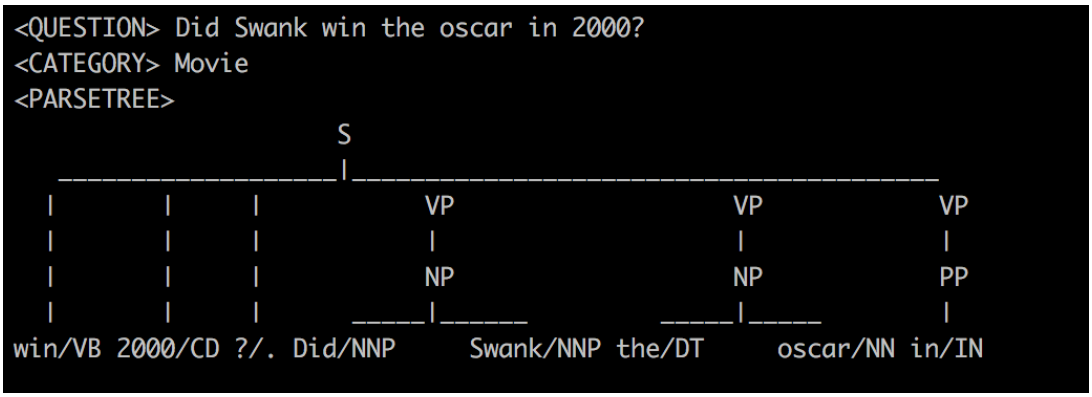
3. https://nlp.stanford.edu/software/lex-parser.shtml

Parse Trees: -

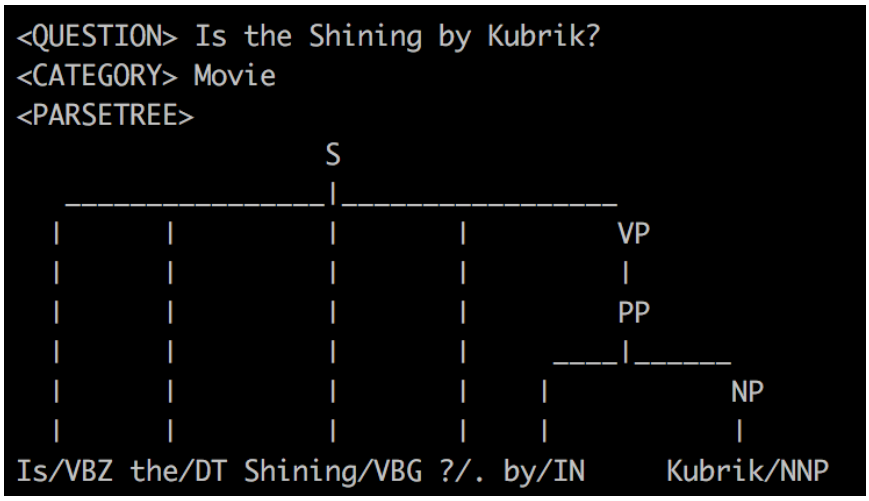
1c) Is the Pacific deeper than the Atlantic?



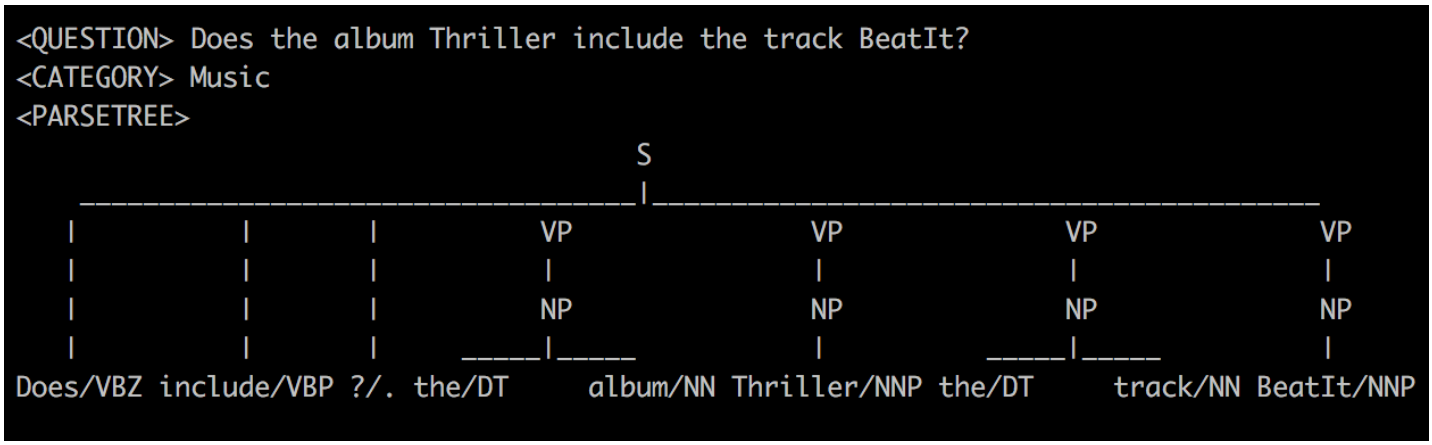
1e) Did Swank win the oscar in 2000?



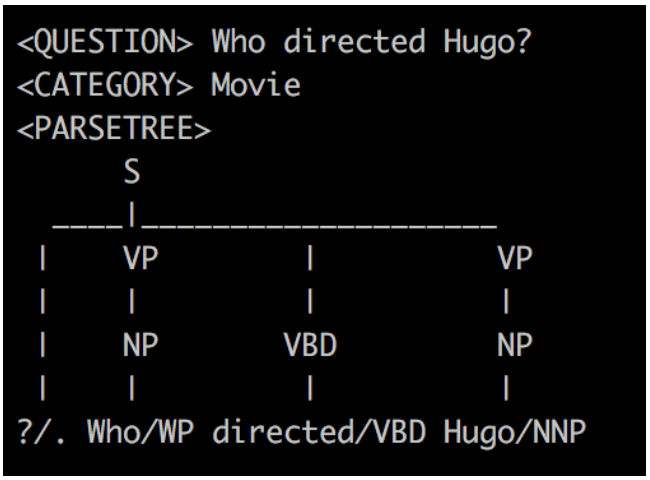
1f) Is the Shining by Kubrik?



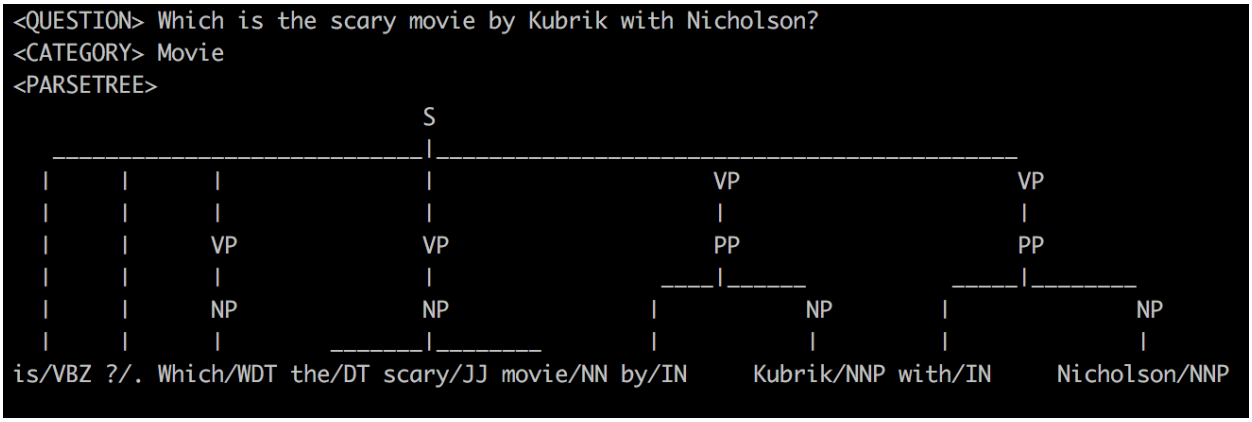
1j) Does the album Thriller include the track BeatIt?



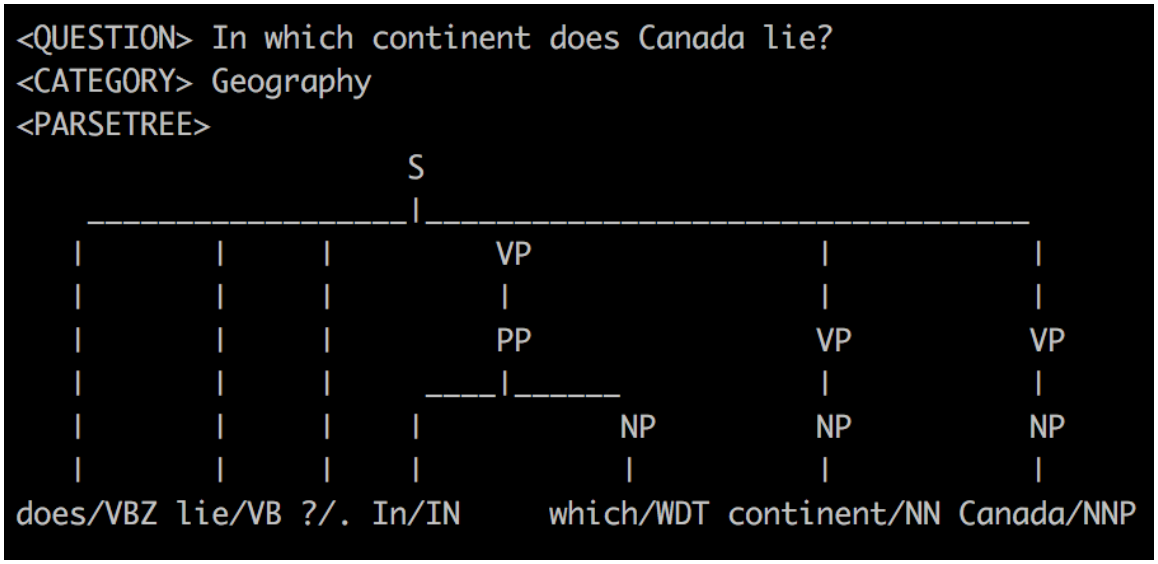
2a) Who directed Hugo?



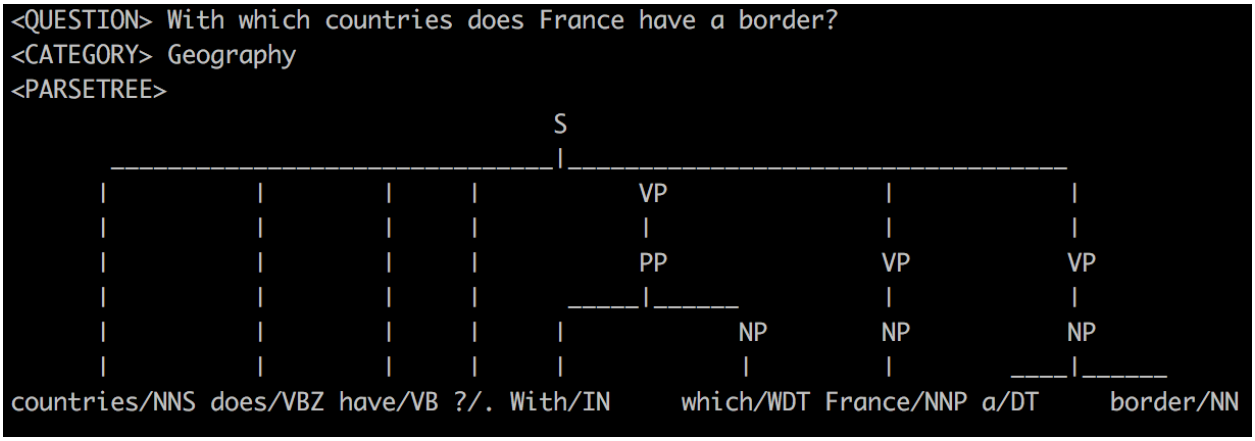
2b) Which is the scary movie by Kubrik with Nicholson?



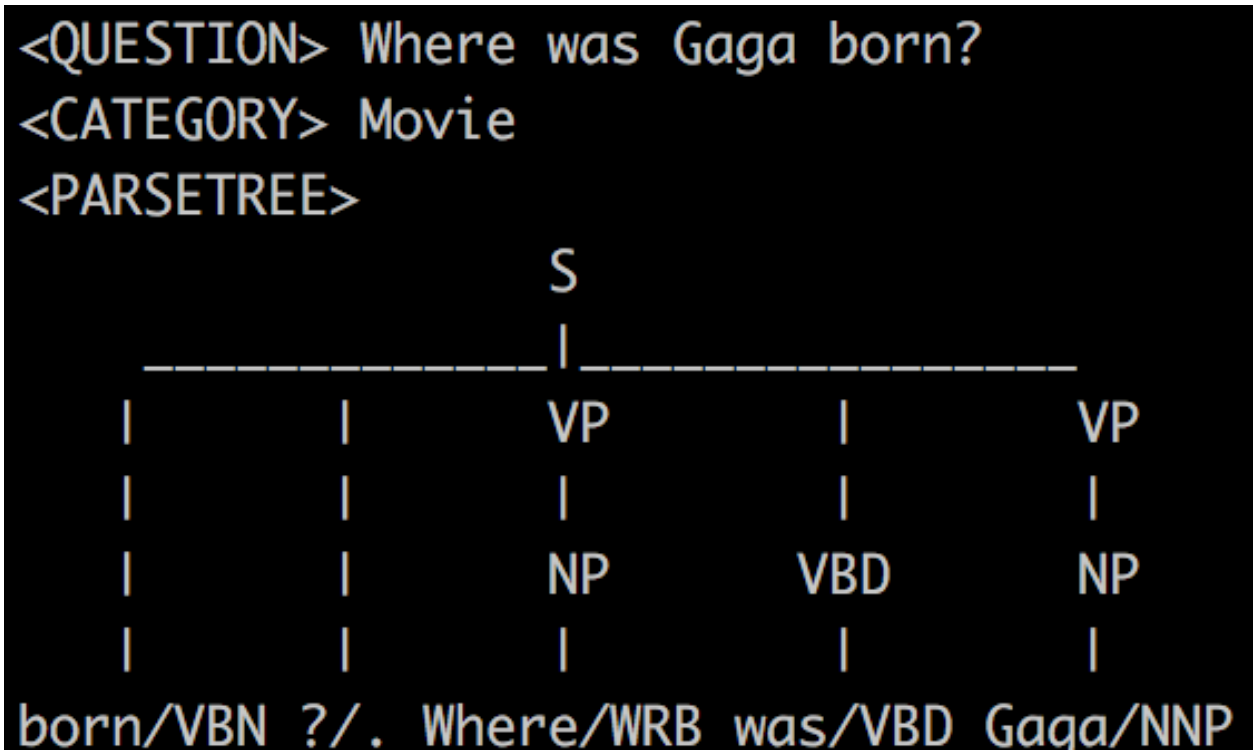
2f) In which continent does Canada lie?



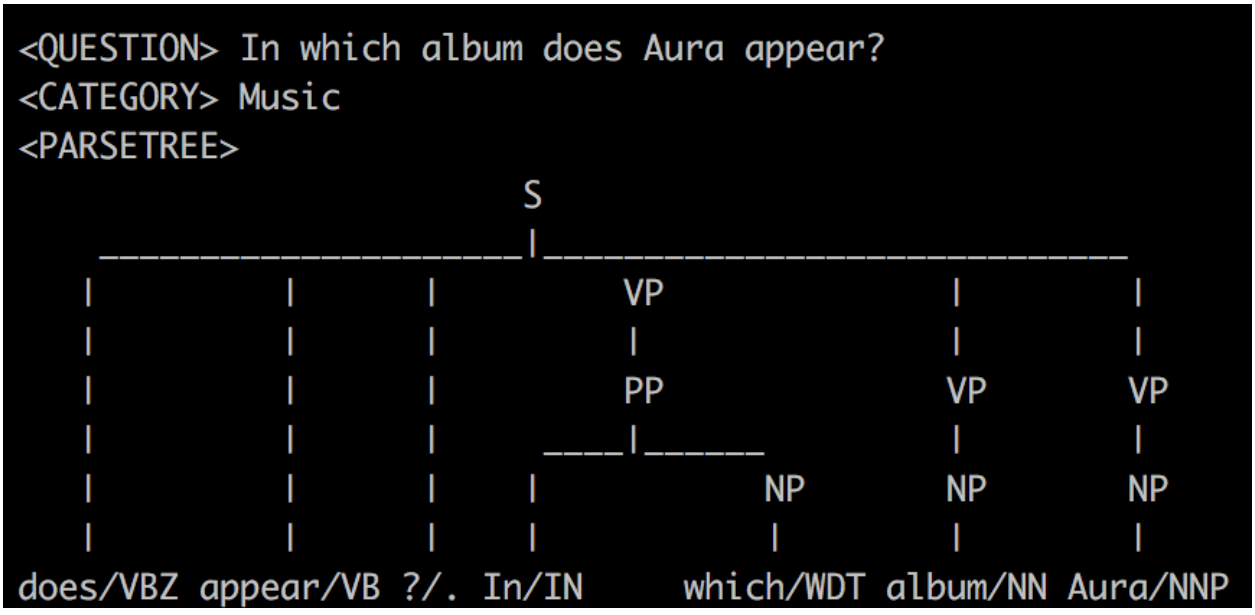
2h) With which countries does France have a border?



2m) Where was Gaga born?



2n) In which album does Aura appear?



2o) Which album by Swift was released in 2014?

```
<QUESTION> Which album by Swift was released in 2014?
<CATEGORY> Music
<PARSETREE>
                                        S
     _________________________________|_____________________________________
    |           |     |      |        |             VP                |      |
    |           |     |      |        |             |                 |      |
    |           |     |      VP       VP            PP                |      VP
    |           |     |      |        |          ____|______          |      |
    |           |     |      NP       NP        |           NP       VBD     PP
    |           |     |      |        |         |           |         |      |
released/VBN 2014/CD ?/. Which/WDT album/NN by/IN      Swift/NNP was/VBD in/IN
```

CODE:

```
def preprocess(sent):
    sent = nltk.word_tokenize(sent)
    sent = nltk.pos_tag(sent)
    return sent

def largestSimilarity(geoPercenatge,musicPercentage,moviePercentage):
    if( geoPercenatge > musicPercentage):
        if( geoPercenatge > moviePercentage):
            return 'Geography'
    elif( musicPercentage > geoPercenatge):
        if( musicPercentage > moviePercentage):
            return 'Music'
    elif ( moviePercentage > geoPercenatge):
        if ( moviePercentage > musicPercentage):
            return 'Movie'
    elif (moviePercentage == geoPercenatge):
        return 'Movie, Geography'
    elif (musicPercentage == geoPercenatge):
        return 'Music, Geography'
    elif (moviePercentage == musicPercentage):
        return 'Movie, Music'
    else:
        # should never reach
        return 'Movie ' + str(moviePercentage) + ' Geography ' + str(geoPercenatge) + 'Music ' + str(musicPercentage)

def nounSimilarity(noun):

    w1 = wordnet.synset(str(noun)+'.n.01')
    w2 = wordnet.synset('location.n.01')
    w3 = wordnet.synset('album.n.01')
    w4 = wordnet.synset('movie.n.01')

    geoPercenatge = w1.path_similarity(w2);
    musicPercentage = w1.path_similarity(w3);
    moviePercentage = w1.path_similarity(w4);
    return largestSimilarity(geoPercenatge,musicPercentage,moviePercentage)

```

```
127 def ProperNounNER(word):
128     words = nltk.word_tokenize(word)
129     tupple = ner_tagger.tag(words)
130 
131     # return the NER tag
132     tag = tupple[0][1]
133 
134     if tag == 'LOCATION':
135         tag =  'Geography'
136     else if tag == 'PERSON':
137         tag =  'Movie, Music'
138     return tag
139 
140 
141 def main():
142 
143     # Get sentences from input file
144     with open('input.txt', 'r') as f:
145         data = [line.strip() for line in f]
146     #POS tag sentences
147     taggedSentences = []
148     from nltk import Tree
149 
150     pattern = """NP: {(<DT>?<NNP>+<JJR>) | (<DT>?<NN>+<JJR>)  | (<DT>?<JJ>*<NN>)|(<DT>?<JJ>*<NNP>+) | (<DT>?<NN>+<JJS>) | (<DT>?<JJS><NN>+) | (<WDT>?<WRB>?<WP>?)}
151         VBD: {<VBD>}
152         PP: { <IN><NP> | <IN>}
153         IN: {<IN>}
154         VP: {<V> | <NP> | <PP> | <NP><PP>+ }"""
155 
156     for s in data:
157         taggedSentences.append(nltk.pos_tag(nltk.word_tokenize(s)))
158 
159     #print(taggedSentences)
160 
161     NounList = []
162     categories = []
163 
164     for sentence in taggedSentences:
165         element = ''
166         for i,j in sentence:
167             if( j == 'NN'):
168                 if( element == ''):
169                     element += nounSimilarity(i)
170             if (j == 'NNP'):
171                 if( ProperNounNER(i) != None):
-- INSERT --
```

```
146     #POS tag sentences
147     taggedSentences = []
148     from nltk import Tree
149 
150     pattern = """NP: {(<DT>?<NNP>+<JJR>) | (<DT>?<NN>+<JJR>)  | (<DT>?<JJ>*<NN>)|(<DT>?<JJ>*<NNP>+) | (<DT>?<NN>+<JJS>) | (<DT>?<JJS><NN>+) | (<WDT>?<WRB>?<WP>?)}
151         VBD: {<VBD>}
152         PP: { <IN><NP> | <IN>}
153         IN: {<IN>}
154         VP: {<V> | <NP> | <PP> | <NP><PP>+ }"""
155 
156     for s in data:
157         taggedSentences.append(nltk.pos_tag(nltk.word_tokenize(s)))
158 
159     #print(taggedSentences)
160 
161     NounList = []
162     categories = []
163 
164     for sentence in taggedSentences:
165         element = ''
166         for i,j in sentence:
167             if( j == 'NN'):
168                 if( element == ''):
169                     element += nounSimilarity(i)
170             if (j == 'NNP'):
171                 if( ProperNounNER(i) != None):
172                     if( element == ''):
173                         element += ProperNounNER(i)
174         categories.append(element)
175         if( element != ''):
176             sentence.append(element)
177 
178     counter = 0
179     for s in data:
180         print("<QUESTION> " + s)
181         print("<CATEGORY> " + categories[counter])
182         print("<PARSETREE>")
183         result = nltk.pos_tag(nltk.word_tokenize(s))
184         NPChunker = nltk.RegexpParser(pattern)
185         result = NPChunker.parse(result)
186         Tree.fromstring(str(result)).pretty_print()
187         counter+=1;
188 
189 if __name__ == "__main__":
190     main()
-- INSERT --
```