NLP Program3

Group members:
Amor Tsai
Kuo Wang
Anan Long

1. Techniques and discussions

To generate questions from a sentence, the techniques we used are dependency parsing, tagging, noun chunking, part of speech, named entities and wordnet.

Given a sentence, we use dependency parsing from spacy to get its root, so that we can rebuild the sentence by its subtree. To identify whether using what, when, who, the program should know the structure of the sentence and named entities. Sometimes, the name entities from spacy can't detect the noun such as "boy", "student", "book", etc. Therefore, we use wordnet from nltk to compare the words' Leacock-Chodorow Similarity between person and object, and use the word generating maximum value as the closest word.

We use first meaning of the word from wordnet to compare, thus it may sometimes causes misunderstanding. For example, the first meaning of "John" in wordnet is "toilet", which is more like an object than a person, but in most of time we see John as a person. We also tried another means of vector similarity by using glove embedding to transform two word into vectors and to calculate cosine similarity between two vectors, but the result tested was not ideal. We think using semantic role labeling may be a solution of this, which indicates that I should build a machine learning model and need a large corpus.

After we got the "What,Who,When", then we can try to rebuild the interrogative sentence. Here we hard-code the structural rule. That's awful, but we don't know how to implement them elegantly.

We reviewed code example from internet using Sandford parser to rebuild the sentence, the method he used are state machine and tree structure made by Stanford parser. We were planning to use his code and manipulate to fit my needs, but it fails in some simple cases, so eventually we decided to write our own instead of using his.

2. The way to use the program and results:

1. input single sentence and output to the terminal

example: python main.py -sentence "John reads a book?"

```
(nnlp) [liangchaoc@p017 NLP_project]$ python main.py -sentence "John reads a book?"
Single sentence input
Who reads a book ?
What does John read ?
```

2. input a file with sentences and output to the terminal
the file format should look like this(a sentence in each line):

```
John gave the book to Mary yesterday?
Apple released new product last year?
I give the box to Mary?
He is reading a book?
He reads the book?
Mary plays soccer?
John is reading a book?
Apple is looking for buying Google in $10 billions?
```

python main.py -input_file "sents.txt"

```
(nnlp) [liangchaoc@p017 NLP_project]$ python main.py -input "sents.txt"
sentences file input
Who gave the book to Mary yesterday ?
What did John give to Mary yesterday ?
Who did John give the book to yesterday ?
When did John give the book to Mary ?
-----------------------------------
Who released new product last year ?
What did Apple release last year ?
When did Apple release new product ?
-----------------------------------
Who give the box to Mary ?
What do i give to Mary ?
Who do i give the box to ?
-----------------------------------
Who is reading a book ?
What is he reading ?
-----------------------------------
Who reads the book ?
What does he read ?
-----------------------------------
Who plays soccer ?
What does Mary play ?
-----------------------------------
Who is reading a book ?
What is John reading ?
-----------------------------------
Who is looking for buying Google in $ 10 billions ?
What is Apple looking for buying in $ 10 billions ?
How much is Apple looking for buying Google in $ ?
-----------------------------------
```

if add one more parameter called output, then it will output the result into file specified instead of outputting to terminal. For examples:
python main.py -sentence "John reads a book?" -output "o.txt"
python main.py -input_file "sents.txt" -output "o.txt"