REPORT ON

CE887-7-AU – Natural Language Engineering

Assignment 2: Parsing and Word Similarity

Students name:

Atiwat Onsuwan 1802514

Yasmin Mohammad 160370

S -> NP VP
NP -> Det Nom | PropN | NP PP
Nom -> Adj Nom | N
VP -> V NP | V S | VP PP
PP -> P NP
PropN -> 'Bill'
Det -> 'the' | 'a' | an
N -> 'bear' | 'squirrel' | 'park'
Adj -> 'angry' | 'frightened'
V -> 'chased' | 'saw' | 'put' | 'eats' | 'eat'
P -> 'on'

Question 1

S1. Put the block on the table
S2. Bob chased a bear in the park along the river
S3. Bill saw Bob chase the angry furry dog

a) Which rules do you need to add to the grammar to parse S1 to S3?

Answer:

To parse these sentences, we updated the following rules to given grammar:

1. Added 'block' and 'table' to N
2. Added 'Bob' to PropN
3. Added 'river' to 'N'
4. Added Adv NP to PP
5. Create new rule Adv an added 'along'
6. Added 'furry' to Adj
7. Added 'dog' to N
8. Added 'chase' to V
9. Added 'in' to P

New Grammar:

```
S -> NP VP |VP
NP -> Det Nom | PropN | NP PP
Nom -> Adj Nom | N
VP -> V NP | V S | VP PP
PP -> P NP | Adv NP
PropN -> 'Bill' |'Bob'
Det -> 'the' | 'a' | 'an'
N -> 'bear' | 'squirrel' | 'park' | 'block' | 'table' | 'river' | 'dog'
Adj -> 'angry' | 'frightened' | 'furry'
V -> 'chased' | 'saw' | 'put' | 'eats' | 'eat' | 'chase'
P -> 'on' | 'in'
Adv -> 'along'
```

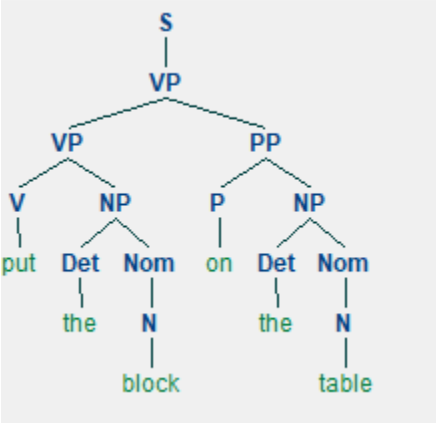
b) How many derivations can you get for each sentence?

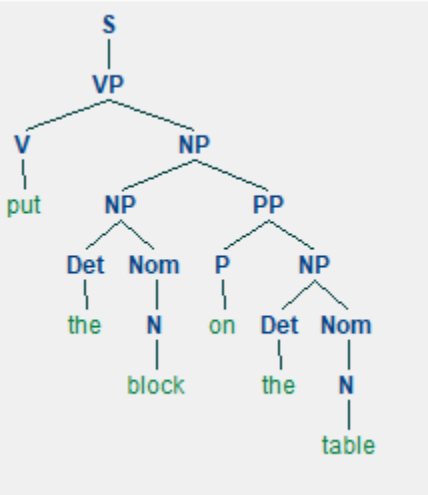
Answer:

2 derivations for S1
5 derivations for S2
1 derivation for S3

Output for S1:

```
(S
  (VP
    (VP (V put) (NP (Det the) (Nom (N block))))
    (PP (P on) (NP (Det the) (Nom (N table))))))
(S
  (VP
    (V put)
    (NP
      (NP (Det the) (Nom (N block)))
      (PP (P on) (NP (Det the) (Nom (N table)))))))
```





Output for S2:

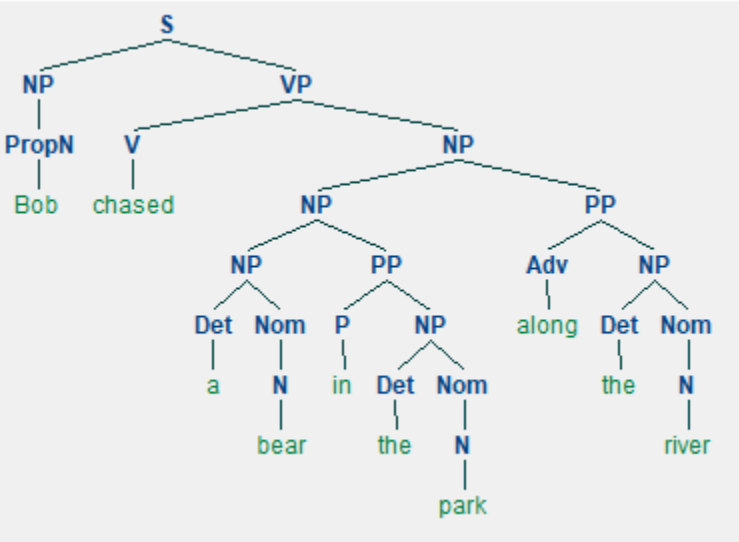
```
(S
  (NP (PropN Bob))
  (VP
    (V chased)
    (NP
      (NP
        (NP (Det a) (Nom (N bear)))
        (PP (P in) (NP (Det the) (Nom (N park)))))
      (PP (Adv along) (NP (Det the) (Nom (N river)))))))
```

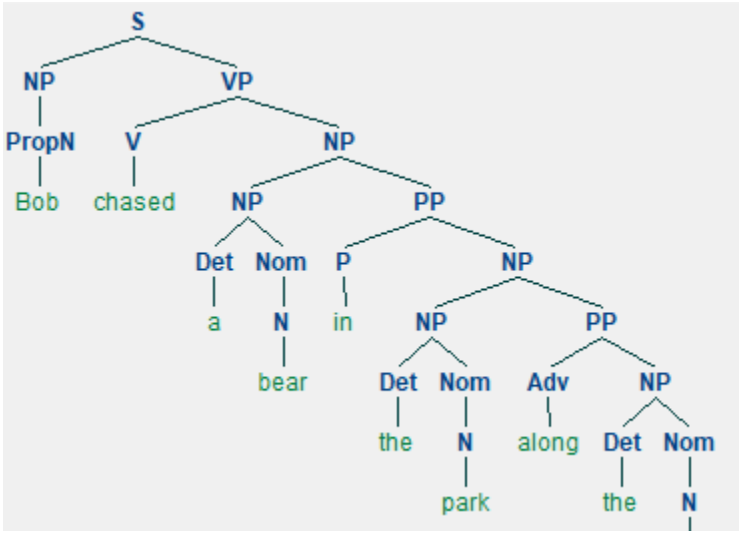
```
(S
  (NP (PropN Bob))
  (VP
    (V chased)
    (NP
      (NP (Det a) (Nom (N bear)))
      (PP
        (P in)
        (NP
          (NP (Det the) (Nom (N park)))
          (PP (Adv along) (NP (Det the) (Nom (N river))))))))))
```

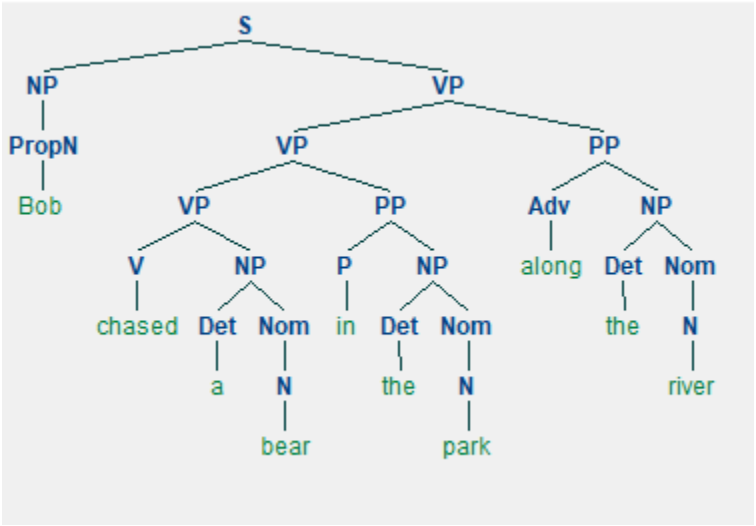
```
(S
  (NP (PropN Bob))
  (VP
    (VP
      (VP (V chased) (NP (Det a) (Nom (N bear))))
      (PP (P in) (NP (Det the) (Nom (N park)))))
    (PP (Adv along) (NP (Det the) (Nom (N river))))))
```

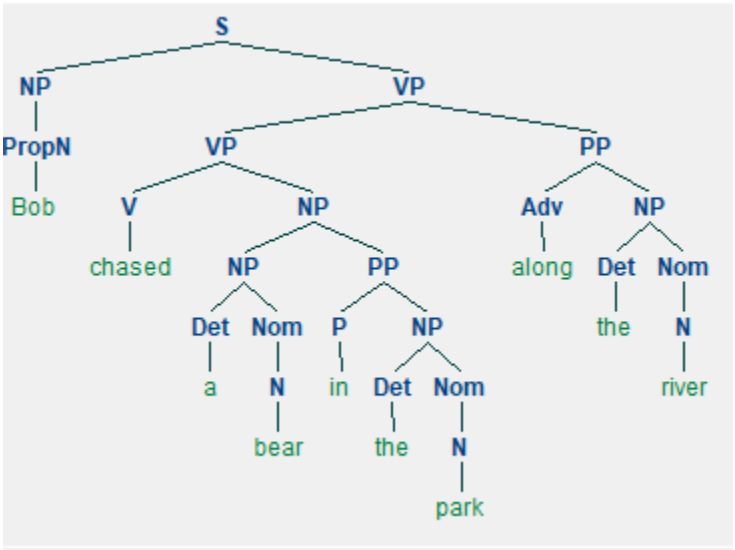
```
(S
  (NP (PropN Bob))
  (VP
    (VP
      (V chased)
      (NP
        (NP (Det a) (Nom (N bear)))
        (PP (P in) (NP (Det the) (Nom (N park))))))
    (PP (Adv along) (NP (Det the) (Nom (N river))))))
```

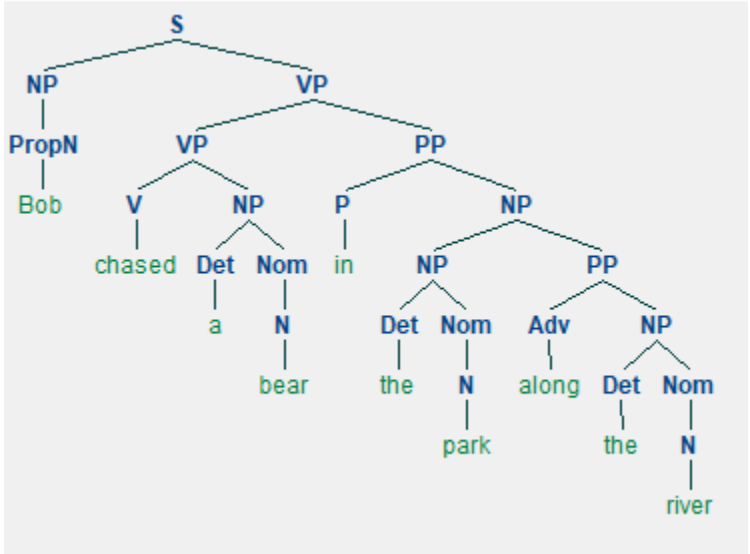
```
(S
  (NP (PropN Bob))
  (VP
    (VP (V chased) (NP (Det a) (Nom (N bear))))
    (PP
      (P in)
      (NP
        (NP (Det the) (Nom (N park)))
        (PP (Adv along) (NP (Det the) (Nom (N river))))))))
```


S
NP
VP
PropN
V
NP
Bob
chased
NP
PP
NP
PP
Adv
NP
Det
Nom
P
NP
along
Det
Nom
a
N
in
Det
Nom
the
N
bear
the
N
river
park


S
NP
VP
PropN
V
NP
Bob
chased
NP
PP
Det
Nom
P
NP
a
N
in
NP
PP
bear
Det
Nom
Adv
NP
the
N
along
Det
Nom
park
the
N

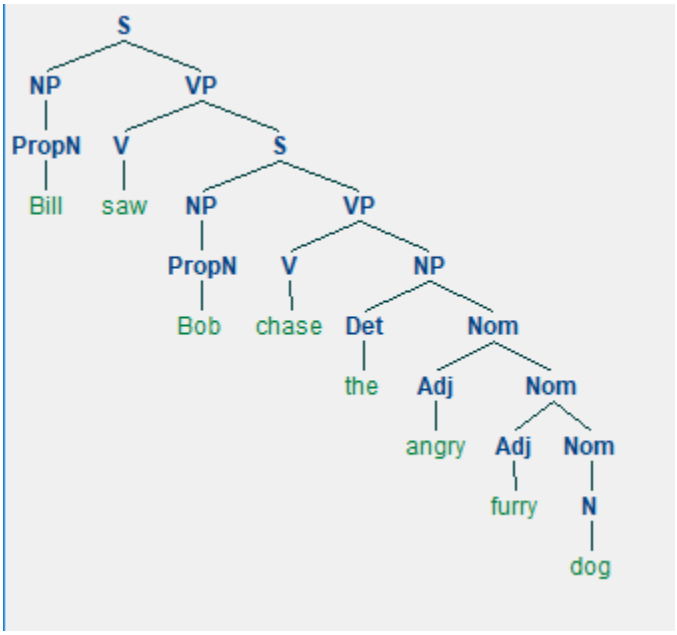

S
NP
VP
PropN
VP
PP
Bob
VP
PP
Adv
NP
V
NP
P
NP
along
Det
Nom
chased
Det
Nom
in
Det
Nom
the
N
a
N
the
N
river
bear
park


S
NP
VP
PropN
VP
PP
Bob
V
NP
Adv
NP
chased
NP
PP
along
Det
Nom
Det
Nom
P
NP
the
N
a
N
in
Det
Nom
river
bear
the
N
park


S
NP
VP
PropN
VP
PP
Bob
V
NP
P
NP
chased
Det
Nom
in
NP
PP
a
N
Det
Nom
Adv
NP
bear
the
N
along
Det
Nom
park
the
N
river

Output for S3:

```
(S
  (NP (PropN Bill))
  (VP
    (V saw)
    (S
      (NP (PropN Bob))
      (VP
        (V chase)
        (NP
          (Det the)
          (Nom (Adj angry) (Nom (Adj furry) (Nom (N dog)))))))))
```



Question 2

S4. An bear eat an squirrel
S5. The dogs eats

a) Are these sentences correct? What are the grammatically correct equivalents for these sentences?

Answer:

S4 is incorrect because of following reasons:

1. 'An' is used for vowels sound but 'bear' is consonant sound
2. 'an' is used for vowels sound but again 'squirrel' is consonant sound
3. 'eat' is used for first and second-person but 'bear' is third-person here

So, we can change S4 to "A bear eats a squirrel"

S5 is incorrect because of following reason:

1. 'eats' is used for a third-person but 'dogs' in third-person plural

So, we can change S5 to "The dogs eat"

b) Run 2 parsers from NLTK on the two sentences. What is the output of the parsers? (Explain why the parsers are correct or incorrect.

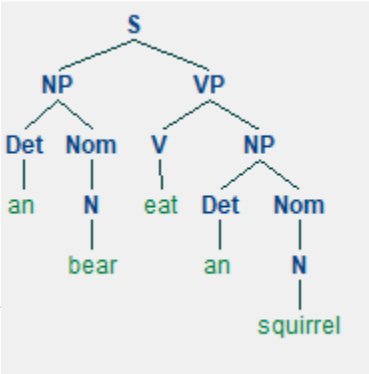
Answer:

Based on the given grammar I have run 2 parsers which are ChartParser and ShiftReduceParser and received the following outputs:

Output for S4 (ChartParser):

This ChartParser returned the output but the grammar is incorrect, so we got the incorrect derivation and tree (the given grammar does not have the rules as explained in "a)")

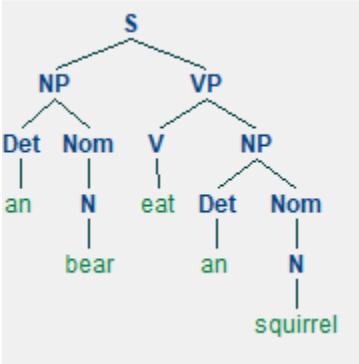
```
(S
  (NP (Det an) (Nom (N bear)))
  (VP (V eat) (NP (Det an) (Nom (N squirrel)))))
```



Output for S4 (ShiftReduceParser):

This ShiftReduceParser returned the output but the grammar is incorrect, so we got the incorrect derivation and tree (the given grammar does not have the rules as explained in “a)”)

```
(S
  (NP (Det an) (Nom (N bear)))
  (VP (V eat) (NP (Det an) (Nom (N squirrel)))))
```



Output for S5 (ChartParser):

This ChartParser did not return the output because the given grammar does not have the word ‘dogs’ in it

```
ValueError: Grammar does not cover some of the input words: "'dogs'".
```

Output for S5 (ShiftReduceParser):

This ShiftReduceParser did not return the output because the given grammar does not have the word ‘dogs’ in it

```
ValueError: Grammar does not cover some of the input words: "'dogs'".
```

c) Generate 2 other correct and 2 other incorrect sentences with this grammar. How would you have to change this grammar to prevent these sentences from being parsed? You can write your own rules to extend the grammar and ensure correct agreement.

Answer:

First correct sentence: An angry squirrel saw the bear.
Second correct sentence: The angry squirrel eats the frightened bear.

First incorrect sentence: The angry squirrel saw an bear.
Second incorrect sentence: The angry squirrel eat the frightened bear.

Explanations: 'an bear' is wrong in 1st incorrect sentence
'squirrel eat' is wrong in 2nd incorrect sentence

To prevent these two incorrect sentences from being parse we have added the following rules to grammar:

1. Created 'NP1' and 'VP1'
2. Added 'Det Nom1', 'PropN', 'NP1 PP', 'Detvowels Nvowels' to NP1
3. Added 'Vplural NP1', 'Vplural S', 'VP1 PP' to 'VP1'
4. Created 'Nom1' and added 'Adj Nom1', 'Nplural'
5. Created 'Detvowels' and moved 'an' from Det
6. Created 'Nvowels' and added 'owl' (as an example)
7. Created 'Nplural' an added 'squirrel' (as an example)
8. Created 'Vplural an moved 'eat' from 'V'

New Grammar:

```
S -> NP VP | NP1 VP1
NP -> Det Nom | PropN | NP PP | Detvowels Nvowels
NP1 -> Det Nom1 | PropN | NP1 PP | Detvowels Nvowels
Nom -> Adj Nom | N
Nom1 -> Adj Nom1 | Nplural
VP -> V NP | V S | VP PP
VP1 -> Vplural NP1 | Vplural S | VP1 PP
PP -> P NP
PropN -> 'Bill'
Det -> 'the' | 'a'
Detvowels -> 'an'
N -> 'bear' | 'squirrel' | 'park'
Nvowels -> 'owl'
Nplural -> 'squirrels'
Adj -> 'angry' | 'frightened'
V -> 'chased' | 'saw' | 'put' | 'eats'
Vplural -> 'eat'
P -> 'on'
```

Question 3

S6. He eats pasta with some anchovies in the restaurant
S7. He eats pasta with a fork in the restaurant

a) Do S6 and S7 have more than one interpretation? If so, draw all derivations and briefly describe each of the interpretations.

Answer:

Both of the S6 and S7 each of them has more than 1 derivation as follows:

S6 has 2 derivations
S7 has 2 derivations

After we created grammar rules for these sentences, we got the outputs:

Output S6:

S6 Grammer Derivations:

```
(S
  (NP (N he))
  (VP
    (VP
      (VP (V eats) (NP (N pasta)))
      (PP (P with) (NP (Det some) (N anchovies))))
    (PP (P in) (NP (Det the) (N restaurant)))))
```

S6 Grammer Derivations:

```
(S
  (NP (N he))
  (VP
    (VP
      (V eats)
      (NP (N pasta) (PP (P with) (NP (Det some) (N anchovies)))))
    (PP (P in) (NP (Det the) (N restaurant)))))
```

Output S7:

S7 Grammer Derivations:

```
(S
  (NP (N he))
  (VP
    (VP
      (VP (V eats) (NP (N pasta)))
      (PP (P with) (NP (Det a) (N fork))))
    (PP (P in) (NP (Det the) (N restaurant)))))
```

S7 Grammer Derivations:

```
(S
  (NP (N he))
  (VP
    (VP (V eats) (NP (N pasta) (PP (P with) (NP (Det a) (N fork)))))
    (PP (P in) (NP (Det the) (N restaurant)))))
```

b) Run the Shift Reduce Parser and the Earley Chart Parser from NLTK on these sentences. Which of the parsers detects the ambiguity for S6 and S7?

Answer:

Shift Reduce Parser detected the ambiguity for S6 and S7 because for Shift Reduce Parser sentences should be unambiguity but for Earley Chart Parser it came to infinite loops.

Question 4

Task1: Build a program to calculate word similarity in BioSim-100.txt using WordNet. For each word pair in BioSim-100.txt you will calculate the WordNet similarity between the pair, using the path similarity function implemented in NLTK, and print this into a file, along with the gold standard similarity.

Example output: (For full output see the BioSim-100-predicted.txt)

```
word1   word2   GoldSimilarity  WordNetSimiliarity
old new 1.58    0.0
smart   intelligent 9.2 0.25
hard    difficult   8.77    1.0
happy   cheerful    9.55    0.0
hard    easy    0.95    0.0
fast    rapid   8.75    0.25
happy   glad    9.17    1.0
short   long    1.23    0.25
stupid  dumb    9.58    0.0
weird   strange 8.93    0.0
wide    narrow  1.03    0.0
bad awful   8.42    0.0
easy    difficult   0.58    0.0
bad terrible    7.78    0.0
hard    simple  1.38    0.0
smart   dumb    0.55    0.25
insane  crazy   9.57    0.0
happy   mad 0.95    0.0
large   huge    9.47    0.0
hard    tough   8.05    1.0
new fresh   6.83    1.0
sharp   dull    0.6 0.0
quick   rapid   9.7 0.125
dumb    foolish 6.67    0.0
wonderful   terrific    8.63    1.0
strange odd 9.02    0.0
happy   angry   1.28    0.0
narrow  broad   1.18    0.25
simple  easy    9.4 0.0
old fresh   0.87    0.0
apparent    obvious 8.47    0.0
inexpensive cheap   8.72    1.0
nice    generous    5   0.0
weird   normal  0.72    0.1111111111111111
weird   odd 9.2 0.0
bad immoral 7.62    0.0
sad funny   0.95    0.0
wonderful   great   8.05    0.0
guilty  ashamed 6.38    0.0
beautiful   wonderful   6.5 0.0
confident   sure    8.27    0.0
dumb    dense   7.27    1.0
large   big 9.55    1.0
```

Task2: Build a program to detect word similarity in other texts. You will need to pre-process the user specified input text, reading the file, performing sentence splitting, tokenization and lemmatization, and removing stopwords and punctuation. The resulting file should contain only content words, one word per line. For each word in the file you will calculate the WordNet path similarity between the pair, and print this into a file. Now apply your program to the file text1.txt.

Example output: (for full output see the original-pairs.txt because the output is very huge 230k+ lines)

```
pleasure    necktie 0.07142857142857142
pleasure    bright  0.0
pleasure    ribbon  0.125
pleasure    citified    0.0
pleasure    ate 0.09090909090909091
pleasure    vitals  0.09090909090909091
pleasure    stared  0.0
pleasure    splendid    0.0
pleasure    higher  0.0
pleasure    nose    0.125
pleasure    finery  0.07692307692307693
pleasure    shabbier    0.0
pleasure    outfit  0.1
pleasure    seemed  0.0
pleasure    grow    0.0
pleasure    neither 0.0
pleasure    spoke   0.07692307692307693
pleasure    moved   0.0
pleasure    sidewise    0.0
pleasure    circle  0.16666666666666666
pleasure    kept    0.0
pleasure    eye 0.125
pleasure    finally 0.0
concerned   chapter 0.1111111111111111
concerned   tom 0.1111111111111111
concerned   answer  0.25
concerned   gone    0.3333333333333333
concerned   boy 0.1111111111111111
concerned   wonder  0.2
concerned   old 0.3333333333333333
concerned   lady    0.1
concerned   pulled  0.25
concerned   spectacle   0.125
concerned   looked  0.3333333333333333
concerned   room    0.2
concerned   put 0.25
concerned   seldom  0.3333333333333333
concerned   never   0.3333333333333333
concerned   small   0.3333333333333333
concerned   thing   0.25
concerned   state   0.25
concerned   pair    0.2
concerned   pride   0.25
concerned   heart   0.125
concerned   built   0.3333333333333333
```

Task3: Replace each word by its hypernym and calculate the similarities between each word pair printing this additional information to the file original-pairs-hypernyms.txt.

Example output: (for full output see the original-pairs-hypernyms.txt because the output is very huge 230k+ lines)

```
blue    ribbon  0.16666666666666666 chromatic_color object  0.1111111111111111
blue    citified    0.2 chromatic_color adjust  0.00
blue    ate 0.16666666666666666 chromatic_color None    0.00
blue    vitals  0.09090909090909091 chromatic_color organ   0.08333333333333333
blue    stared  0.16666666666666666 chromatic_color look    0.125
blue    splendid    0.2 chromatic_color None    0.00
blue    higher  0.2 chromatic_color None    0.00
blue    nose    0.16666666666666666 chromatic_color chemoreceptor   0.07142857142857142
blue    finery  0.25    chromatic_color attire  0.07142857142857142
blue    shabbier    0.2 chromatic_color None    0.00
blue    outfit  0.25    chromatic_color unit    0.1111111111111111
blue    seemed  0.16666666666666666 chromatic_color be  0.08333333333333333
blue    grow    0.25    chromatic_color change  0.125
blue    neither 0.2 chromatic_color None    0.00
blue    spoke   0.2 chromatic_color support 0.09090909090909091
blue    moved   0.25    chromatic_color None    0.00
blue    sidewise    0.2 chromatic_color None    0.00
blue    circle  0.25    chromatic_color ellipse 0.1
blue    kept    0.2 chromatic_color None    0.00
blue    eye 0.16666666666666666 chromatic_color sense_organ 0.07692307692307693
blue    finally 0.2 chromatic_color None    0.00
cloth   chapter 0.08333333333333333 artifact    section 0.2
cloth   tom 0.125   artifact    Black   0.25
cloth   answer  0.1 artifact    statement   0.1111111111111111
cloth   gone    0.0 artifact    None    0.00
cloth   boy 0.125   artifact    male    0.16666666666666666
cloth   wonder  0.09090909090909091 artifact    astonishment    0.1
cloth   old 0.09090909090909091 artifact    past    0.1111111111111111
cloth   lady    0.1111111111111111  artifact    woman   0.14285714285714285
cloth   pulled  0.0 artifact    move    0.08333333333333333
cloth   spectacle   0.16666666666666666 artifact    sight   0.1111111111111111
cloth   looked  0.0 artifact    None    0.00
cloth   room    0.2 artifact    area    0.3333333333333333
cloth   put 0.06666666666666667 artifact    option  0.09090909090909091
cloth   seldom  0.0 artifact    None    0.00
cloth   never   0.0 artifact    None    0.00
cloth   small   0.1111111111111111  artifact    body_part   0.14285714285714285
cloth   thing   0.3333333333333333  artifact    situation   0.14285714285714285
cloth   state   0.1111111111111111  artifact    administrative_district 0.14285714285714285
cloth   pair    0.09090909090909091 artifact    set 0.25
cloth   pride   0.1 artifact    feeling 0.1111111111111111
cloth   heart   0.125   artifact    intuition   0.09090909090909091
cloth   built   0.0 artifact    make    0.07142857142857142
cloth   style   0.16666666666666666 artifact    property    0.25
cloth   service 0.16666666666666666 artifact    work    0.25
cloth   could   0.0 artifact    None    0.00
cloth   seen    0.0 artifact    perceive    0.00
cloth   stovelids   0.0 artifact    None    0.00
cloth   well    0.25    artifact    excavation  0.5
cloth   perplexed   0.0 artifact    confuse 0.00
cloth   moment  0.1 artifact    point   0.25
cloth   said    0.0 artifact    express 0.2
cloth   fiercely    0.0 artifact    None    0.00
cloth   still   0.16666666666666666 artifact    photograph  0.25
```

Task4: What are the 10 most similar pairs that you found for text1.txt? Print them to the file top.txt.

Example output: (In this output we have shuffled the order of the list of data because in this case we have more than 10 pairs which contained the maximum path similarity so that we can see the top ten path similarity differently)

We have tried to run the program 3 times and the outputs as following:

1st:

word1	word2	Similarity
gave	broke	1.0
look	searched	1.0
question	wonder	1.0
tell	says	1.0
wait	hold	1.0
gave	made	1.0
seen	looked	1.0
vexed	got	1.0
instant	minute	1.0
move	gone	1.0

2nd:

word1	word2	Similarity
remembers	thought	1.0
book	scripture	1.0
vexed	bother	1.0
seemed	looking	1.0
part	broke	1.0
look	feel	1.0
think	thought	1.0
place	lay	1.0
kept	keep	1.0
bit	moment	1.0

3rd:

word1	word2	Similarity
dead	beat	1.0
got	make	1.0
sorry	dark	1.0
book	scripture	1.0
tell	said	1.0
looking	searched	1.0
beat	dead	1.0
moved	run	1.0
vexed	got	1.0
face	look	1.0