CPSC 422 - Assignment 4

Aiden Smith - 25037169 - p4y0b
Nicholas Pun - 27872167 - d0p1b

Question 1: Similarity Metrics in Ontologies

1. dirt#n#1 / earth#n#2
2. person#n#1 / human#n#1
3. WiFi#n#1 / internet#n#1
4. computer#n#2 / calculator#n#1
5. girl#n#1 / boy#n#1
6. eat#v#1 / drink#v#1
7. bird#n#1 / egg#n#1
8. mark#v#1 / marker#n#3
9. key#n#1 / unlock#v#2
10. run#n#7 / jog#v#3
11. rock#n#1 / paper#n#1
12. flower#n#1 / flour#n#1
13. bird#n#1 / egg#v#1
14. right#adj#5 / write#v#2
15. carpet#v#2 / launch#v#2

We then checked each of these with http://ws4jdemo.appspot.com/. Presented in spreadsheet form and sorted by our ranking, we got:

word 1	word 2	Our rank	lesk	jcn	path
dirt#n#1	earth#n#2	1	464	3.1775	0.5
person#n#1	human#n#1	2	342	0.1667	0.1
WiFi#n#1	internet#n#1	3	28	0	0.25
computer#n#2	calculator#n#1	4	668	0	1
girl#n#1	boy#n#1	5	81	0.0992	0.1667
eat#v#1	drink#v#1	6	69	0.2809	0.3333
bird#n#1	egg#n#1	7	156	0.0825	0.0909
mark#v#1	marker#n#3	8	18	-1	-1
key#n#1	unlock#v#2	9	6	-1	-1
run#n#7	jog#v#3	10	8	-1	-1
rock#n#1	paper#n#1	11	140	0.0725	0.1111
flower#n#1	flour#n#1	12	65	0.0576	0.0667
bird#n#1	egg#v#1	13	6	-1	-1
right#a#5	write#v#2	14	8	-1	-1
carpet#v#2	launch#v#2	15	12	0	0.1667

Now, sorting by each similarity measure. ‘lesk’:

word 1	word 2	Our rank	lesk	jcn	path
computer#n#2	calculator#n#1	4	668	0	1
dirt#n#1	earth#n#2	1	464	3.1775	0.5
person#n#1	human#n#1	2	342	0.1667	0.1
bird#n#1	egg#n#1	7	156	0.0825	0.0909
rock#n#1	paper#n#1	11	140	0.0725	0.1111
girl#n#1	boy#n#1	5	81	0.0992	0.1667
eat#v#1	drink#v#1	6	69	0.2809	0.3333
flower#n#1	flour#n#1	12	65	0.0576	0.0667
WiFi#n#1	internet#n#1	3	28	0	0.25
mark#v#1	marker#n#3	8	18	-1	-1
carpet#v#2	launch#v#2	15	12	0	0.1667
run#n#7	jog#v#3	10	8	-1	-1
right#a#5	write#v#2	14	8	-1	-1
key#n#1	unlock#v#2	9	6	-1	-1
bird#n#1	egg#v#1	13	6	-1	-1

Sorting by ‘jcn’:

word 1	word 2	Our rank	lesk	jcn	path
dirt#n#1	earth#n#2	1	464	3.1775	0.5
eat#v#1	drink#v#1	6	69	0.2809	0.3333
person#n#1	human#n#1	2	342	0.1667	0.1
girl#n#1	boy#n#1	5	81	0.0992	0.1667
bird#n#1	egg#n#1	7	156	0.0825	0.0909
rock#n#1	paper#n#1	11	140	0.0725	0.1111
flower#n#1	flour#n#1	12	65	0.0576	0.0667
computer#n#2	calculator#n#1	4	668	0	1
WiFi#n#1	internet#n#1	3	28	0	0.25
carpet#v#2	launch#v#2	15	12	0	0.1667
mark#v#1	marker#n#3	8	18	-1	-1
run#n#7	jog#v#3	10	8	-1	-1
right#a#5	write#v#2	14	8	-1	-1
key#n#1	unlock#v#2	9	6	-1	-1
bird#n#1	egg#v#1	13	6	-1	-1

And finally, sorting by ‘path’:

word 1	word 2	Our rank	lesk	jcn	path
computer#n#2	calculator#n#1	4	668	0	1
dirt#n#1	earth#n#2	1	464	3.1775	0.5
eat#v#1	drink#v#1	6	69	0.2809	0.3333
WiFi#n#1	internet#n#1	3	28	0	0.25
girl#n#1	boy#n#1	5	81	0.0992	0.1667
carpet#v#2	launch#v#2	15	12	0	0.1667
rock#n#1	paper#n#1	11	140	0.0725	0.1111
person#n#1	human#n#1	2	342	0.1667	0.1
bird#n#1	egg#n#1	7	156	0.0825	0.0909
flower#n#1	flour#n#1	12	65	0.0576	0.0667
mark#v#1	marker#n#3	8	18	-1	-1
run#n#7	jog#v#3	10	8	-1	-1
right#a#5	write#v#2	14	8	-1	-1
key#n#1	unlock#v#2	9	6	-1	-1
bird#n#1	egg#v#1	13	6	-1	-1

So yes, different similarity measures produce quite different results; all of these rankings are different from each other. In general it seems like we didn’t do a great job predicting the rankings, but ‘lesk’ seems to be closest to our ranking.

As for inconsistencies and errors:

We were surprised by the ‘-1’s. Turns out two of the three similarity measures we were investigating don’t support the comparison of words in different parts of speech. ‘lesk’, meanwhile, is valid for all of the word pairs.

We were surprised to find ‘computer#n#2’ and ‘calculator#n#1’ score very differently in some of the similarity measures. ‘path’ attempts to measure similarity by finding the shortest distance between nodes in the is-a structure of WordNet, this pair scores a 1 because in WordNet they literally link directly to each other (if you search up one you get a link straight to the other!) so the shortest path is minimal. The dictionary definitions are also essentially the same, so ‘lesk’ scores very high. However, they get a score of zero in ‘jcn’. We’re not quite sure why this is, but we theorize that it has to do with the relative rareness of these usages of these words and the fact that ‘jcn’ looks at the probability of each given a parent synset. These usages are quite rare and there may be very few instances in the entire dataset; so it may not be able to properly estimate similarity by this measure.

We can see the measurements for ‘WiFi#n#1’ and ‘internet#n#1’ were scored lower than our personal rankings. We believe this is due to the fact that we use ‘WiFi’ and ‘internet’ interchangeably in almost a slang-like fashion, despite their different meanings. However, the measurements are calculated based on their technical definitions and synsets. We sometimes say “the internet is down” when we refer to poor local WiFi connection. In reality, the internet being “down” would be a global catastrophe! Another example of words that we use interchangeably but have different meanings could be ‘sex#n#2’ and ‘gender#n#1’. We would rank the pair high, but it has a similar measure to ‘WiFi#n#1’ and ‘internet#n#1’.

Question 2: Probabilistic Context-Free Grammar

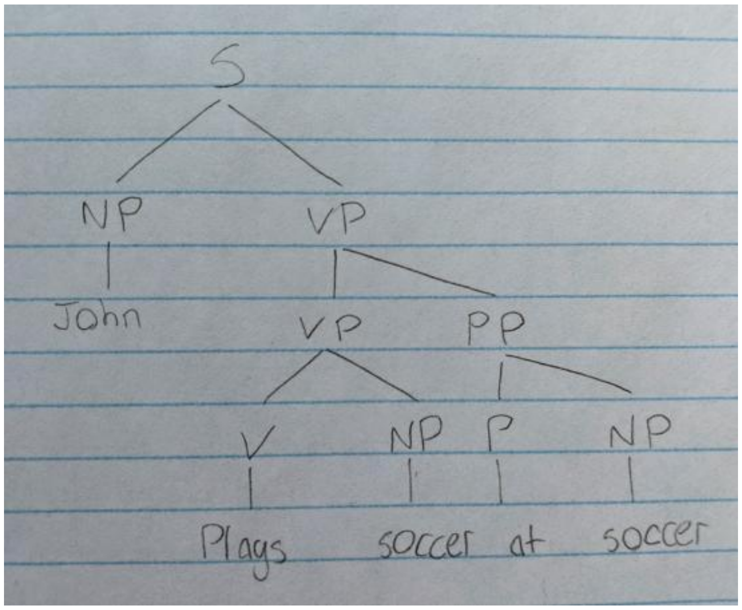
Part a:
We run the following command in the unix terminal:

```
>>>> perl ./mystery-code.pl ./pcfg-grammar.pl
```

We get the following results:

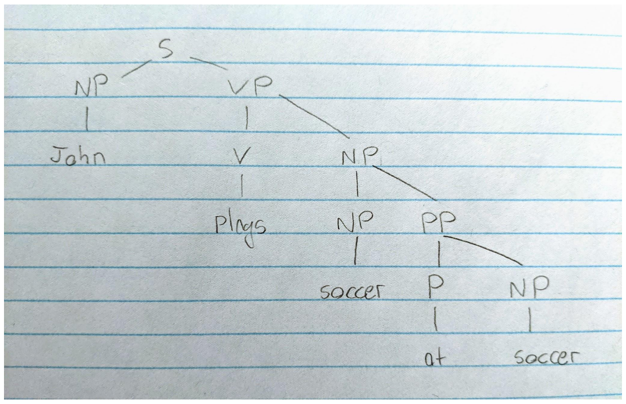
```
BEST DERIV: -5.54517744447956
BEST PARSE: (S (NP /John/) (VP (VP (V /plays/) (NP /soccer/)) (PP (P /at/) (NP /soccer/))))
```

Visualized as a tree, the results of this call to ‘mystery-code.pl’ looks like the following:



We’re sure this is the most probable parse because:

- The only mapping of S is to NP - VP, so that’s the most probable.
- There is no mapping of NP to the words at the start of the sentence *other* than ‘NP -> John’, so that must be the most probable choice.
- The ‘VP’ in the second level of the tree must map to VP - PP, if it mapped to V - NP it would map to the tree pictured below instead- which is valid, but has a lower probability. The subtree rooted at VP in the second level has a probability of 0.5*0.5*1*1*0.25*1*0.25 = 0.01625 in the above parsing. The following tree would be if the topmost VP mapped to V - NP. The subtree rooted at VP has a probability of 0.5*1*0.25*0.25*1*1*0.25 = 0.0078125. This is lower than the original probability and hence why it is not the result of the program.



- After that there are no other valid parsings than the one produced by the program, so it must be the most likely result. VP can only map to V - NP and PP can only map to P - NP. Since there are only four words left in the sentence, these mappings are the final ‘layer’ of the tree.

Part b:

To compute estimates for the probability of each rule, we counted the number of instances of each mapping for each non-terminal. The probability of a given mapping is then the number of its instances in the corpus of that mapping divided by the total number of instances of any mapping for the non-terminal. Our results are presented below:

```
$prob{"S -> NP VP"}    = 10/10          = 1;
$prob{"VP -> V NP"}    = 10/13          = 0.7692;
$prob{"VP -> VP PP"}   = 3/13           = 0.2308;
$prob{"PP -> P NP"}    = 4/4            = 1;
$prob{"NP -> NP PP"}   = 1/25           = 0.04;
$prob{"NP -> John"}    = 10/25          = 0.4;
$prob{"NP -> soccer"}  = 10/25          = 0.4;
$prob{"NP -> school"}  = 4/25           = 0.16;
$prob{"V -> plays"}    = 10/10          = 1;
$prob{"P -> at"}       = 4/4            = 1;
```

We then apply these new probabilities to the two following sentences by making a new file pcfg-grammar.pl file with the updated probabilities. Then, we ran the ‘mystery-script’ with the sentences “John plays soccer at school” and “John plays soccer” as the inputs. The program found the most likely parses to be the following. We then manually computed the probabilities:

John plays soccer at school
(S (NP /John/) (VP (VP (V /plays/) (NP /soccer/)) (PP (P /at/) (NP /school/))))
Probability = 1*0.4*(3/13)*(10/13)*1*0.4*1*1*0.16 = 0.004544

John plays soccer
(S (NP /John/) (VP (V /plays/) (NP /soccer/)))
Probability = 1*0.4*(10/13)*1*0.4 = 0.1231

Finally, we used the script to find the most likely parse for each of these sentences in the old grammar, and manually computed their probabilities in the old grammar:

John plays soccer at school
(S (NP /John/) (VP (VP (V /plays/) (NP /soccer/)) (PP (P /at/) (NP /school/))))
Probability = 0.25*0.5*1*0.25*1*1*0.25 = 0.0078125

John plays soccer
(S (NP /John/) (VP (V /plays/) (NP /soccer/)))
Probability = 1*0.25*0.5*1*0.25 = 0.03125

In both cases, most likely parse is the same for the old and new grammar.

For the first sentence, the most likely parse has a lower probability in our grammar because of the differing probabilities in the grammar. Just like the first example (‘John plays soccer at soccer’) this also has multiple possible parses, but it appears that the change in probabilities did not happen to affect which of the two parses was considered most likely by the algorithm. This makes sense because the other option for this parsing involves a mapping of NP -> NP PP, which was made *much less* likely in the new grammar than the old grammar. The probability of the parsing is just the product of the probabilities of all the rules used, so this greatly decreases the likelihood of the alternate parsing with NP -> NP PP and so the algorithm stuck with the same most-likely-parsing as before.

For the second sentence, the most likely parse has a higher probability in our grammar because the probability of each individual mapping was increased. Due to the structure of the sentence, it does not have the option to use any of the mappings whose probabilities were decreased. Thus the total probability of the parse was higher for our grammar.

Question 3: IBM Watson and what we covered in 322/422

Confidence Estimation:
A key idea in Watson's implementation is 'Pervasive confidence estimation' (Ferrucci et al.), where all of the many components of Watson's architecture, rather than committing to answers, produce 'theories' with estimated confidence which can then be combined. This seems reminiscent of a number of algorithms we looked at in 422, for instance PCFG's, where the final estimate of the probability of some possible world (like a possible parsing of a sentence in the case of PCFGs) is used to decide which possible world is most likely. Like Watson, we see variations on this idea everywhere in 322 and 422; where in Watson we can think of the confidence estimates as a kind of non-normalized probability of being correct.

Parsing
The Watson algorithm is based on the DeepQA architecture, and "DeepQA uses rule-based deep parsing and statistical classification methods both to recognize whether questions should be decomposed and to determine how best to break them up into subquestions" (Ferrucci et al.). In the Jeopardy case, Watson uses parsing to determine whether a question for a given category needs to be decomposed. If so, it breaks the question into the appropriate parts of speech so that Watson can be probabilistically better at processing an answer based on the parts of speech.

Shallow Parsing
Watson uses the parsing technique above to acquire accurate estimations for parses of the questions. However, the total number of questions parsed may be inadequate as deep parses may not be feasible. Therefore, Watson also uses shallow parsing to ensure that "the search-based system has better performance at 100 percent answered" (Ferrucci et al.). In some cases, shallow parsing may be less accurate, but it offers a wider coverage. Thus, using both parsing methods offers both high precision and accurate confidence estimation.

Entity and Relation Detection
When the game presents Watson with a passage, it is possible the answer is already contained within the passage, or at least related to something within the passage. In order to extract any possible answers from the passage, Watson may use entity detection (Ferrucci et al.). Relation detection is used to see if there are any relations in the parsed question. If there are, it can influence the generated answers.

Machine Learning
As Ferruci et Al. points out, maintaining the precision vs. resource-intensivity tradeoff is important. One step in doing this involves what they call 'soft filtering', where they examine a large number of possible answers using some number of 'lightweight analysis scores'. These are methods of scoring the possible answers which are relatively easy to calculate. These scores are then analyzed by their soft filtering model, and those (about 100) which pass some threshold are admitted to further and more computationally intensive examination to better determine which is best. The model and threshold used in the soft filtering step is determined

via machine learning methods, where the model is trained on some set of training data. Watson has also been described (Bringsjord, Govindarajulu) as a “meta-learning system”, where the DeepQA system learns over not just the base data, but the outputs of a large number of components which it has no assumptions about (those components could be hand-made, probabilistic reasoning, neural networks, etc).

Logical Form

Logical form is used for scoring. The score is compared with other scoring methods. Watson uses multiple scoring methods to accurately cover multiple bases. Logical form scoring is used when Watson deduces evidence for a given parsing and needs to know how this evidence affects the answer (Ferrucci et al.).

Temporal and Spatial Reasoning

Temporal and spatial reasoning are used by Watson when analyzing clues and possible answers. It eliminates possibilities based on time and location constraints. The answer may not be possible due to the time or location contradicting with those given in the clue (Ferrucci et al.).

Ontologies

Ontologies are used when Watson generates both shallow and deep parsings and needs to balance their semantics. It can result in the optimal combination of the parsings (Ferrucci et al.).

Ferrucci et al., (2010) *Building Watson: An Overview of the DeepQA Project*, AI Magazine 2010

Bringsjord, Govindarajulu, (2018) *Watson’s DeepQA Architecture,* Stanford Encyclopedia of Philosophy