Journal Report 1
8/29/19-9/5/19
Rahul Mittal
Computer Systems Research Lab
Period 2, White

Daily Log

Thursday August 29

Created IntelliJ Gradle project, copy paste example code from online, try to import Stanford parser libraries using Gradle, classes used in the example code aren't showing up, try downloading the Stanford parser stuff from the website and using the jars as libraries, classes still not showing up, decompile the jars and the classes aren't there, eventually somehow figure out that I should be importing 'stanford-corenlp' instead of 'stanford-parser'. The code compiles, run it, get an exception because some model classes aren't there even though I also imported those using Gradle.

Tuesday September 3

I used 'stanford-parser' instead of 'stanford-corenlp' for the models so I change that, then the code runs for a while and then gives an OutOfMemoryException. Build a jar using Gradle, try running it with 1GB of memory, OutOfMemoryException. 2GB, 4GB, same. Google it, find that the 'ner' (named entity recognition) annotator uses a lot of memory, disable it, 'coref' annotator depends on it, disable that. Now getting NPE from the 'kpb' annotator, still getting NPE even after switching from my sample proof to the sample text in the example. Disable that, now it works, right now only printing part-of-speech tags, google what the abbreviations mean. Decide to then look at what each annotator does, read that kpb extracts subject-relation-object triples from sentences, seems important, go back to Stanford website to try to figure out how I can get kpb to work.

Thursday September 5

Look at the source code for the kpb annotator, figure out that it implicitly depends on the 'ner' annotator, remember that my laptop has 16GB of ram, try adding back 'ner' annotator and everything else and running my jar with 8GB, works, yay! The kpb stuff isn't actually what I thought it was, but now I'm looking at constituency tree and dependency tree stuff which deal more with the sentence structure stuff I wanted to focus on. Looking at constituency tree right now, it's printed right now on a single line unlike dependency tree which is printed with newlines and spacing like trees are usually printed. Write a function to print the constituency tree in that format, then look at the constituency tree more closely and create a function to write some basic code of how I might approach dividing sentences into the necessary clauses and determining actions for each. Also look up combinatorial proofs online so that I can have more to work with, find a document with 5 additional combinatorial proofs.

Timeline

Date	Goal	Met
August 26	Not die before school starts	Yes, actually still alive even after two weeks of school
September 2	Run Stanford parser on my sample proof	No, didn't import the proper model files or run it with enough memory
September 9	Run Stanford parser on a couple of proofs	No, but I ran it on mine and looked at a couple of proofs
September 16	Run Stanford parser on more proofs, write a function trying to separate sentences into clauses with different actions	
September 23	Test my function with more proofs, make necessary revisions, write a basic function trying to assign actions to those clauses	

Reflection

Installing and running Stanford parser was a lot harder than I expected. I really did think I could successfully run Stanford parser on my text by the end of that first Thursday, which did not happen, and so I'm not where I expected to be having spent most of the last week trying to get all the annotators to work properly, though I don't think it's to any extent that I should worry about.

Here are the holy sacred Gradle imports that work:

```
dependencies {
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk8"
    testCompile group: 'junit', name: 'junit', version: '4.12'
    //compile group: 'edu.stanford.nlp', name: 'stanford-parser', version: '3.9.2'
    compile group: 'edu.stanford.nlp', name: 'stanford-corenlp', version: '3.9.2'
    compile group: 'edu.stanford.nlp', name: 'stanford-corenlp', version: '3.9.2',
}
```

You can see the commented out useless import as well.

Here's the current iteration of the sample proof that I'm using:

```
val SAMPLE_PROOF = ("Theorem: For integers b,u such that bu, "
        + "the amount of ways to place b labelled balls into u labelled urns "
        + "such that each urn contains at most one ball is u factorial divided by u
        + "\nProof: We can choose b urns which will contain a ball in bu ways, "
        + "and the balls can be matched up with the urns in b factorial ways, "
        + "making the total amount (u choose b) times b factorial = u factorial div
```

Looking at just the second sentence, we have three clauses, the first two of which state premises, and the third of which uses those premises to make a deduction which finishes the proof.

The whole constituency tree for the second sentence is very long, so this is just the third clause excluding most of the equations:

```
|   |   |   |   |   |   |   |   | (VP
|   |   |   |   |   |   |   |   |   | (VBG making)
|   |   |   |   |   |   |   |   |   | (NP
|   |   |   |   |   |   |   |   |   |   | (NP
|   |   |   |   |   |   |   |   |   |   |   | (DT the)
|   |   |   |   |   |   |   |   |   |   |   | (JJ total)
|   |   |   |   |   |   |   |   |   |   |   | (NN amount))
|   |   |   |   |   |   |   |   |   |   | (PRN
|   |   |   |   |   |   |   |   |   |   |   | (-LRB- -LRB-)
|   |   |   |   |   |   |   |   |   |   |   | (S
|   |   |   |   |   |   |   |   |   |   |   |   | (NP
|   |   |   |   |   |   |   |   |   |   |   |   |   | (NN u))
|   |   |   |   |   |   |   |   |   |   |   |   | (VP
|   |   |   |   |   |   |   |   |   |   |   |   |   | (VB choose)
|   |   |   |   |   |   |   |   |   |   |   |   |   | (NP
|   |   |   |   |   |   |   |   |   |   |   |   |   |   | (NN b))))
|   |   |   |   |   |   |   |   |   |   |   | (-RRB- -RRB-)))
|   |   |   |   |   |   |   |   |   |   | (FRAG
|   |   |   |   |   |   |   |   |   |   |   | (NP
|   |   |   |   |   |   |   |   |   |   |   | (NP
|   |   |   |   |   |   |   |   |   |   |   |   | (NNS times)
```

One thing I see here is that the ”u choose b” part, which was in parentheses, was recognized as its own sentence inside parentheses, and everything else starting with times was recognized as a fragment following the main ”making” part, which is inconvenient. This suggests that using parentheses to enclose formulas might be a good approach and also makes it clear that I should figure out how I’m going to enclose formulas before I write the function separating the sentences into clauses so that I know how to deal with them. I had previously thought that it would be a good idea to put formulas in quotes, which would be recognized and sort of separated by the parser, and I still think this is the better approach if it works the way I expect it to since I think parentheses are more commonly used in proofs for actual purposes than quotes, and the usage of formulas is much more similar to the use of quotes in normal text than it is to the use of parentheses, which are more for side comments or minor clarifications where formulas tend to be the main focus. I’ll have to start of with testing this parentheses and quotes stuff this coming week.