Maciej Medyk – COT6777 – Web Mining

Question 1 – [6.00pt]

What is Information Extraction? Please explain how to use information extraction to support information retrieval or search [1.00pt].

Information Extraction is a process that automatically extracts structured information from unstructured documents. The goal of Information Extraction is to allow computation to be done on previously unstructured data and to apply semantic recognition to the context and identify and extract relevant information from documents. Information retrieval is an automatic method for indexing large document collections and classifying them. Information Extraction also employees natural language processing which is a model for human language processing. Information extractions often focus on Named Entity Recognition where system is detecting people names, location names, and company names. It additionally focuses on Relationship Extractions where relationship is defined between two named entities like person works for organization.

What is a "Wrapper" in information extraction? Please provide an example of a Wrapper for information extraction [1.00pt].

Wrapper is a procedure that is designed to extract certain pieces of information from an unstructured document like HTML page and translate it to relational form. There are two approaches of wrapper generation which are wrapper induction and automated data extraction. Wrapper induction uses supervised learning to learn data extraction rules while automated data extraction which is using unsupervised pattern mining and in this approach extraction process follows templates and patterns. Example of a wrapper can be a regular expression for extracting emails which is as follows

```
' ([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})'
```

Another example of the wrapper can be a URL extraction procedure that would look for beginning boundary `<a href="` and ending boundary which is next occurrence of `">` giving us beginning and ending boundaries of the wrapper and allowing to extract the content from in between those two tags to obtain an URL.

What is "Boosted Wrapper Induction (BWI)"? Please explain how does BWI determine whether a particular input field is a target field (e.g., a name of a speaker) [1.00pt]?

Boosted Wrapper Induction is an enhanced Wrapper technique that improves performance of simple pattern matching of boundary wrappers through fore and aft detectors. Initially the procedure extracts weak wrapper then combining them to relationship phrases. The procedure is enhanced through semantic analysis of words and phrases that are predictable by association. For example if in the sentence we see phrase containing `"given by Dr. <CapitalizedWord>"` fore boundary would give a very high probability (0.95) that the following words would contain the person's name. If phrase would contain `"given by <CapitalizedWord>"` the fore boundary would still give somewhat high probability (0.65) for capitalize word to be a person's name, but not as high as the prior one as there is likelihood that capitalized word could be some other named entity like organization.

What is Named Entity Extraction? Please briefly explain two methods for named entity extraction [1.00pt].

Named Entity Extraction is a process of extracting semantic meaning from the text and identifying the category the entity belongs to. Name Entity can be done through identification of capitalized words as named entries and assigning the category like Person, Organization, or Location to the content of this named entity. This allows to find out objects within the text and through well-defined tools define relationship between those objects. Example of text after named recognition would be `"<PER> Dr. Hill</PER> teaches CAP6777 at <ORG>Florida Atlantic University</ORG>"`. There are two methods for extracting named entity out of the text. First one is called Knowledge Engineering which is very precise as roles are hard-coded and requires small amount of training data; however, it is not adaptable to changes and changes are very difficult over time. This system often uses regular expressions or context patterns. Second one is called Learning Systems which requires a lot of training data, but it does not require for programmers to develop grammar and rules and has high adaptability to changes. This system often uses Decision Trees and k-Nearest Neighbors methods.

Please use Regular Expression to define a pattern to extract phone numbers from a webpage (your pattern must be able to detect phone numbers in the following format (xxx) xxx-xxxx, or xxx.yyy.xxxx) [1.00pt].

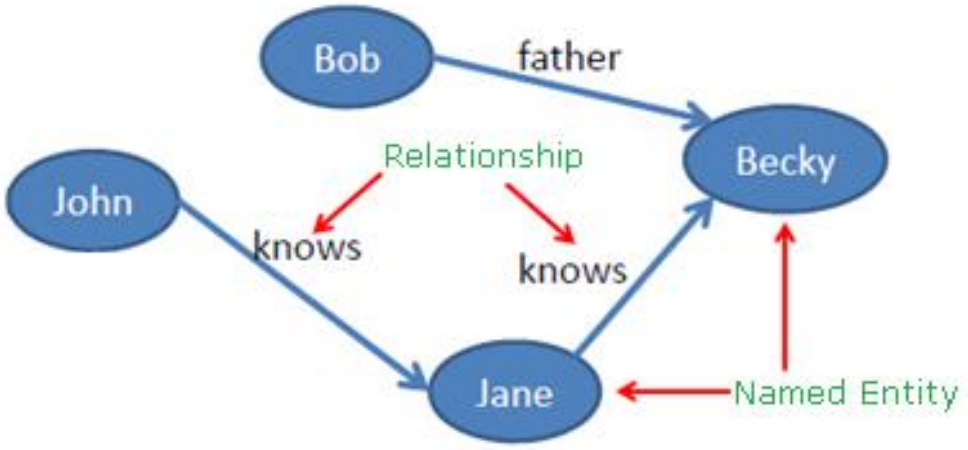
Regular expression that will capture those two formats is `'\(?\d{3}\)?[\s.]\d{3}[\.-]\d{4}'`

```
STRING = "Phone number that should show up " \
         "are (561) 888-9999 and 561.777.8888 " \
         "but not (561) 777 5555 or 561-555-3333"

result = re.findall('\(?\d{3}\)?[\s.]\d{3}[\.-]\d{4}', STRING)
print(result)
```

```
Run  Homework
C:\Python\Python\python3.exe
['(561) 888-9999', '561.777.8888']

Process finished with exit code 0
```

What is Named Entity Relation Extraction? What are the examples of relations? Please briefly explain two methods for Named Entity Relation Extraction [1.00pt].

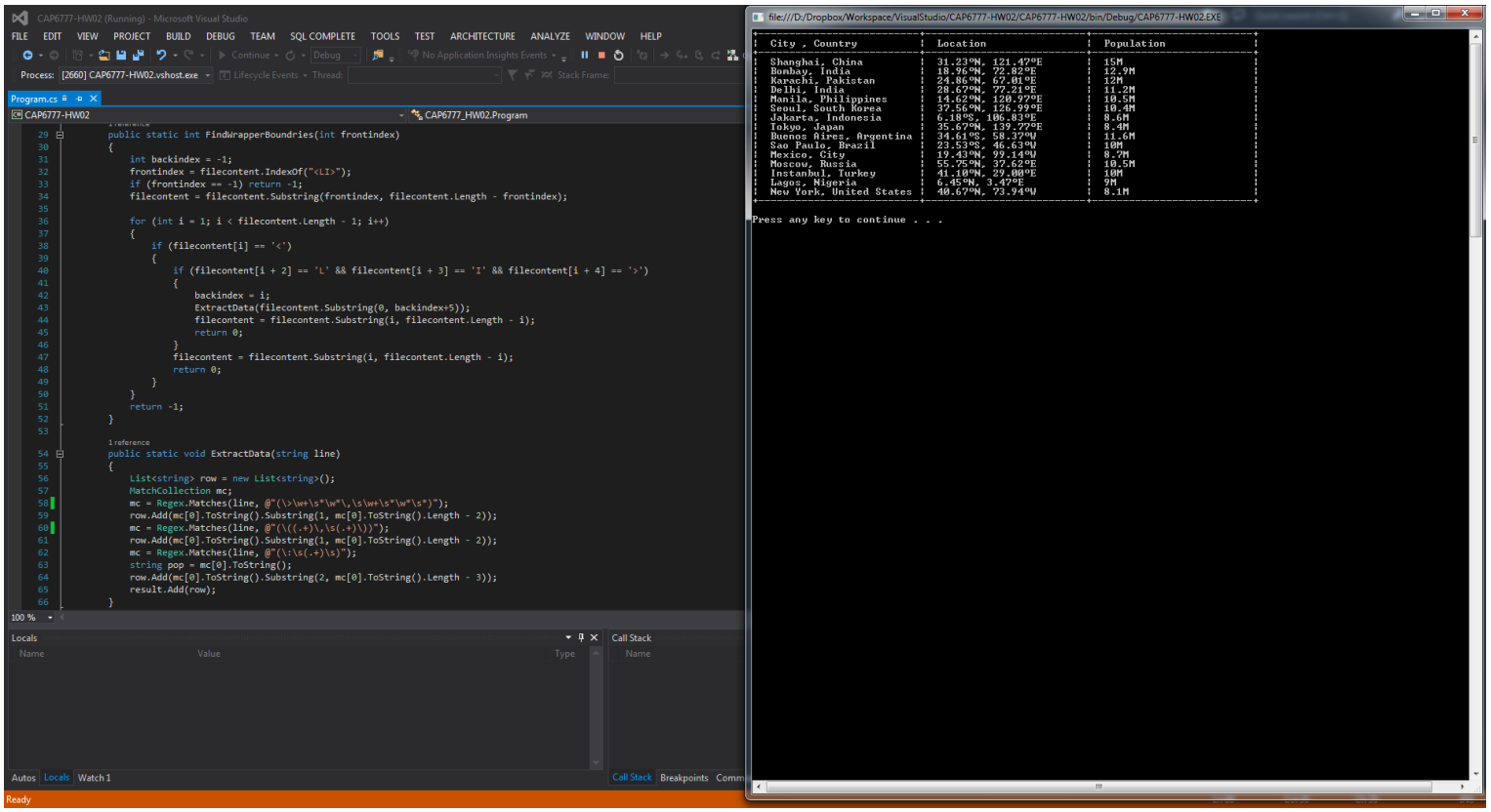
Named Entity Relation Extraction is focused on extracting Named Entity objects and establishing relationship between those objects. This is an approach in web ontology used to establish objects and relationships in Ontology Web Language OWL files. Another form of file that stores this type of relationship is Resource Description Framework RDF file. Those file formats store objects and establish relationships like IS-A between those objects. Examples of relationships like that would be phrases like "JOHN IS A FATHER" or "GINA KNOWS BECKY" or "LOCATION HAS ADDRESS". Those types of relationships are called relation triplets in which object has relation to other object. One of the methods of extracting Named Entity relations is Hearst's patterns for extracting IS-A relations using hyponyms to define object semantic similarities. Second method of extracting Named Entity relations is Richer Relations which are using rules where certain relationships are often true between specific entities like for example DRUG and DISEASE the typical relationship would be CURES or TREATS.



Question 2 – [3.00pt]

Please design a left-right wrapper to extract information as "City (including country)", "Coordination", and "Population". Please clearly mark the left-right wrapper for "City (including country)", "Coordination", and "Population", respectively. (You do NOT need to carry out any implementation to extract the information, but only need to define the wrapper for each field) [3.00pt].

Approach to extracting data is to find a <LI> tag which is followed closely by </LI> tag to establish fore and aft boundaries of the wrapper which will isolate a string for information extraction. Afterwards program will use regular expression to extract city and country via `"(\>\w+\s*\w*\,\s\w+\s*\w*\s*)"` location via `"(\((.+)\,\s(.+)\))"` and population via `"(\:\s(.+)\s)"`. Then those savings are saved to a data structure that will store the results.



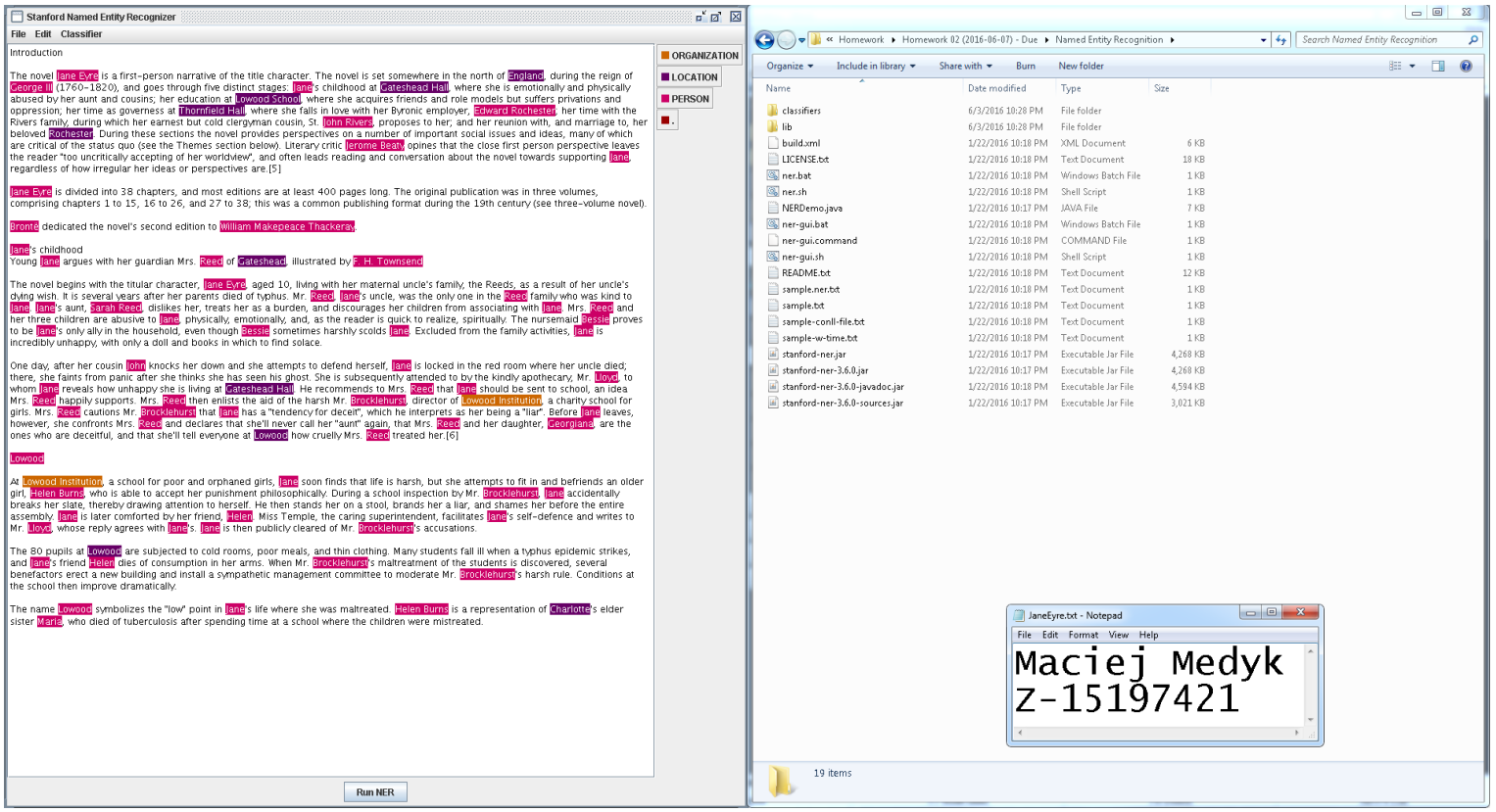

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Text.RegularExpressions;

namespace CAP6777_HW02
{
    class Program
    {
        public static List<List<string>> result = new List<List<string>>();
        public static string filecontent = "";
        static void Main(string[] args)
        {
            filecontent = ReadFile("Homework 02 - Figure2.txt");
            AnalyzePage();
            PrintResults();
        }

        public static void AnalyzePage()
        {
            int index = 0;
            do index = FindWrapperBoundries(index); while (index != -1);

        }
```

```
        public static int FindWrapperBoundries(int frontindex)
        {
            int backindex = -1;
            frontindex = filecontent.IndexOf("<LI>");
            if (frontindex == -1) return -1;
            filecontent = filecontent.Substring(frontindex, filecontent.Length - frontindex);

            for (int i = 1; i < filecontent.Length - 1; i++)
            {
                if (filecontent[i] == '<')
                {
                    if (filecontent[i + 2] == 'L' && filecontent[i + 3] == 'I' && filecontent[i + 4] == '>')
                    {
                        backindex = i;
                        ExtractData(filecontent.Substring(0, backindex+5));
                        filecontent = filecontent.Substring(i, filecontent.Length - i);
                        return 0;
                    }
                    filecontent = filecontent.Substring(i, filecontent.Length - i);
                    return 0;
                }
            }
            return -1;
        }

        public static void ExtractData(string line)
        {
            List<string> row = new List<string>();
            MatchCollection mc;
            mc = Regex.Matches(line, @"(\>\w+\s*\w*\,\s\w+\s*\w*\s*)");
            row.Add(mc[0].ToString().Substring(1, mc[0].ToString().Length - 2));
            mc = Regex.Matches(line, @"(\((.+)\,\s(.+)\))");
            row.Add(mc[0].ToString().Substring(1, mc[0].ToString().Length - 2));
            mc = Regex.Matches(line, @"(\:\s(.+)\s)");
            string pop = mc[0].ToString();
            row.Add(mc[0].ToString().Substring(2, mc[0].ToString().Length - 3));
            result.Add(row);
        }

        public static string ReadFile(string filename)
        {
            string result = "";
            var lines = File.ReadLines(filename);
            foreach (var line in lines)
            {
                result += line;
            }
            return result;
        }

        public static void PrintResults()
        {
            Console.Out.WriteLine("+--------------------------+--------------------------+--------------------------+");
            Console.Out.WriteLine("|  " + PadRight("City, Country", 25) + "|  "
                                  + PadRight("Location", 25) + "|  "
                                  + PadRight("Population", 25) + "|  ");
            Console.Out.WriteLine("+--------------------------+--------------------------+--------------------------+");
            for (int i = 0; i < result.Count; i++)
            {
                string output = "";
                for(int j = 0; j < result[0].Count; j++)
                {
                    output += PadRight(result[i][j], 25) + "|  ";
                }
                Console.Out.WriteLine("|  " + output);
            }
            Console.Out.WriteLine("+--------------------------+--------------------------+--------------------------+");
            Console.Write("\nPress any key to continue . . . ");
            Console.ReadKey(true);
        }

        public static string PadRight(string input, int length)
        {
            for(int i = 0; i < length; i++)
            {
                if (i > input.Length) input += " ";
            }
            return input;
        }
    }
}
```

File program is available at following link: https://www.dropbox.com/s/69j7rff0xo96ig0/CAP6777-HW02.zip?dl=0

Question 3 – [4.00pt]

Please download the package and report a screenshot of running the program on your computer. [1.00pt]



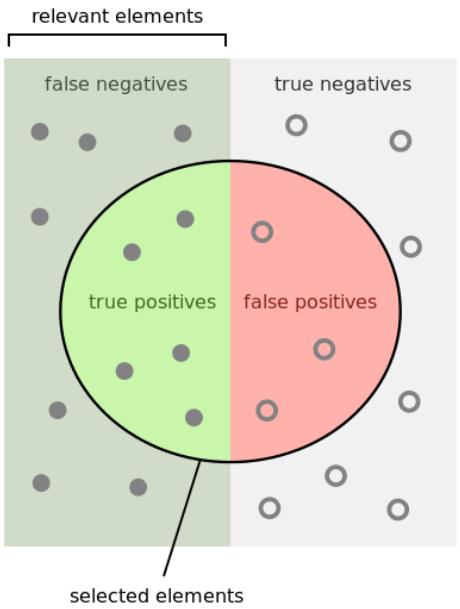
Please explain what is Precision, Recall, and F-Score [1.00pt]

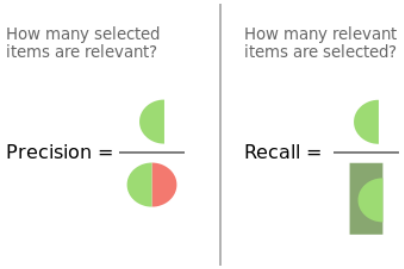
Precision, Recall and F-Score are all used as performance metrics. Precision is a measure that will try to evaluate percentage of correctly extracted fields so it is a fraction of the retrieved instances that are relevant. Recall is a fraction of relevant instances that are retrieved. Both precision and recall are therefore based on an understanding and measure of relevance. F-Score is a measure that combines precision and recall into harmonic mean where precision and recall are evenly weighted.

$$\text{Precision} = \frac{\text{Number of Correctly Extracted Fields (true positives)}}{\text{Number of Extracted Fields (all positives)}}$$

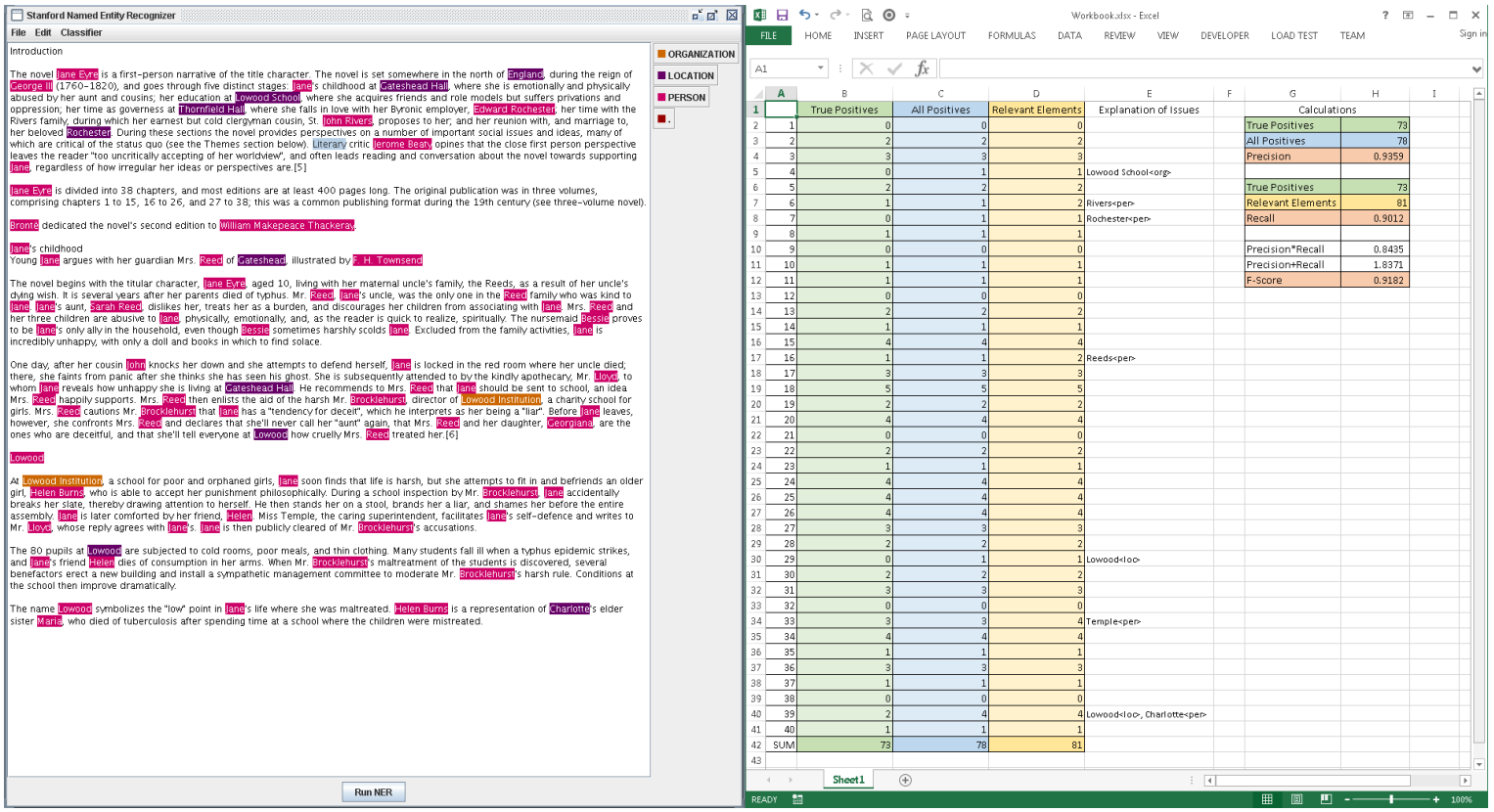
$$\text{Recall} = \frac{\text{Number of Correctly Extracted Fields (true positives)}}{\text{Number of Genuine Fields (relevant elements)}}$$

$$\text{F-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$





Please copy the text from Jane Eyre (WikiPedia: http://en.wikipedia.org/wiki/Jane_Eyre) [using content from "Introduction", "Jane's childhood", and "Lowood"], and use NER's "english.all.3class.distsim.crf.ser.gz" classifier to calculate the Precision, Recall, and the F-Score in identifying the 3 classes of named entities (Location, Person, and Organization) on the text collected from Jane Eyre. Please show a screenshot of the program, and also report the precision, recall and F-score in a table. [2.00pts]



Most of the words were properly captured but there were some false positives and some false negatives as table illustrates. Another table shows precision, recall, and F-score calculations based on the count from columns from image above. Overall there were 78 named entities identified and only 73 were positively identified. There were 81 relevant words in the entire texts so 3 missed identification by the program.

Line	True Positives	All Positives	Relevant Elements	Explanation of Issues
4	0	1	1	Lowood School should have been classified as organization rather than location
6	1	1	2	Rivers should have been classified as a person but it was not identified
7	0	1	1	Rochester should have been classified as a person rather than location
16	1	1	2	Reeds should have been classified as person but it was not identified
29	0	1	1	Lowood should have been classified as location rather than person
33	3	3	4	Temple should have been classified as person but it was not identified
39	2	4	4	Lowood should have been classified as location rather than person and Charlotte should be identified as person rather than location

Calculations

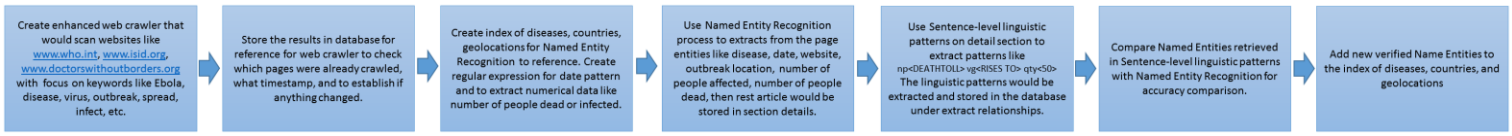
True Positives	73
All Positives	78
Precision	0.9359

True Positives	73
Relevant Elements	81
Recall	0.9012

Precision x Recall	0.8435
Precision + Recall	1.8371
F-Score	0.9182

Question 4 – [2.00pt]

Assume you were given a task to collect all Named Entities from Internet and find their relations to "Ebola" (e.g., the origin of Ebola, Ebola virus outbreak regions etc.). Please draw a flowchart (or diagrams) to elaborate the major steps of the project [1.00pt].



Please also explain the design of the experiments and the measurements to validate whether your method is working or not [1.00pt].

To see if the method is working the program would compare classifications done by Named Entity Recognition to Sentence-level linguistic pattern extraction to see if Named Entities have same classification like person, location, organization. Named Entity Recognition process would be self-learning process updated by verification mentioned earlier adding verified entities to the index and removing the misclassified entities. The process would be setup that previously recognized site would be rescanned after several days to see if the process would extract same Named Entities or if it would extract more.