Name: Ashish Singh Tomar

Email: ast2124@columbia.edu

UNI: ast2124

NLP

HW Assignment #1 – Stock Market Question Answering System

Programming Language: Java

Files Description

--src

--FileParser.java

--POSFileReader.java

--Questions.java

--Answer.java

--Makefile

-build.xml

--run.sh

--README.pdf

FileParser.java--This is the main file. It takes two files as arguments – POS file and Questions file. It writes the output to a file named **outputAnswer.txt** in the current directory. It first reads the POS file and then parses each Question from the Question file one by one and writes the output to the output file.

POSFileReader.java—This class is called by the main class FileParser to read the POS file and create an ArrayList of Strings containing each line as separate element of the arraylist. It also creates a free text ArrayList to output the line and line number in the output file along with the answer. The two arraylists,

one for POS Lines and the other one for free text lines both are used for finding the answers for the questions from the question file.

Questions.java— This file is called by the main class FileParser to parse the question from the Questions file using regular expressions and it helps in understanding what the question is actually about.

Answer.java— After the question and the noun from the Question file is identified the Answer class is call by the main class to extract the answer for the questions. This file contains all the regular expressions and code to extract the answer from the Arraylists created from the POS file.

Makefile—This file is used to compile the code using ant.

build.xml—This file is used by ant to compile the code.

run.sh—This shell script is used to run the code.

README.pdf—This file explains about the code and QA System.

QA System—

The main class is FileParser.java. The main method takes two arguments. The first argument needs to be POS file and the second needs to be the Questions file. It creates the output file --outputAnswer.txt. It first parses the POS File and creates two Arraylists of Strings for POS tagged lines and free text lines using the POSFileReader class. The lines from the POS file are stored in the Arraylists so that we don't have to read from the file again and again and matching can be done in a faster way from these ArrayLists.

It then reads the Questions from the Question file one by one and once it identifies the question using the Questions class it tries to find the answer and writes it into the output file **"outputAnswer.txt**". If the question is not identified then No Information available is written in the output file.

The different variations of verbs in the Questions class are accumulated using **WORDNET**.

The main method calls parseQuestion function to parse the questions one by one and to identify the Question and extract the noun and the question type. The function tries to match the question with the list of Questions from the Questions class. Once it identifies the question type and noun it searches for extracting an answer for the question asked using the Answer class.

The getAnswer method form the Answer class is called from the main class to retrieve the answers for the questions. It takes two arguments the nouns and verbs retrieved from the questions in the form of a matcher object and the question type. It iterates over the entire Arraylists containing the lines from the POS file and based on the Question type using a switch case statement it tries to match a specific pattern on each line to get the answers for the question.

For Example if the Question asked was –

Did the Nikkei Index gain or lost?

Then first it would extract the noun as Nikkei Index and identify the question type of this question. Then it would identify the appropriate regular expression pattern to look for the answers.

Then it will try to match the appropriate regular expression on each line and extract the verbs.

From line 3, it would extract that Nikkei index fell but will print the output as LOST as asked in the Question.

Tokyo's Nikkei Index of 225 issues, which fell 136.28 points Wednesday, closed at 34795.05, down 445.02. (line 3)

It will still keep looking for matching more patterns.

Similarly from line 5, it would extract that Nikkei index rose but will print the output as GAIN as asked in the Question.

In the first hour of trading in Tokyo Friday, the Nikkei Index rose 145.96 points to 34941.01. (line 5)

So the output in the output file would be like this –

Q: Did the Nikkei Index gain or lost?

A 1: LOST.

Source 1: Tokyo's Nikkei Index of 225 issues, which fell 136.28 points Wednesday, closed at 34795.05, down 445.02. (line 3)

A 2: GAIN.

Source 2: In the first hour of trading in Tokyo Friday, the Nikkei Index rose 145.96 points to 34941.01. (line 5)

It also tries to match also the paraphrases for the nouns. For Example if asked about Tokyo's Nikkei Index it will get results for Tokyo's Nikkei Index, Tokyo's Index, Nikkei Index and Tokyo's Nikkei. But this will be done only if they exist exclusively in a given phrase. If the answer is found it's written into the output file. If no answer is found then it writes No Information Available in the output file. It does regular expression matches on either free text or POS text or both depending on the question type. In

the case of extracting company names form the text it also uses Distance heuristic to not select the answer if the distance between the proper noun and rise/fall variants of the verb is more than 10 words.

The output can be found in **outputAnswer.txt** file in the directory where the code exists.

Special features—

Some of the special features are –

1) Distance heuristic for not selecting the answer if the distance between the proper noun and rise/fall variants of the verb is more than 10 words.
2) Extensive list of verbs manually selected from Wordnet.
3) Fast and memory efficient execution.
4) Paraphrasing and using tight regular expressions while using the paraphrases to exclude the false positives.
5) Design is Modular and clean code.

How To Run

1) To compile the code-

$make

2) Change the mode of shell script-

$chmod +x run.sh

3) Run the code with two arguments –

$ ./run.sh posfile.txt question.txt

The file outputAnswer.txt will be created in the current directory and will contain the answers.