Course Work Report

I use three external resources to help label the text, including a list of person names(5163 entries), a list of country names(241 entries) and a list of city names(13486 entries). For the reason of time complexity, I compare at most two continuous words from the original text to the entries of external resources(except organization entities, for them I use another function to check). When testing, I also find that the program can hardly recognize persons' name, so I use the edit distance of words to help expand the searching. In the program, I set edit distance < 2, for example, word like "Verdi", can not be found at external resource, but when adding a word "e", it become "Verdie", which exists in the resource. The edit distance is 1 so that "Verdi" can be accepted in the function. The distance algorithm I used was proposed by Lloyd Allison. The algorithm can be found in his paper *Lazy Dynamic-Programming can be Eager.*

The details of the program are described below:

1. The program starts at "main" function. First it reads the sentences from "untagged.txt" and "tagged.txt". It also loads the content from "countries.txt","cities.txt" and "name.txt" for further use.

2. The program then break the content of "untagged.txt" into lines and for each line in the list, the program use the function nameEntityRecog :: [String] -> [String] -> [String] -> [String] -> [String] -> [String] to label the text and return the result. The nameEntityRecog has 5 parameters, when calling this function, the program sends the lines of untagged.txt, countries, cities, name, [] to the function (at first the result is [], and it will be added with strings during recursion).

3. In the nameEntityRecog function, there are several functions to help label the text. The function nameEntityRecog will first feed the parameters to function getOrgEntities :: [String] -> [String] -> [String]. This function will label the organization entities by checking if the first letter of word is uppercase or if the word is "of" or "the", if yes, the function will store the word and check next word in the list. If no, it will return the current result. Finally it will return a list of sentences with their organisation entities been labelled.

4. Next, the nameEntityRecog function will go into getTwoWordsEntities :: [String] -> [String] -> [String] -> [String] -> [String] -> [String]. This function will iterate every continuous 2 words in the list and then compare it with the external resources to see if they are LOCATION and PERSON. It also check if the 2 words is date element or time element or money element by comparing every word to a predefined date/time words list and money words list. It also takes into consider the situations when punctuation is the last element of the word, like "….last month." or "…$200 million?" or "Abby's". This function will return a list of sentences with their two-words entities been labelled.

5. Then, the nameEntityRecog function will go into getOneWordEntities :: [String] -> [String] -> [String] -> [String] -> [String] -> [String]. This function will check every word in the list to see if it belongs to any categories. The algorithm is similar to the getTwoWordsEntities. This function will return a list of sentences with their one-word entities been labelled.

6. Finally, the nameEntityRecog will return the result list to main function. The result will be written to “result.txt”.

7. For the convince of calculating statistics, the program will extract only the labelled entities by using getlbdEty :: [String] -> [String] -> [String]. Both labelled entities of “tagged.txt” and “result.txt” will be extracted to 2 different lists.

8. The program will then compare every labelled entity from my result to the labelled entities from “tagged.txt” to get the correct labelled entities. The function is getCorrectLabel :: [String]->[String]->Float->Float. After that, the program can then calculate the recall, precision and F1 measure respectively.

Because the time complexity of Lloyd Allison”s distance algorithm is O(length a * (1 + dist a b)), so it may take some time (several seconds) to compute the result if the original file is not very small.

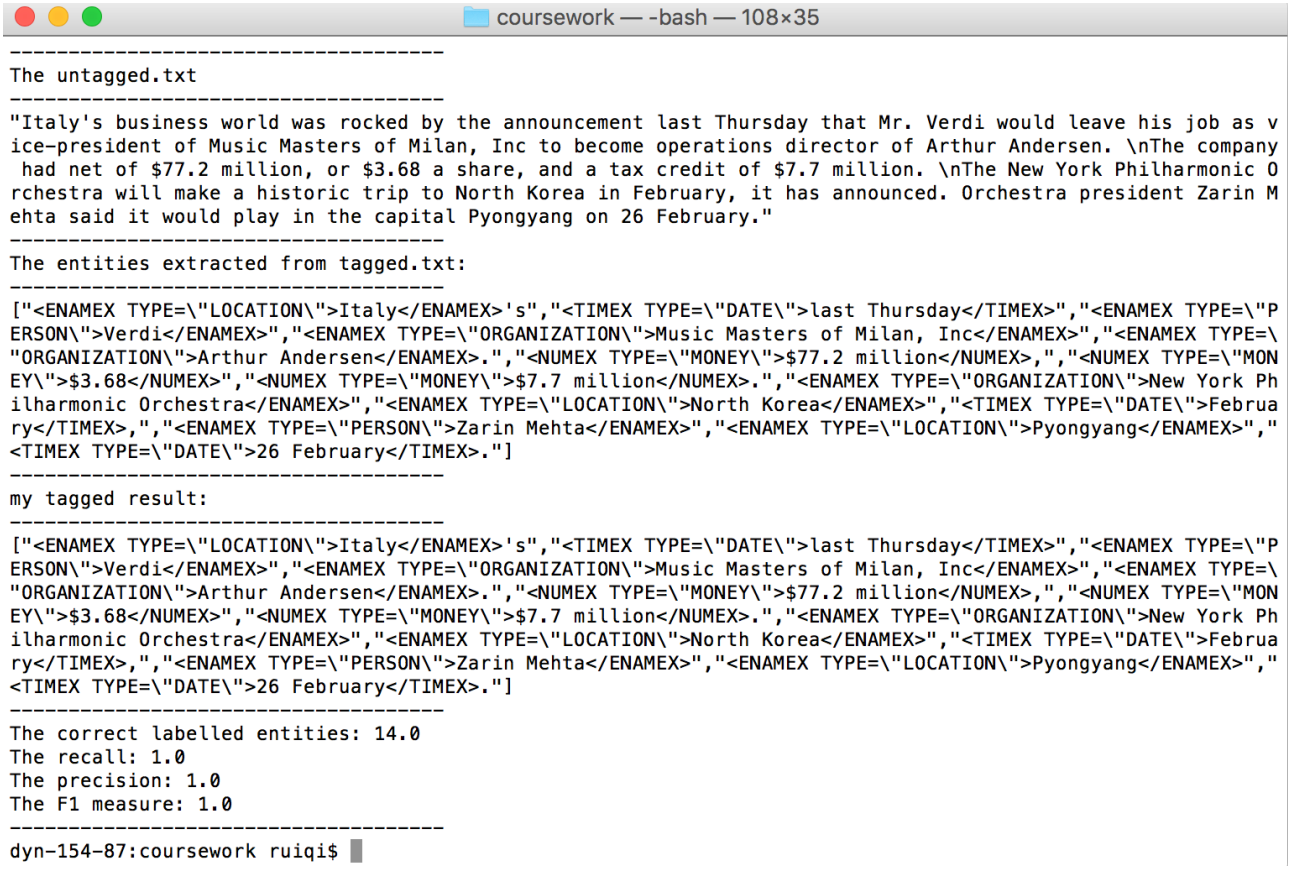

Fig. The output of the program