Homework2 report

20180169 김진석

1. Underlying intuition of my algorithm

In my case, I used very different algorithms for snippet-context and page-context. In snippet-context, I used total 3 filters to improve accuracy of the result. The three filters are counter_filter, names_corpus_filter, and offset_filter. On the other hand, in page-context, I thought that statistical method is better than using local optimization, so tried to look at whole raw text, and find patterns.

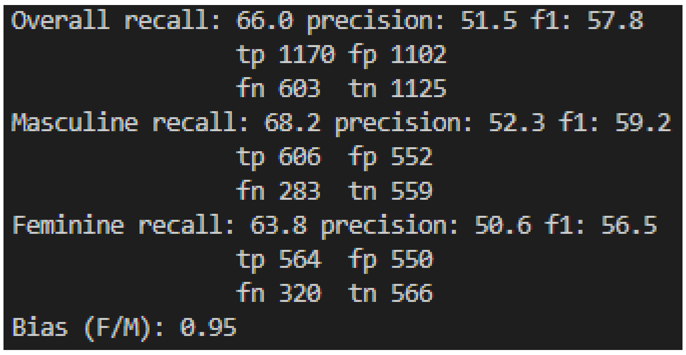
For snippet-context, counter_filter, I used basic counting method. First, I tokenized the given text, and tagged each word with part-of-speech tags. Then, using named entity recognition using ne_chunk, I found chunks which has "PERSON" label. If a particular chunk is part of the given instances, then increase the count of a particular instance by 1. By checking all the chunks in the given text, we can get final count_a and count_b variable. If both count_a and count_b is 0, then return ("FALSE", "FALSE"), which is obvious. Else, If one of the count is bigger than the other, bigger one should return should return "TRUE" and smaller one return "FALSE". Since this is just first filter of the three, I thought this simple assumption could make sense.

For names_corpus_filter, I used names corpus in NLTK. Firstly, I got the result from the first filter, so that I can refine it. Then, I categorized the possible pronouns by gender. For example, 'he', 'his' and 'him' is in male category, but 'she', 'her' is in female category. Then, I got male names and females names in NLTK corpus, to decide the gender of a particular instance of name. I only extracted the first name of the instances, since the given names corpus is for the first name. I realized that there are several instances of names that are not in the corpus. Therefore, I decided to make algorithm that if the first name of the instance is in the opposite gender group of the given pronoun, and also is not in the same gender group of the pronoun, then I can exclude the particular case and return false. For example, if the given pronoun is "her", and the first name of instance 'a' is in male category, and not in female category, I can assume that the pronoun has very low possibility that it refers to the instance, so I can return false.

For offset_filter, I thought that if the given pronoun and the instance a or b is very close, which means that the difference of the offset is less than 30 letters, then I can assume that the instance refers to the given pronoun. Therefore, I can return true for the particular instance.

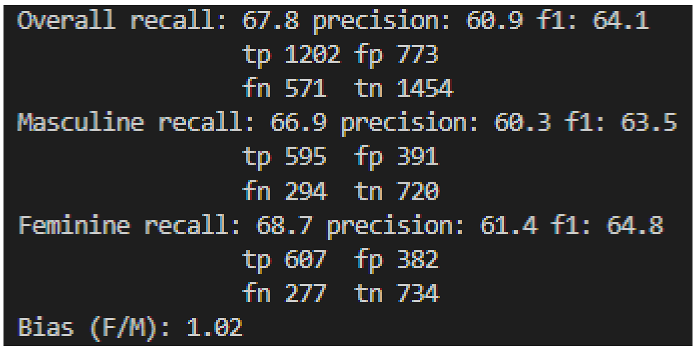
In page-context, I chose to count how many times the instances appeared in the total Wikipedia raw text. I assumed that the more the instance appeared in the text, the more possibility that the instance would be the referred to the given pronoun. So first, I opened the URL and parsed to text with module BeautifulSoup. Then I tokenized the raw text into sentences, and counted the number of instances appeared in raw text. If both appeared 0 times, return ("FALSE", "FALSE"), if one is bigger than the other, than return bigger one as "TRUE" and smaller one as "FALSE". IF there is any problem in getting information from the URL, just get the result from snippet-context.

2. Result & Performace table



```
Overall recall: 66.0 precision: 51.5 f1: 57.8
                tp 1170 fp 1102
                fn 603  tn 1125
Masculine recall: 68.2 precision: 52.3 f1: 59.2
                tp 606  fp 552
                fn 283  tn 559
Feminine recall: 63.8 precision: 50.6 f1: 56.5
                tp 564  fp 550
                fn 320  tn 566
Bias (F/M): 0.95
```

<figure1. Snippet-context performance table>



```
Overall recall: 67.8 precision: 60.9 f1: 64.1
                tp 1202 fp 773
                fn 571  tn 1454
Masculine recall: 66.9 precision: 60.3 f1: 63.5
                tp 595  fp 391
                fn 294  tn 720
Feminine recall: 68.7 precision: 61.4 f1: 64.8
                tp 607  fp 382
                fn 277  tn 734
Bias (F/M): 1.02
```

<figure2. Page-context performance table>

As you can see in the above table, recall and precision are higher for page-context than snippet-context, which is as expected. However, for masculine recall, snippet is better than page-context, which shows that page-context is not always better in my algorithm.

3. How to improve my algorithm

I think the way to improve snippet-case algorithm is to find deeper relations between entities. The sad point of my algorithm is that I could only filter when the distance of instance and the pronoun is very close. Not only using simple offset difference but also make clear relation by considering many other noun phrases can help the algorithm to exclude many variances. Therefore, we can get "TRUE" result, even if the distance between two words are long enough.

Second way to improve my snippet-case algorithm is that I used three filters, counter_filter, names_corpus_filter, and offset in a row, which obviously weakened the influence of first filter. At first, I though that I could give a coefficient for each filter, and make score system for the algorithm. However, I could not get proper ratio for the coefficients, because of the limitation of time, and felt that using filters in a row is the optimal option for me. However, if I have enough time to test the cases, maybe I can get the best coefficients, making the algorithm works better.

The way to improve my page-context algorithm is that it took way too long time to actually get .tsv result. By using multi-threading or other accelerating modules, maybe I can get better, without HTTP Error, and faster results.