Project 4 – Corrections

Group 1

11/29/2018

For this project, we used the groundtruth papers as training data, and seeked to correct as many spelling errors in the tesseract data as possible. First, we flagged any spelling differences between groundtruth (the correct spelling) and tesseract (the incorrect spelling) as a misspelled word. Then, as the "Probability Scoring for Spelling Correction" paper stated, we used deletion, insertion, and substitution techniques to look for any possible corrections for these words. Deletion would see if deleting an unnecessary character from the word would lead to a correctly spelled word, insertion would see if inserting an additional character into the word would lead to a correctly spelled word, and substitution would see if replacing a character in the word with a different character would lead to a correctly spelled word. Therefore, this method is limited to words that are only off by one character.

Once there is a list of potential spelling corrections as a result of these three techniques, we picked the most likely word, taking into account the word frequency (how often did that potential word occur in the groundtruth papers), and bigram frequency (how often did that potential word occur next to the left and right neighbor words).

The end goal was to calculate the accuracy of this method – to see how many of these correction attempts matched the corresponding groundtruth word (the correct spelling). Since the method only fixes words that are off by one character, we performed the algorithm on all misspelled words, and on misspelled words that are only off by one character, and we calculated the accuracy for each situation.

After reading in the data, we first look at the word frequencies and the bigram frequencies.

```
##   tokens     n
## 1    the 42153
## 2    and 19187
## 3    for  7318
## 4   that  5585
## 5    cma  5239
## 6   with  4435
```

```
##   word1 word2 freq
## 1    of   the 5577
## 2    in   the 2809
## 3    to   the 2292
## 4    on   the 1774
## 5   and   the 1388
## 6  will    be 1321
```

Then, we want to look for differences between tesseract and groundtruth spelling. Since the number of words in each are not exactly the same and the word indices do not match up, we can't just compare spelling based on a given word index. The idea is to look for words near that word index. For example, if we want to see if word number twelve in tesseract is spelled correctly, we are not going to just look at word number twelve in groundtruth because it may not match up, but we are going to look at words within a given window of word number twelve. If the same word is found with the same spelling in that window, then the word is spelled

correctly. If a word is found with a similar spelling in that window, then the word is flagged as being misspelled. If no similar word is found, then “No Match Found” is returned; this could be because the word is so badly misspelled that it is too far off from the match, or the word itself was a mistake and was taken out in groundtruth.

The function get_mistake_list takes in a paper number (there are multiple papers in the dataset, but due to run time constraints we do not use all of them, so the paper number specifies which paper to use), and a range. This range is determined by how big the search window will be. We decided that 6 was a good number: so the search window would be 6 words before and after the word index.

The function takes each word and puts it in one of four categories:
1. Spelled correctly (exact spelling match was found)
2. A match was found with different spelling (word is misspelled)
3. No match was found
4. Index was out of range (in some cases the tesseract data is larger than groundtruth, so the last few indices are too far from the groundtruth indices).

The function chooses only the words in scenario number two, and prints the pair of words along with the word location in tesseract.

As I mentioned before, we are calculating two different accuracies (words that are only one character off, and words more than one character off). So one dataframe shows all pairs of words that only differ by one character, and the other dataframe shows all pairs of words that differ by at most 30% of the number of characters in the word.

```
##    wordlist_tesseract wordlist_groundtruth word_index
## 10                423                 1423         10
## 46                 cm                    c         46
## 47          redactidn            redaction         47
## 52         1nvestment           investment         52
## 58        1ntolerable          intolerable         58
## 86             perlod               period         86
```

```
##    wordlist_tesseract wordlist_groundtruth word_index
## 8              exhlblt               exhibit          8
## 15             exhlblt               exhibit         15
## 20             exhlblt               exhibit         20
## 22             exhlblt               exhibit         22
## 24             exhlblt               exhibit         24
## 26             exhlblt               exhibit         26
```

Now that we have all the words that need to be corrected, we call the function print_corrections, which will apply deletion, insertion, and substitution. First it takes the dataframe calculated previously. The deletion, insertion, and substitution functions take the misspelled word, and the two neighboring words (which are found using the word_index in the dataframe calculated previously), and finds all existing words using that technique. Then each word is “scored” by the average of its word frequency and its bigram frequencies (using the left and right neighbors). The word with the highest score is chosen. Therefore, each word has at most one potential correction for each technique.

Again, this is calculated on the words off by one character, and the words off by more than one character. For each dataframe, “wordlist_tesseract” is the misspelled words; “deletion_correction”, “insertion_mean”, and “substitution_correction” are the “highest scoring” corrections; and “deletion_mean”, “insertion_mean”, and “substitution_mean” are the “scores” for that correction.

```
##    wordlist_tesseract deletion_correction deletion_mean
## 17            exhlblt      No Match Found            NA
## 18            exhlblt      No Match Found            NA
## 19             perlod      No Match Found            NA
## 20              19797      No Match Found            NA
## 21       preparatlons      No Match Found            NA
## 22             pollcy      No Match Found            NA
## 23          commlttee      No Match Found            NA
## 24             revlew      No Match Found            NA
## 25               thls                 ths             3
## 26        approprlate      No Match Found            NA
## 27     ecommendatlons      No Match Found            NA
##    insertion_correction insertion_mean substitution_correction
## 17       No Match Found             NA          No Match Found
## 18       No Match Found             NA          No Match Found
## 19       No Match Found             NA                  period
## 20       No Match Found             NA          No Match Found
## 21       No Match Found             NA            preparations
## 22       No Match Found             NA                  policy
## 23       No Match Found             NA               commlttae
## 24       No Match Found             NA                  review
## 25       No Match Found             NA                    thos
## 26       No Match Found             NA             appropriate
## 27       No Match Found             NA          No Match Found
##    substitution_mean
## 17                NA
## 18                NA
## 19        65.3333333
## 20                NA
## 21         9.6666667
## 22       311.0000000
## 23         0.3333333
## 24       199.6666667
## 25         0.3333333
## 26       102.6666667
## 27                NA
```

```
##    wordlist_tesseract deletion_correction deletion_mean
## 11             revlew      No Match Found            NA
## 12               thls                 ths             3
## 13        approprlate      No Match Found            NA
## 14            meetlng      No Match Found            NA
## 15           adlpates      No Match Found            NA
## 16            arsenlc      No Match Found            NA
## 17            ethylne      No Match Found            NA
## 18    polychlorlnated      No Match Found            NA
## 19          blphenyls      No Match Found            NA
## 20           knowledg      No Match Found            NA
##    insertion_correction insertion_mean substitution_correction
## 11       No Match Found             NA                  review
## 12       No Match Found             NA                    thos
## 13       No Match Found             NA             appropriate
## 14       No Match Found             NA                 meeting
## 15       No Match Found             NA                adipates
## 16       No Match Found             NA                 arsenic
## 17             ethyline      0.6666667          No Match Found
## 18       No Match Found             NA         polychlorinated
## 19       No Match Found             NA               biphenyls
## 20            knowledge     13.0000000          No Match Found
##    substitution_mean
## 11       199.6666667
## 12         0.3333333
## 13       102.6666667
## 14       297.6666667
## 15         1.6666667
## 16        25.3333333
## 17                NA
## 18        22.0000000
## 19        21.3333333
## 20                NA
```

Next, the choose_best_corrections function is called, which takes each word and decides whether the deletion correction, insertion correction, or substitution correction should be used, based on whichever one has the highest “score”. The function takes in the previous dataframe, and returns a vector of the chosen corrections.

```
##  [1] "ethyline"        "polychlorinated" "biphenyls"
##  [4] "knowledge"       "production"      "agenda"
##  [7] "wlch"            "particularly"    "facing"
## [10] "issues"          "wlch"
```

```
##  [1] "No Match Found" "redaction"      "investment"     "intolerable"
##  [5] "No Match Found" "No Match Found" "No Match Found" "No Match Found"
##  [9] "period"         "No Match Found"
```

The get_finalized_df function takes the dataframe that showed the misspelled pairings, and adds a column of the corrections calculated above.

```
##    wordlist_tesseract wordlist_groundtruth word_index final_corrected
## 10                423                 1423         10  No Match Found
## 46                 cm                    c         46             bcm
## 47          redactidn            redaction         47       redaction
## 52         1nvestment           investment         52      investment
## 58        1ntolerable          intolerable         58     intolerable
## 86              perlod               period         86          period
```

```
##    wordlist_tesseract wordlist_groundtruth word_index final_corrected
## 8              exhlblt               exhibit          8  No Match Found
## 47          redactidn            redaction         47       redaction
## 52         1nvestment           investment         52      investment
## 58        1ntolerable          intolerable         58     intolerable
## 86              perlod               period         86          period
## 91               19797                 1977         91  No Match Found
```

Then we calculate the accuracy for each dataframe. For words that are only off by one character: 72.73%
For words that are off by more than one character: 64.74%

There were 3 main limitations we came across when working on this project:
1. The deletion, insertion, and substitution techniques only work for words off by one character. The method performed approximately 8% worse when it was applied to all mistakes, including the ones off by more than one character.
2. The groundtruth data was not 100% correct. It was mostly correct, but some words were misspelled there. This leads to an incorrect training dataset, and accuracy scores that are not 100% reliable.
3. When searching for matches, if a one-letter word was not an exact match, we immediately returned “No Match Found”, otherwise any other one-letter word in the search window would be chosen as a match because it is only one character different. This would miss any typos such as “John O Smith” vs. “John 0 Smith”, which would have needed to be corrected, but otherwise, it was pairing one-character words together that had nothing to do with each other.