Lecture 2 Project: Document Distance

Yanuar Heru Prakosa

31-05-2021

The Document Distance

The assignment from the Lecture 2 in 6-006 is related to one called document distance. Thus I need to find out first what is exactly document distance is. Form this MIT discussion webpage, I learn that document distance will be measured using vector equations. Let D be a text, then a word is a consecutive of alphanumeric characters. We will not distinguish upper case and lower case letters, but we use non alphanumeric characters as delimiter between words. For example the word "can't" consist of 2 words: can and t.

Okay now we go to the formulation: The word word frequency distribution of a document D ia a mapping from words w to their frequency count, denoted as $D(w)$. We can view the frequency distribution D as vector, with one component per possible word. Each component will be a non-negative integer ≥ 0.

The norm of this vector is defined by:

$$N(D) = \sqrt{D \cdot D} = \sqrt{\sum_{w} D(w)^2}$$

Alright this calculation still does not make any sense right now. But let's move on, I need to get to how to calculate document distance first. Now when we have two documents to be compared to each other, let's name them D and D'. The inner product between D and D' is defined as:

$$D \cdot D' = \sum_{w} D(w)D'(w)$$

So basically this is the sum of products on all word frequencies in two documents. Thus, if a word exist 1000 times in one document but never existed in other document the inner product of that part is 0!

Okay, since our objective is to define distance which is defined here as angle between two documents we need to go back to definition of vector dot products:

$$D \cdot D' = N(D)N(D')cos\theta$$

As we looking for an angle we are focusing on θ right? Therefore:

$$cos\theta = \frac{D \cdot D'}{N(D)N(D')}$$

$$\theta = arccos\left(\frac{D \cdot D'}{N(D)N(D')}\right)$$

Deriving from other equations above we will get:

$$\theta = arccos\left(\frac{\sum_w D(w)D'(w)}{\sqrt{\sum_w D(w)^2}\sqrt{\sum_w D'(w)^2}}\right)$$

From those formula we can infer that if both Documents are the same then the $angle(D, D') = \theta = 0$. On the other hand if both documents are completely different in the sense there are no similarity in terms of word used in it then the $angle(D, D') = \theta = \frac{\pi}{2}$. Please note we are using radians in this matter.

Pre Example

To make it clearer I think a simple practical example is mandatory. Let us have two sentences that we will use as documents. One is "To be or not to be" and the other is "Doubt truth to be a liar". We will calculate the distance between these two documents.

Now we already list all unique set of words and it can be seen in this table plus its frequencies:

word	D1(w)	D2(w)
to	2	1
be	2	1
or	1	0
not	1	0
doubt	0	1
truth	0	1
a	0	1
liar	0	1

Now let's get back to the formula:

$$\theta = arccos\left(\frac{\sum_w D(w)D'(w)}{\sqrt{\sum_w D(w)^2}\sqrt{\sum_w D'(w)^2}}\right)$$

we begin with the denominator part first.

To calculate the denominator we need to find $\sum_w D(w)^2$ and $\sum_w D'(w)^2$ first. Here is the results of the calculation:

word	D1(w)	D2(w)	D1(w)^2	D2(w)^2
to	2	1	4	1
be	2	1	4	1
or	1	0	1	0
not	1	0	1	0
doubt	0	1	0	1
truth	0	1	0	1
a	0	1	0	1
liar	0	1	0	1
		sums	10	6

As you can see the sums already being included in the table above. We can use it as our denominators, but remember those sum products must be subject to a square root operations. Now we need to calculate the nominator side: $sumD(w)D'(W)$, which we have already calculated also using spreadsheet:

word	D1(w)	D2(w)	D1(w)^2	D2(w)^2	D1(w).D2(w)
to	2	1	4	1	2
be	2	1	4	1	2
or	1	0	1	0	0
not	1	0	1	0	0
doubt	0	1	0	1	0
truth	0	1	0	1	0
a	0	1	0	1	0
liar	0	1	0	1	0
		sums	10	6	4

As you can see on the table above the $sumD(w)D'(W) = 4$, now we can put all of them in the formula:

$$\theta = arccos\left(\frac{4}{\sqrt{10}\sqrt{6}}\right)$$

$$\theta = arccos\left(\frac{4}{\sqrt{10.6}}\right)$$

$$\theta = arccos\,(0.52)$$

$$\theta = 1.028157225$$

There you are, that is in summary how we will measure the distance between to documents.

Summary

Here is what you should do when measuring the distance between two documents:

1. list all words in a set, meaning there are no double words in a set all words are unique
2. count the frequency of occurance for each word in each document (WARNING: EACH NOT TOTAL)
3. use vector inner product and normalization also dot products to calculate the angle (SEE THE FORMULAS AND EXAMPLE ABOVE)
4. that angle when is closer to 0 (the minimum angle is 0) then the chance both documents are the same is higher (some use this as an indication of plagiarism)
5. Vice versa when the angle is closer towards $\frac{\pi}{2}$ then the documents are less similar.
6. Just remember this is only a basic algorithm on how to measure document distance, many ways can be used to cheat this algorithm thus this kind of measurements always evolving in terms of algorithms.
7. Stay tune and keep learning!

1 Coding in Python

Now after we know how to formulate the document distance, we can start use it to formulate pseudocode. The algorithm will be mesured after the pseudocode is implemented into code in python. Each python file here uses different algorithm or should I say method on how to handle the words inside the documents. These words are what used to measure the document distance later on using vector normalized multiplication.

Well it is better if we research each python file to understand the way they were calculated. I need to have method in order to debug the code later on to make sense of the algorithms they use. However, preparing method to debug the code is not as simple as it might sounds. For once the built in VSCode debugger does not allow the *args input from the user. Meanwhile the Python own debug library is lacking in the user interface features.

If I need to choose between the two, right now I more lean towards the VSCode internal debugger. I must make some adjustment in the main function in order to put the *args into the system. I need to make the test document (t1 and t2) still passed into the argument after the main function is called.

```
1         def main():
2             if len(sys.argv) != 3:
3                 print ("Usage: docdist1.py filename_1 filename_2")
4             else:
5                 filename_1 = sys.argv[1]
6                 filename_2 = sys.argv[2]
7                 sorted_word_list_1 = word_frequencies_for_file(filename_1)
8                 sorted_word_list_2 = word_frequencies_for_file(filename_2)
9                 distance = vector_angle(sorted_word_list_1,sorted_word_list_2)
10                print ("The distance between the documents is: %0.6f (radians)"%distance)
11
```

As you can see in the main function if the user does not include additional arguments in the CLI when invoking the docdist program it will invoke warning and stop the program altogether. In order to solve this problem I need to make the run will include the t1.verne.txt and t2.bobsey.txt in the initial parameters. Why we use t1.verne.txt and t2.bobsey.txt? Well because both files are the smallest of all documents in the project. This is merely a test run and debug run to see how the code works. Thus, choosing the smallest file as example for all docdist program (docdist1 to docdist6) will result comparable and comprehensive results.

In order to make the VSCode built in debugger works I need to modify the main function a bit. The main idea here is to make the test documents files part of the arguments from the first time debug run is initiated. As we cannot use CLI to run the debug state of the program then I need to modify the main function as this is the function called first on run. Here is the modification of the code:

```
1         def main():
2     # import pdb; pdb.set_trace() # <--- this is for debug purpose only
3     if len(sys.argv) != 3:
4         print ("Usage: docdist4.py filename_1 filename_2")
5         #-- FOR DEBUG PURPOSES ONLY
6         filename_1 = 'E:\\python_me\\6-006_python\\lec02_code\\t1.verne.txt'
7         filename_2 = 'E:\\python_me\\6-006_python\\lec02_code\\t2.bobsey.txt'
```

```
        sorted_word_list_1 = word_frequencies_for_file(filename_1)
        sorted_word_list_2 = word_frequencies_for_file(filename_2)
        distance = vector_angle(sorted_word_list_1,sorted_word_list_2)
        print ("The distance between the documents is: %0.6f (radians)"%distance)
        # --- comment out this after finish debugging!
    else:
        filename_1 = sys.argv[1]
        filename_2 = sys.argv[2]
        sorted_word_list_1 = word_frequencies_for_file(filename_1)
        sorted_word_list_2 = word_frequencies_for_file(filename_2)
        distance = vector_angle(sorted_word_list_1,sorted_word_list_2)
        print ("The distance between the documents is: %0.6f (radians)"%distance)

```

NOTE: the file paths are in Windows format since the files are stored in the local Windows storage. I think it will be safer to use Windows path format.

1.1 UML Sequence Diagram

I need tool to help me understand how the program works. As takin notes are too random and the contents are easier to forget, I think I need better methods to make sense how the program works. The Sequence Diagram from UML sounds like a good tool for this job. Unlike static Class Diagram the Sequence Diagram will record the interaction between function(?) in the program.