True or False? Elves *are* tall …
and Tolkien won't let you think otherwise

Experiments with Tolkien

Due: Wed. March 26 (by classtime, 2pm)

Conjecture: J.R.R. Tolkien wanted his readers to fully appreciate that his elves were large, thus he used the word *"tall"* (or other variants such as *"big", "giant", "large",* etc.) in close proximity to the name of an elf (e.g., *"Legolas"*, *"Galadriel"* or even the generic word, *"elf"*).

Execute a Python program to generate data that will help experimentally verify if this conjecture is true or false. Note: I already wrote the program. Yup, I'm giving you a working Python program. Your task in this assignment is to *use* a program to help collect evidence to make your case and then *professionally* write up a report that summarizes your evidence and conclusion.

Analysis: *This part is up to you; I've just written the program.* Work with the results, pour over the texts, and establish metrics that might provide some evidence one way or the other. For example, you could find the **average number of characters** between "tall" and [elf names]. But is this enough?

The best way to begin is to make up very simple files and play with the text. See the Starter Kit of course.

The beginning of a sample run with the input file **`test.txt`** is shown below.

```
Enter file to check: test.txt

Enter the keyword to search for (e.g., tall): tall

|foo tall foo Legolas foo Idril tall tall Legolas\n
|012345678901234567890123456789012345678901234567890123456789012345
|          |          |          |          |          |
|         10         20         30         40         50
|
```

Output file: **`outputTallElf.csv`**

Sheets

	A	B	C	D
1	input file used in this experiment: **test.txt**			
2				
3	MAX characters between words allowed: **500**			
4				
5			**Distance Between words**	Actual Text
6				
7	Location of tall	Location of legolas		
8	4	13	5	tall foo legolas
9	31	13	11	legolas foo idril tall
10	36	41	1	tall legolas

```
|foo tall foo Legolas foo Idril tall tall Legolas\n
|012345678901234567890123456789012345678901234567890123456789012345
|          |          |          |          |          |
|         10         20         30         40         50
|
```

Ok, now more (serious) play with the text …

Note: We must *not* count situations where another "*tall*" is found between an elf name and the keyword. For example: If you are searching *left* from the last "**tall**", you will find "Legolas" to the left, but in between those two words, you also find (another) "*tall"*: "**Legolas**" …. "*tall*" …. "**tall**". You would *not* include the count from "**tall**" left to "**Legolas**" since *"tall"* is in between.

```
================================================================================
Test distance between tall and legolas ....
================================================================================
```

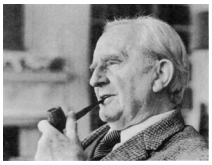
```
# we'll use a regular expression to find all matches of "tall" in the text
pattern = u'\s' + 'tall'
unicode_pattern = re.compile(pattern, re.UNICODE)
for match in re.finditer(unicode_pattern, text):
```

```
NEXT keyword: tall was found at: 4

**** LOOK LEFT -- no keyword tall found to left.
**** LOOK LEFT -- no elfname    legolas found to left.
      Distance between (legolas ... tall) is: 0

**** LOOK RIGHT -- the next tall is found at:         31
     LOOK RIGHT -- the next    legolas is found at:   13
      Distance between (tall ... legolas) is: 5
================================================================================
NEXT keyword: tall was found at: 31

**** LOOK LEFT -- the next tall is found at:          4
     LOOK LEFT -- the next    legolas is found at:    13
      Distance between (legolas ... tall) is: 11

**** LOOK RIGHT -- the next tall is found at:         36
     LOOK RIGHT -- the next    legolas is found at:   41
      Distance between (tall ... legolas) is: 0

================================================================================
NEXT keyword: tall was found at: 36

**** LOOK LEFT -- the next tall is found at:          31
     LOOK LEFT -- the next    legolas is found at:    13
      Distance between (legolas ... tall) is: 0

**** LOOK RIGHT -- no keyword tall found to right.
     LOOK RIGHT -- the next    legolas is found at:   41
      Distance between (tall ... legolas) is: 1

Done.
```

Consider the cases each time we find a new "tall" (LOOK LEFT)

```
def LookLeft (text, whereIsKeyword, keyword, elf):

   # text holds the entire text we are searching
   # whereIsKeyword holds the character location where "tall" was found
   # keyword holds "tall" (in this example)
   # elf holds "legolas" (in this example)

   ---------------------------------------------------------------------
   # nextKey = the location of keyword ("tall") when searching from
   #               right-to-left from [whereIsKeyword-1 to 0]
   nextKey   = text.rfind(keyword, 0, whereIsKeyword-1)

   #
   #
   nextElf   = text.rfind(elf, 0, whereIsKeyword-1)

   ---------------------------------------------------------------------
   # case (1): if no elfName is found to the left ( NOT_FOUND is -1 )

   if (nextElf == NOT_FOUND):

            return 0

   ---------------------------------------------------------------------
   # case (2): elfName was found, but no keyword is found to the left

   elif (nextKey == NOT_FOUND):

            distanceToElf = whereIsKeyword - (nextElf + len(elf))

            if (distanceToElf <= MAX_SEPARATION):

                  return distanceToElf

            else:

                  return 0

   ---------------------------------------------------------------------
   # case (3): both keyword and elf were found to the left

   else:
            # if the elfName was found further to right (later in text)?
            if(nextElf > nextKey):

                  distanceToElf = whereIsKeyword - (nextElf + len(elf))

                  if(distanceToElf <= MAX_SEPARATION):
                        return distanceToElf
                  else:
                        return 0
            else:
                  return 0
```

Grading Rubric



The more you do, the more points you will earn …

Average Effort

____correct statistics printed to the .csv (comma-separated-value) output file
____ number of instances where the distance is within a valid range
____ the constant MAX_SEPARATION is included in the output

____ correct AVERAGE DISTANCE (one place after the decimal point, e.g., `7.3`)
(Note: you can use an Excel formula to compute the average, e.g., **`=AVERAGE(C8:C25)`**)

____ neat formatted Excel formatted Tables of your experimental results; *(you must work in Excel to make these look nice but be careful to resave the file with a new name;* **Save As** *an Excel Workbook, .xlsx).* Each Table must have an appropriate last row (with no borders) that contains the text for the Table's legend. Submit your Excel file in your .zip folder of your work. Submit a professional-looking **hardcopy** printout of this work.

Above Average Effort

____ All Tables must be copied into a Word file that contains your final report. Your report must be professional; please do *not* submit reports with spelling errors, messy formatting, etc. The report must contain the sections:

1. **Introduction** – explain the experiment
2. **Method** – describe how the program works; give the "big picture" (not the nitty-gritty details).
3. **Results** – a professionally presented "story" of the evidence you have generated.
   All Tables *must* have professionally formatted legends and your report must refer to the Tables, e.g., "See Table 1 for a summary of all allowed distances between "tall" and all elf names."
   I will assume that your final report will have many Tables.
4. **Discussion** – So, what do you conclude? Why?

Submit your Word file in your .zip folder of your work. Submit a professional-looking **hardcopy** printout of this report.

Superior Effort

____ In addition to individual statistics for each pair ("tall", *elf name*), compute and print the overall statistics for *all* the elf names in a given file *(e.g., an overall average distance between all elf names and "tall").*

and/or

____ Determine how to present the data in a graphical form. If you were presenting this to your boss, she would give you perhaps(!) five minutes. How can you *best* make your case?

and/or

____ Research the elves used in particular books. Alter the mix of adjectives and elf names in the lists.

and/or

____ Alter the Python program to work in a more sophisticated fashion. Document the change in your report. *(Please check with me about the nature of the changes).*

and/or

____ Add an additional section to your report called "5. Future Work". Discuss, given more time, what you would like to explore.

and/or

____ Make a proposal for additional work. Impress us.

So, according to your evidence, is the conjecture true?