Baliganahalli Narayana Murthy, Sunil
Feb 3, 2017

CSCI 5832 : Natural Language Processing

Assignment -1

Prerequisite: The program should be run in python version 3 or later environment.

Preprocessing:

First the given document is pre-processed to clean up hyphens, apostrophes, remove periods in abbreviations, punctuations in numbers and remove dots in honorifics. This preprocessing will improve the accuracy and precision.

Remove Hyphens:

The words like *New-York*, *state-of-the-art*, *off-campus* etc are treated as a single word rather than two words. The entire document is processed to squeeze all the hyphens. As a result words like "*New-York*", "*state-of-the-art*" etc would now become one word - "*NewYork*", "*stateoftheart*" etc. It also removes hyphens separating words like "*Song - immodestly*", but it would still be treating them as two words because these would now be separated by whitespace.

Remove Apostrophes:

The words containing apostrophes should be considered as single word for e.g. '80s show, song's tempo, texas' weather, child's play etc. In order to consider them as one word, I remove all the occurrences of apostrophes. The main drawback of removing apostrophes blindly is that, it doesn't work for compacted words such as *We've, I've, wouldn't* etc which are in fact the composed of two words (*We have, I have, would not*). But I have treated them as one word.

Remove Dotted Abbreviations:

Abbreviations such as *B.P.M, W.H.O, 45 m.p.h* etc are treated as a one word in my program. I look for all the letters with at-least one period(dot) between them. This can be done using regular expression - `/(?:[a-zA-Z]\.[a-zA-Z]?\.?){2,}/`. If a match to such an expression is found, then I remove all the dots in that word. This processing would convert all the words such as *B.P.M* to *BPM.*, *45 m.p.h.* to *mph* etc.

The sentences are delimited by periods. Removing all the dots is not a good idea, because then I wouldn't be able to count the sentences. If the abbreviations has a period at the end, those are not trimmed in my earlier work. Hence they would count as an extra sentence.

For e.g. "*The bike was going at 20 m.p.h. and crashed into another bike.*" would count as 2 sentences because of period at mph.

I also tried to catch honorifics such as *Mr., Mrs.* etc to remove dots but it was hard to write a regular expression that matches abbreviations and honors. So I decided to tried honorifics on their own using dictionary.

Remove Periods or comma in numbers:

The document may also contain numbers which are separated by comma (for grouping) or period (for decimal). These would be treated as two words if comma or period are not removed. For e.g. *78,000* and *$45.56* should be treated as one word. Removing these punctuations from numbers would fix this. I used the regular expression - `/(\d+)([,.])(\d+)/` to find all the matches and replaced it with using group reference `/\1\3/`.

Expand honorifics:

Honors such as *Mr., Mrs., Prof.* etc should be treated as one word and also the period at the end shouldn't be treated as a end of sentence. I wasn't able to write a regular expression that would catch only these and not the words at the end of sentence. So I created a dictionary of limited but commonly used honorifics and their expanded form.

For e.g. `map = {"Mr." : "Mr", "Dr.": "Dr", "Rev.": "Rev", "Jr." : "Jr", "Sr.": "Sr", "Ms." : "Ms", "Prof." : "Prof"}`. And replace "Mr." with "Mr" etc.

Counting Words, Sentences and Paragraphs

Once the corpus has been preprocessed, I can use the regular expression to count number of words, sentences and paragraphs.

Words:

I assumed the english words are separated by whitespaces. I counted the words using the regular expression, `/\w+/`.

Sentences:

Sentences are separated by a period, or an exclamation or a question. I used regular expression `/[^\s](\.|\!|\?)(?!\w)/` to count number of sentences in the document. However abbreviations with period at the end would still be wrongly counted as a sentence.

E.g. : Mr. Jones who works for W.H.O. is here. [sentences = 2. Actual count = 1]

Paragraphs:

The paragraphs are separated by at least two new lines. Counting only the newlines would give us the number of lines in a document and not the paragraphs. I counted the paragraphs using the regex: `/\n{2,}/` and added one at the end to account for the last paragraph.

Improvements:

There is a lot of scope for improvement.

1. Improve the word count for compact words such as *"We've", "I've", "couldn't"* etc. By somehow be able to distinguish these words from words such as *"child's", "80's", "texas' girl"* etc.
2. Right now the words with hyphens are treated as one word. As a side effect it also treats range of time/year such as 9am-10am, 1960-1970 also as one word. One could argue that these are in-fact two words.
3. Sentences are delimited with either an period, exclamation or question mark. I can include semicolon, also as a end of sentence.
4. The list of honorifics are limited. I can include more honorifics to improve the precision of sentence recognition.

Output:

On development file (*10fitness.txt*):

```
Paragraphs: 8
Sentences: 20
Words: 646
```

On test file (*test_file_PA1.txt*):

```
Paragraphs: 10
Sentences: 36
Words: 646
```