

# *MANUAL*

# *TEXTEXTRACTOR*

Regular Expressions in Textfiles with Python (*Windows Version*)

*Jasper van Boven*

*Radboud Universiteit*

## Table of Contents

<b>1. What does the script do? .....</b>	<b>2</b>
<b>2. How to prepare my data for the script? .....</b>	<b>2</b>
Filetypes: .....	2
Filenames: .....	2
Map with files.....	2
<b>3. How do regular expressions work? .....</b>	<b>2</b>
Regular expressions language .....	2
Basic expressions.....	3
One word:.....	3
One of the following words:.....	3
Combinations of words: .....	3
Words that can be written differently: .....	4
The easy way out: .....	4
Combined: .....	5
Expressionsfile:.....	5
<b>4. How does the script work? .....</b>	<b>5</b>
<b>5. FAQ &amp; Troubleshoot.....</b>	<b>7</b>

## 1. What does the script do?

This script enables the user to look up certain words and combinations of words in batches of text documents in an automatic way. This automated script brings contentanalysis to the next level by analyzing amounts of documents that would have been too much to do manually.

## 2. How to prepare my data for the script?

### Filetypes:

The script only works with textdocuments (.txt). So make sure all the worddocuments , pdf-files etc. are converted to text. There are multiple ways achieve this. Click [here](#) to go to a site that gives you multiple options.

### Filenames:

It is necessary to remove all spaces from the names of the documents you want to analyze. You can achieve this by using bash scripts in the command prompt window or use an application like [this](#) one and follow the manual they have put online.

### Map with files

The script needs a map with the files as input. Make sure this map only contains the textdocuments you want analyzed. Remove all other files from this map. It is recommended to make sure the name of the map does not contain any spaces.

## 3. How do regular expressions work?

### Regular expressions language

In order to search analyze the textdocuments, the script uses so called Regular Expressions. Regular Expressions are a universal formal computer language. Based on a code it enables the script to look for certain words, multiple words or combination of words in sentences. Following [this](#) link you find a book giving very clear explanations on the Regular Expression code.

## Basic expressions

In the book which has been mentioned in the previous part, a few basic regular expressions are mentioned that are very useful in the content analysis as aimed for with this script. In the next part these will be explained. The expressions always start and end with a basic code and then the words you need can be inserted between the brackets. The basic code makes sure the whole sentence is selected and gets written in one of the output files.

### One word:

```
^\.*\b(test)\b.*$
```

This expressions looks for one word. The word “test” can be replaced by any word you want to look for in the textdocuments.

### One of the following words:

```
^\.*\b(one|two|three)\b.*$
```

With this expression you can look for one of the following words. The piper can be interpreted as “or”. So this one would say: “This sentence is a hit if one or two or three is mentioned in the sentence.”

### Combinations of words:

```
^(?=.*?\b(one)\b)(?=.*?\b(two)\b)(?=.*?\b(three)\b).*$
```

This expression already becomes a bit more complex. This one searches in both one, and two, and three are found in the sentence. By removing `(?=.*?\b(three)\b)` for example, it will look for just one and two. By adding `(?=.*?\b(four)\b)` directly after `(?=.*?\b(three)\b)`

the expressions will also look for the appearance of four in combination with the others. The order of appearance does not matter.

#### Words that can be written differently:

Some words can be written differently or conjugated in different ways. Still you want to use an efficient way to look for all these forms of the word. The Regular Expression language is built for this. Let's look at the word information, it would end up something like this:

Inform(ed|ing|ation)?

By placing "|", "?" and "(" on strategic places the expression can be made smart. The above expressions always looks for "inform" but also for "informed", "informing" and "information". The parts between the brackets says to look for "ed" or "ing" or "ation" in combination with inform. The questionmark after the closing bracket tells that it is not necessary to find one of these three per se, so that only "inform" will also be a hit.

In the book, the explanation of the code goes way deeper. Take a good look at that and don't feel overwhelmed. It is a great skill to master and you can use the basics within a few tries.

Conclusion:

With ( ) you can make groups

With ? the group of single character before becomes optional

with | you can make a distinction between multiple options

#### The easy way out:

The easy way, however way less efficient and esthetically appealing is to just put the options in one long row like this:

Inform|informed|information|informing

### Combined:

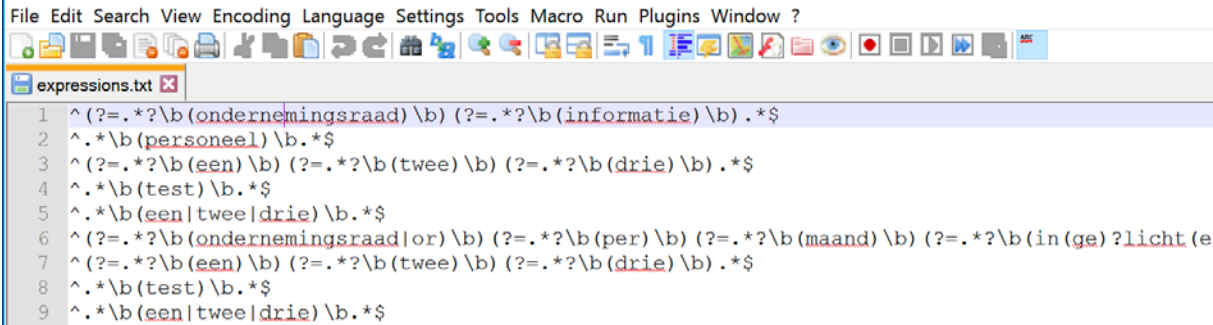
These codes can all be combined into one big expression that looks for multiple combinations of groups and words. The following is one to strive for in your own analysis

```
^(?=.*?\b((works?)?council)\b)(?=.*?\b(month(ly)?)\b)(?=.*?\b(Inform(ed|ing|ation)?)\b).*
```

In other words: look for sentences where the workcouncil, works council or council gets some kind of information or other form every month or monthly.

### Expressionsfile:

In order to use the expressions they need to be put in a textfile. One of the best applications in Windows is "Notepad++". Put each expressions in its own individual row respectively which will result in something like this:



```
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
expressions.txt
1 ^(?=.*?\b(ondernemingsraad)\b)(?=.*?\b(informatie)\b).*
2 ^.*\b(personeel)\b.*
3 ^(?=.*?\b(een)\b)(?=.*?\b(twee)\b)(?=.*?\b(drie)\b).*
4 ^.*\b(test)\b.*
5 ^.*\b(een|twee|drie)\b.*
6 ^(?=.*?\b(ondernemingsraad|or)\b)(?=.*?\b(per)\b)(?=.*?\b(maand)\b)(?=.*?\b(in(ge)?licht(e
7 ^(?=.*?\b(een)\b)(?=.*?\b(twee)\b)(?=.*?\b(drie)\b).*
8 ^.*\b(test)\b.*
9 ^.*\b(een|twee|drie)\b.*
```

## 4. How does the script work?

The script is written in the computer language "Python" and executed in the command prompt which is a part of every windows distribution. The one necessity is to have Python 2 installed on your machine. If it is not on your computer yet you can find it following the next link:

<https://www.python.org/downloads/>

Make sure to install Python 2.7.XX because it does NOT work in Python 3.

To run the script you have to drag the file "textextractor.py" to the command prompt window. Press enter and you will get the following lines: "[ ] drag the map with your files here" and "path to map:". Drag the map with your textdocuments to the command prompt window and the path to the map will show up in the window. Depending on your version of windows the path might need a backslash "\" at the end of the path. Press enter and you will go to the "save in map" section. If you get an error, try again and this time check for typos, the backslash at the end and try to add or remove the "" at the beginning and end of the path.

The window will now display "[ ] drag the map where you want to save the output here" and "save in map:". Drag the map where you want to save the output. Use another map than the one with the textdocuments. Depending on your version of windows the path might need a backslash "\" at the end of the path. Press enter and you will go to the "regexfiles" section. If you get an error, try again and this time check for typos, the backslash at the end and try to add or remove the "" at the beginning and end of the path.

Now drag the textfile with your expressions to the window. Press enter and all the regular expressions you created appear. They might look different, because Python automatically transformed the expressions to the preference of the Python language. If the expressions do not appear, an error in the path occurred. Try again and this time try to add or remove the "" at the beginning and end of the path and check for typos.

Press enter again and the analysis will start. Depending on the amount of textdocuments and regular expressions it might take a while. If you get an error, there is a problem with your regular expressions. Take a good look at them and identify the error. Tip: run the expressions individually on a small test batch to identify the one with the mistake.

The script creates the textfiles (count and output) where the results are stored. Count is the list where each document and its matched expression are saved. This one is perfect for import in excel and transform it using a pivot table. In output you find all the sentences that have been identified, which might be useful to get more insight into the contents of your analysis.

## 5. FAQ & Troubleshoot

The main errors happen through either typos in the regular expressions or problems with the paths to the maps and file with regular expressions. I suggest to just try the suggestions that have been written down at each specific part of this manual.

No other errors are currently known.

*This script is licensed under the MIT license*

*Jasper van Boven*

*2018*