

Python Tool Text-analysis

Handover document

30 december 2020

# Version Control

Student

Author J. Prins

Student number S1118825

Study course Logistics Engineering

Professorship Supply Chain Finance

Educational institute Windesheim University of Applied Sciences Zwolle

Engineering & Design

Email jacco.prins@windesheim.nl

jaccoprins98@gmail.com

30-12-2020

Versie 1.0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Adjustments** | **Author(s)** | **Status** |
| 0.1 | 29-12-2020 | Draw up headers of document | J. Prins | Template version |
| 1.0 | 30-12-2020 | Fill-in document | J. Prins | Finished for hand-in |

Table of Contents

[Version Control 1](#_Toc60229683)

[1. Introduction 3](#_Toc60229684)

[2. User guide 4](#_Toc60229685)

[2.1. Installation 4](#_Toc60229686)

[2.2. Starting the analysis 6](#_Toc60229687)

[2.3. Analysing results 6](#_Toc60229688)

[3. Code design 8](#_Toc60229689)

[3.1. Import statements 8](#_Toc60229690)

[3.2. Preparation statements 9](#_Toc60229691)

[3.3. Creating window 10](#_Toc60229692)

[3.4. Keywords table 10](#_Toc60229693)

[3.5. Top-10 table 11](#_Toc60229694)

[4. Recommendations 13](#_Toc60229695)

[4.1. Parsing process 13](#_Toc60229696)

[4.2. General add-ons 13](#_Toc60229697)

[4.3. Output design 13](#_Toc60229698)

[Appendix 1: Output Report.docx 14](#_Toc60229699)

# Introduction

This tool has been made as a part of an internship at the Professorship Supply Chain Finance of Windesheim University of Applied Sciences. It has been developed to identify more quickly what papers and interview-transcripts were likely to contain useful information for the assignment of the internship. The goal was to scan a set of .docx- and/or .pdf-documents for the occurrence of certain keywords and to retrieve a top-10 list of the most common words in the text. Based on these outcomes, a selection of the documents could be made that could be subjected to further analysis.

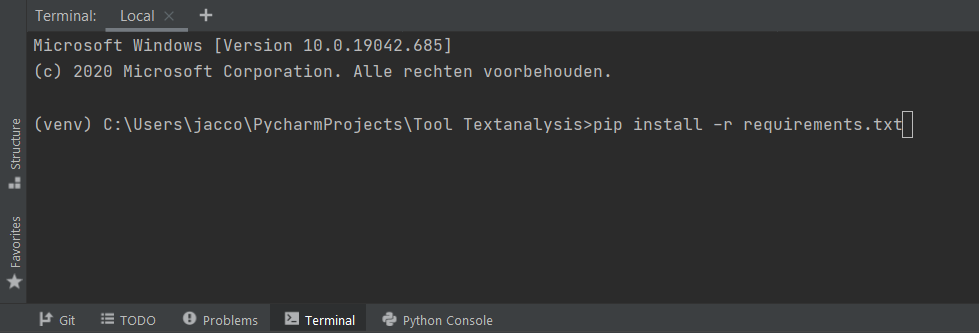
This document will serve as a handover document to transfer this tool to the professorship, even though the student left the organization. It will start with a user guide to explain the capabilities of the tool, and the steps the user needs to take to receive the wanted outcomes. After this, the actual Python code will be explained. This will clarify what steps were taken to create the model. Finally, list of recommendations for further development will be presented. This list mostly resulted from the progress meetings that were held with the researchers and other students of the professorship.

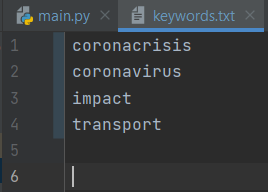
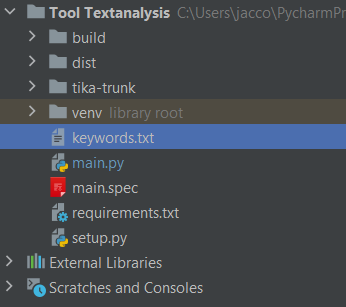
# User guide

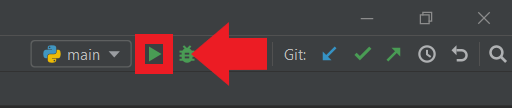
## Installation

This tool can be installed by two methods:

1. Downloading the required documents and code from GitHub
   1. Afbeelding met tekst

      Automatisch gegenereerde beschrijvingGo to <https://github.com/JaccoPrins/Tool_Textanalysis> or <https://github.com/Hogeschool-Windesheim/Tool_Textanalysis> and download the documents as shown below
   2. Extract the .zip-file with help of, for example, 7-Zip (<https://www.7-zip.org/>), and save the directory somewhere on the computer
   3. Download a code editor for Python, for example PyCharm (<https://www.jetbrains.com/pycharm/>) and download Python (<https://www.python.org/downloads/>.
   4. Configure a Python interpreter, which is step-by-step explained for PyCharm on the following webpage: <https://www.jetbrains.com/help/pycharm/configuring-python-interpreter.html>
   5. Open the extracted directory in PyCharm and run ‘pip install -r requirements.txt’ in the terminal like shown below. This will install all the needed packages for the tool.
   6. Open the file keywords.txt as shown below on the left, which opens the document like shown below on the right. Fill in the keywords of which the occurrence in the documents need to be counted. Please mind: The keywords need to be in lower-case!



* 1. Run the script by pressing [Ctrl + Shift + F10] or by clicking the button like shown below.
  2. In case an error pops up, or the next window does not show up in 1 minute, downloading the latest version of Visual C++ Redistributable (<https://support.microsoft.com/en-ca/help/2977003/the-latest-supported-visual-c-downloads>) might be necessary. After downloading, restart the computer and try again.

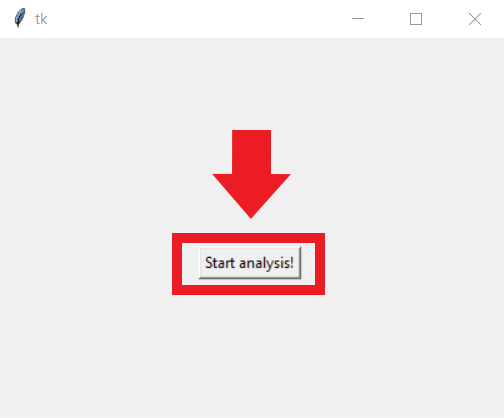
1. Run without Python by using the .exe-file.
   1. Get the .exe-file of the tool, available on request from Jacco Prins ([jacco.prins@windesheim.nl](mailto:jacco.prins@windesheim.nl) or [jaccoprins98@gmail.com](mailto:jaccoprins98@gmail.com).).
   2. Save the .exe-file in an empty directory and add a .txt file to the same directory, named keywords.txt.
   3. Open the file keywords.txt and fill in the keywords of which the occurrence in the documents need to be counted. Please mind: The keywords need to be in lower-case.
   4. Run the .exe-file by double-clicking it, it will take about 1,5 minute before the next window shows up.
   5. In case an error pops up, or the next window does not show up in 3 minutes, downloading the latest version of Visual C++ Redistributable (<https://support.microsoft.com/en-ca/help/2977003/the-latest-supported-visual-c-downloads>) might be necessary. After downloading, restart the computer and try again from step a.

## Starting the analysis

After the tool started running, the following steps need to be taken to start the analysis.

1. The tool will directly ask to select the directory in which the files that need to be analysed are located. The tool supports the following languages:
   1. Dutch
   2. English
   3. Spanish
   4. French
   5. German
   6. Italian

Besides that, make sure that the first 3 characters serve as index for the documents and that all the files in this directory are supported by the tool, of which a list is drawn up on the following webpage: <https://www.tutorialspoint.com/tika/tika_file_formats.htm>

1. Select the file and press ‘Start Analysis’, like shown below.
2. Dependently on the number and size of the files in the directory, and the performance of the computer, this will take some time. Eventually, a document called report.docx will show up on the Desktop.
3. In case an error pops up or file does not show up, downloading the latest version of Java (<https://www.java.com/nl/download/ie_manual.jsp>) might be necessary. After downloading, restart the computer and try again from step 1.

## Analysing results

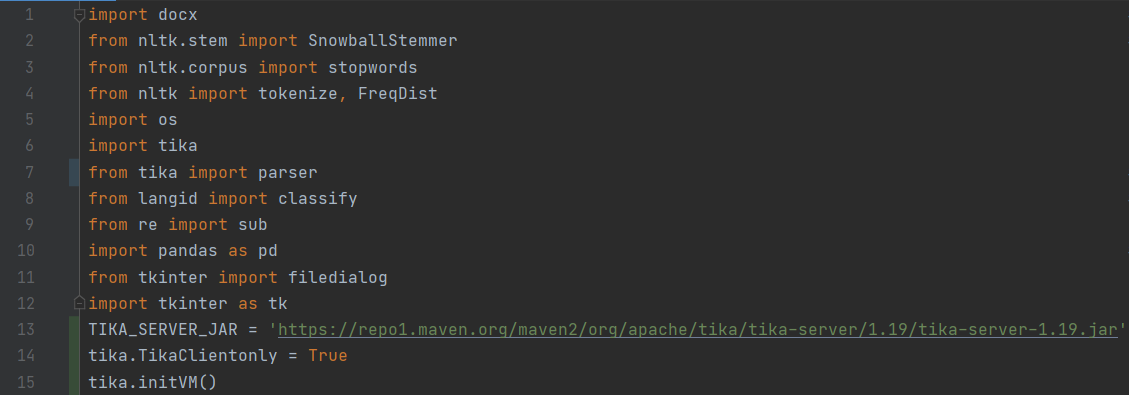
After the file showed up on the Desktop, the results can be analysed. The document (of which an example can be found in Appendix 1), results in a two kinds of tables:

1. The first table shows the occurrence of the pre-set keywords in the file keywords.txt. The first column of this table returns the first 3 characters of each source. The second column returns the language in which the document is written. Next to this column, the occurrence of each keyword is returned. Finally, the last column returns the total amount of the occurrence of all the keywords, this is also the column in which the documents will be sorted descending.
2. The other tables return the top-10 list of the most common words in the text and the number in which theses words occur per document. These words are often in a shortened version, since the plural- and singular-form of words needed to be added up. In this way, it does not matter in which form the words occurs in the text.

# Code design

The code has been built up in several sectors, containing a couple of rows of code. After most rows, a short description of the action of the specific row has been incorporated. This chapter will explain more thoroughly what steps have been taken by making the tool.

## Import statements

Like shown in the figure below, the tool has been built up by several packages. Besides that, the Tika server is configurated in the rows 13, 14, and 15.

**Docx**

|  |  |
| --- | --- |
| Install command | Pip install python-docx |
| Current version | 0.2.4 |
| Application | Creating and updating .docx-files with Python with customizable fonts and table-layouts |
| Installation | <https://python-docx.readthedocs.io/en/latest/user/install.html> |
| Documentation | <https://python-docx.readthedocs.io/en/latest/index.html#api-documentation> |

**NLTK**

|  |  |
| --- | --- |
| Install command | Pip install nltk |
| Current version | 3.5 |
| Application | Library for Natural Language Processing, such as counting the occurrence of certain words, deleting stopwords, removing morphological affixes from words to leave only the word stem |
| Installation | <https://www.nltk.org/install.html> |
| Documentation | <https://www.nltk.org/> |

**Apache Tika**

|  |  |
| --- | --- |
| Install command | Pip install tika |
| Current version | 1.24 |
| Application | Java tool that detects and extracts metadata and text from over a thousand different file types |
| Installation | <https://github.com/chrismattmann/tika-python> |
| Documentation | <https://tika.apache.org/0.5/documentation.html> |

**Langid**

|  |  |
| --- | --- |
| Install command | Pip install langid |
| Current version | 1.1.6 |
| Application | Detects language from text |
| Installation | <https://github.com/saffsd/langid.py> |
| Documentation | <https://github.com/saffsd/langid.py> |

**Pandas**

|  |  |
| --- | --- |
| Install command | Pip install pandas |
| Current version | 1.1.4 |
| Application | Data analysis and creating dataframes |
| Installation | <https://pandas.pydata.org/docs/getting_started/index.html> |
| Documentation | <https://pandas.pydata.org/docs/> |

**Tkinter**

|  |  |
| --- | --- |
| Installation | Not necessary, package is built into the Python standard library |
| Application | A platform independent windowing toolkit, which creates simple graphical user interfaces |
| Documentation | <https://docs.python.org/3/library/tk.html> |

**Re**

|  |  |
| --- | --- |
| Installation | Not necessary, package is built into the Python standard library |
| Application | Provides regular expression (re) matching operations to process raw text |
| Documentation | <https://docs.python.org/3/library/re.html> |

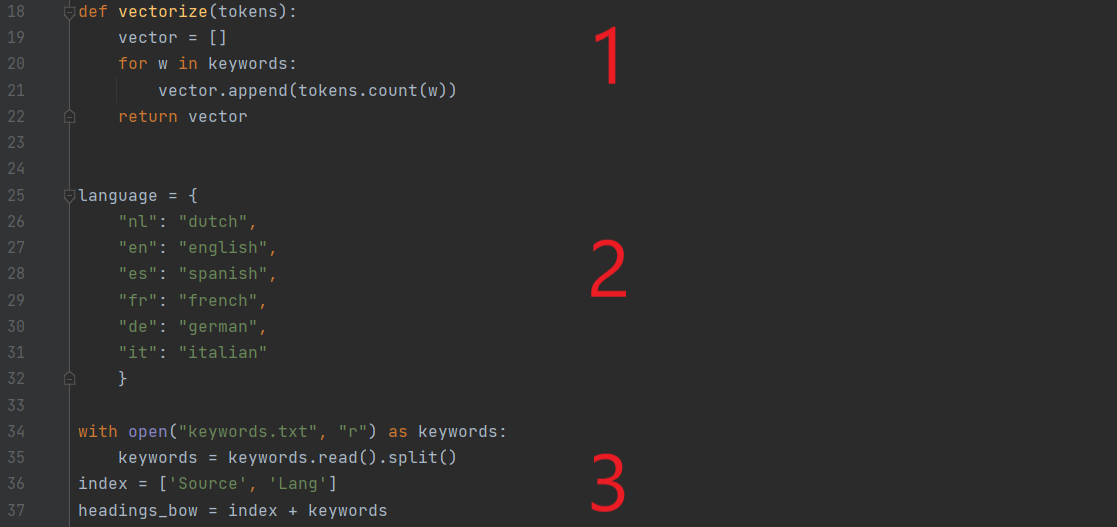
**Os**

|  |  |
| --- | --- |
| Installation | Not necessary, package is built into the Python standard library |
| Application | Provides a portable way of using operating system dependent functionality, as saving, and opening documents |
| Documentation | <https://docs.python.org/3/library/os.html> |

## Preparation statements

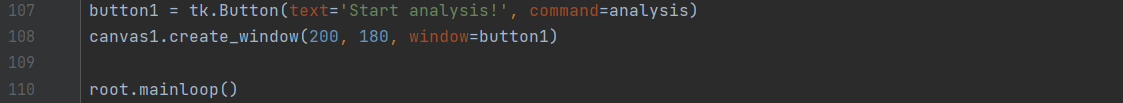
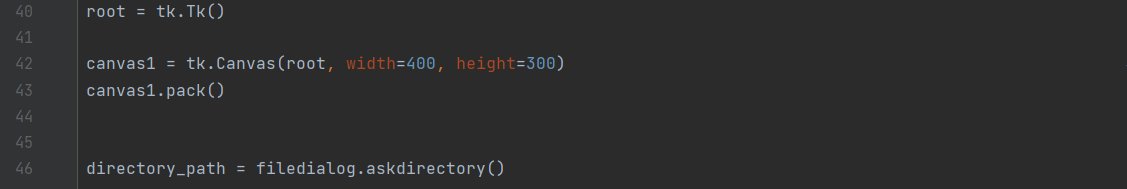
The following statements serve as preparation for the actual script and have the following functions:

1. An empty list that gets filled in with information for the keywords table. This statement will count the occurrence of the pre-selected keywords and add this to the list. With help of Pandas, a dataframe can subsequently be created.
2. A transformation statement to change the Alpha-2 code of countries to full country names. This also indicates the languages that are supported by the tool.
3. Opens the keywords.txt-file. This statement also creates the keywords as headings in the keywords table.



## Creating window

The rows below show the window that shows up by executing the tool. Row 46 makes it directly ask to select the directory in which the to-be-analysed documents are located. The rows 107 and 108 create a button to execute the analysis.



## Keywords table

The keywords table is one of the output tables of the tool. The figure below shows the code that provides this table like shown as the first table in Appendix 1. After the initial format of report.docx is created in rows 50 up to and including 53, the following steps are taken for each document in the selected directory:

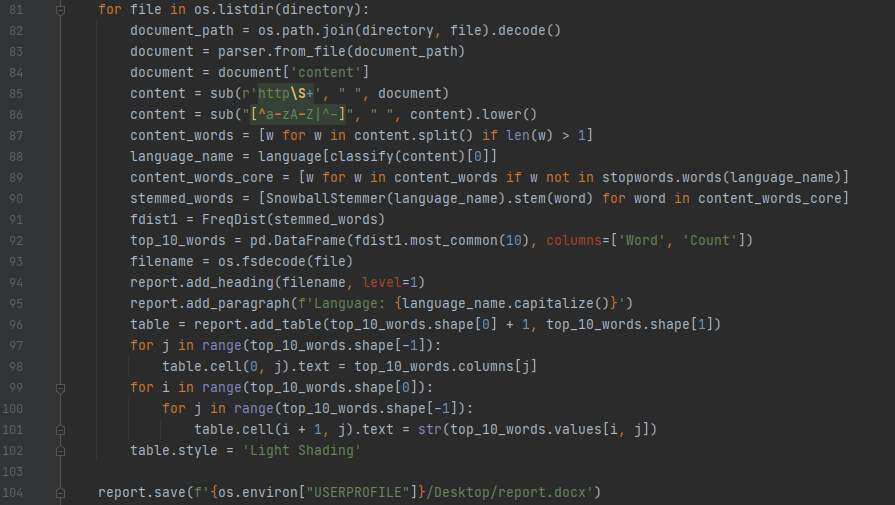
1. Retrieve the text (parsing) from the document (row 55-56)
2. Delete all the weblinks and punctuation of the document (row 57-59 and 62)
3. Put the remaining words in a list (row 60)
4. Detect the language of the text (row 61)
5. Delete all words of the text, except for the words that are preliminary filled in in the keywords.txt-file (row 62)
6. Count the occurrence of the keywords in the text (row 63)
7. Add the occurrence per keyword, together with the language and the first three letters of the document, to a dataframe (row 64-67)
8. Add headings and a totals-column to the dataframe and sort the documents in descending order (row 68-72)
9. Add the dataframe as a table to report.docx and change the style of the table (row 73-79)



## Top-10 table

The top-10 most common words table is the other output of the tool, like shown in Appendix 1 after the keywords table. After the first table has been made, this script like shown below will run. The following steps are taken for each document in the selected directory:

1. Retrieve the text (parsing) from the document (row 82-83)
2. Delete all the weblinks, punctuation, words with only one letter, and adverbs of the document (row 84-87 and 89)
3. Detect the language of the text (row 88)
4. Group different forms of a word to a single item (row 90)
5. Count the occurrence of each word and put the top-10 in a dataframe (row 91-92)
6. Add the filename and the language of the document to report.docx as a subtitle (row 93-95)
7. Add the top-10 dataframe as a table to report.docx, add headers, and change the style of the table (row 96-102)
8. Save the file to the Desktop of the computer (row 104)



# Recommendations

While developing the tool and asking for feedback from other students and researchers from the professorship, several recommendations and add-ons were thought of. These will be discussed in this chapter.

## Parsing process

The process of parsing (identifying words in a document) could be improved at the following points:

1. Automatically delete, for example, cookie notices and weblinks from webpages to only parse the core-text of the article.
2. Accept letters with diacritical marks like ü or è.
3. The combination of ‘fl’, ‘fi’, and ‘ff’ can not yet be parsed from a .pdf-file, which results in ‘broken’ words (like ‘dene’ in stead of ‘define’). This could be fixed by replacing these the resulting ‘broken’ words.
4. In the current program, all ‘words’ with only 1 letter get deleted from the analysis. However, ‘u’ is an important word in Dutch. An exception for this could be implemented in a new version.
5. The current model supports one language per document. This could be improved by making it possible to support several languages per document.
6. This tool is also used to analyse transcripts of interviews. It would be useful when a new version could distinguish questions and answers, since only the answers are actually important for further analysis.

## General add-ons

The code of the tool itself could be improved at the following points:

1. Mainly by using the tool by the .exe-file, the waiting times are quite long. This performance could, for example, be improved by deleting unnecessary lines, since some lines are executed twice (for both the table for the occurrence of keywords as the table with the top-10 most common words).
2. Adding a sentiment analysis to make it possible to distinguish positive and negative perceptions of certain keywords.
3. The text of the documents automatically gets transformed into lower-case letters. This is not yet realised for the words entered in keywords.txt, which could be done in a new version.
4. Adding a web-scraper to make it possible to not only parse documents in a certain directory, but also articles on webpages.
5. Automatically add numbers to the documents to identify them in the keywords table of the output. However, mind that these numbers should be traced back to the documents.
6. By making the list of the keywords, the user need to think of the language in which these keywords need to be written down. In case these keywords will be used for documents with different languages, it might not give the right results. This could be improved by automatically translating the keywords to the language of each document separately.

## Output design

The output of the tool can be improved at the following points:

1. The current dialog in which the directory gets selected and the analysis is started is just a grey window. This could be improved into a better-looking layout.
2. The output document could be improved by using graphs instead of tables. This will result in a clearer overview of the relevance of certain documents.
3. The words in the tables with the top-10 most common words are often in a shortened version. This could be improved by returning the stem of the words.

# Appendix 1: Output Report.docx

Analysis Articles Python

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Source | Lang | coronacrisis | coronavirus | impact | transport | Total |
| 001 | Dutch | 6 | 1 | 0 | 8 | 15 |
| 002 | Dutch | 3 | 5 | 4 | 3 | 15 |
| 003 | Dutch | 1 | 2 | 0 | 10 | 13 |
| 004 | English | 0 | 5 | 6 | 0 | 11 |
| 005 | English | 0 | 3 | 4 | 2 | 9 |

**001. Transport-Online.nl - Transportsector vraagt ondersteuning overheid om vakmanschap en werknemers tijdens de coronacrisis te behouden.pdf**

Language: Dutch

|  |  |
| --- | --- |
| Word | Count |
| logistiek | 10 |
| sector | 9 |
| nieuw | 8 |
| transport | 8 |
| werknemer | 7 |
| chauffeur | 6 |
| coronacrisis | 6 |
| we | 6 |
| verwacht | 5 |
| nederland | 4 |

**002. vanoers.nl - Coronavirus en wegtransport - Van Oers.pdf**

Language: Dutch

|  |  |
| --- | --- |
| Word | Count |
| nieuw | 9 |
| procent | 8 |
| coronados | 7 |
| menu | 7 |
| wel | 6 |
| oer | 6 |
| coronavirus | 5 |
| wegtransport | 5 |
| international | 5 |
| verwacht | 5 |

**003. waltherploosvanamstel.nl - Crisis in transport\_ creëer je eigen kansen - Walther Ploos van Amstel.pdf**

Language: Dutch

|  |  |
| --- | --- |
| Word | Count |
| crisis | 21 |
| kans | 20 |
| transport | 19 |
| eig | 18 |
| cre | 16 |
| cookies | 14 |
| our | 13 |
| ondernemer | 11 |
| waltherploosvanamstel | 10 |
| klant | 10 |

**004. mckinsey.com - Coronavirus’s impact on supply chain \_ McKinsey.pdf**

Language: English

|  |  |
| --- | --- |
| Word | Count |
| suppli | 20 |
| supplier | 20 |
| chain | 17 |
| product | 17 |
| capac | 15 |
| custom | 14 |
| use | 12 |
| demand | 12 |
| supply-chain | 11 |
| sourc | 11 |

**005. s-ge.com - The Corona outbreak and its impact on logistics \_ S-GE.pdf**

Language: English

|  |  |
| --- | --- |
| Word | Count |
| servic | 9 |
| capac | 9 |
| custom | 8 |
| china | 7 |
| dhl | 6 |
| europ | 6 |
| impact | 5 |
| global | 5 |
| carrier | 5 |
| equip | 5 |