Handbuch für

Module zur Texttrennung

Mark Unger und Siegfried Kienzle

16. Februar 2017

Erklärung

Die in diesem Projekt verwendete Software unterliegt den rechtlich jeweiligen Bestimmungen der einzelnen Organisationen und Firmen.

Inhaltsverzeichnis

1 Modul

1.1 Über die Software

Mit diesen Modulen kann man die Kopf- und Fusszeilen entfernen und die entsprechenden Zeilen trennen. Es wurde für Python 3.4.3 entwickelt und unter Ubuntu 14.04.05 LTS getestet. Zur Installation liegt ein Bash-Script vor.

1.2 Über das Handbuch

Dieses Handbuch beschreibt die Installation und die Handhabung mit den Modulen.

2 Grundlagen

2.1 Installation

1. Installationsscript mittels ./inst.sh aufrufen:

Abbildung 1: Nach Aufruf des Installationscriptes ./inst.sh

2. sudo-Passwort eintippen und die Enter-Taste drücken.
3. Es werden nun einige Abhängigkeiten installiert, die zur Ausführung dieses Moduls benötigt werden.

4. Geben Sie nun den Pfad an, in den das Modul installiert werden soll. Sollte der Pfad nicht existieren, werden Sie wie in Abbildung 4 gefragt ob der Pfad erstellt werden soll. Existiert der Pfad, entfallen die Schritte 6 bis 8.

Abbildung 2: Nach Aufruf des Installationscriptes ./inst.sh

Abbildung 3: Nach Eingabe des Installationspfads

5. Wenn der OK-Button blau hinterlegt ist, können Sie mit der Enter-Taste den Pfad bestätigen.

6. Sollte kein Pfad existieren, erscheint folgendes Fenster:

Abbildung 4: Pfad erstellen?

7. Wählen Sie nun mit den Pfeiltasten aus, ob Sie den Pfad erstellen möchten oder nicht und drücken Sie dann die Enter-Taste.

Abbildung 5: Pfad wurde erstellt

8. Es wurde nun der Pfad erstellt. Drücken Sie nun die Enter-Taste, um die Dateien in das entsprechend vorher erstellte Verzeichnis, zu entpacken.

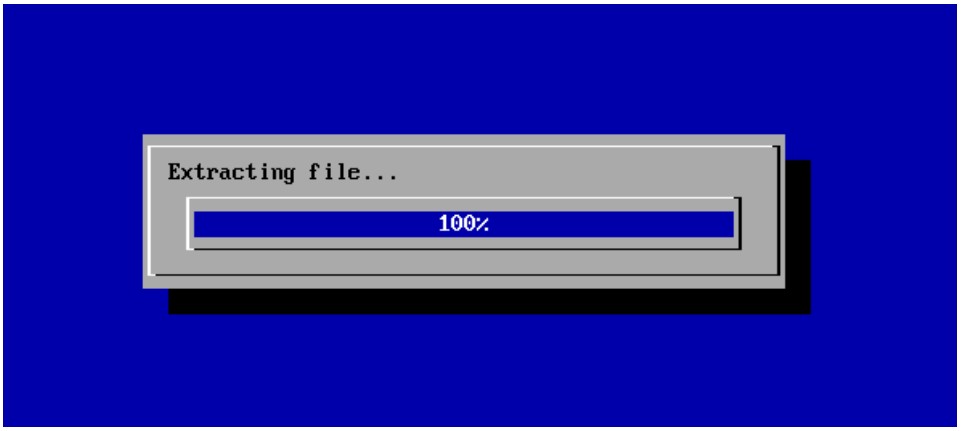

Abbildung 6: Pfad wurde erstellt

9. Die Installation ist nun abgeschlossen. Prüfen Sie nun bitte ob alle Dateien installiert wurden. Eine genaue Auflistung finden Sie unter dem Punkt 2.3.

2.2 Bestandteile Installationspaket

Datei	Beschreibung
inst.sh	Bash-Script für die Ausführung als Super-User (sudo) unter Ubuntu
ubuntSep.sh	Bash-Script für die Installation unter Ubuntu
files.tar	Tar-Datei, die die Python-Module enthält

Tabelle 1: Bestandteile Installationspaket

2.3 Modulbestandteile

Datei	Verwendung
checker.py	Entfernt die Kopf- und Fußszeilen
seperator.py	Trennt die Sätze
trainy.py	Erstellt Regelwerk zum Trennen von Sätzen
business.pickle	Regelwerk zum Trennen von Sätzen

Tabelle 2: Modulbestandteile

2.4 Erste Schritte

2.4.1 Genereller Aufruf

Es gibt drei Skripte:

- checker.py
- seperator.py
- trainy.py

Alle drei besitzen entsprechende Parameter die im allgemeinen so aufgerufen werden:

`python3 <PYTHONSCRIPT> <PARAMETER> (<PFAD>)`

Dabei sollte <PYTHONSCRIPT> durch eins der oben genannten Pythonskripte und <PARAMETER> durch Parameter jeweils aus den Tabellen 3, 4 und 5 ersetzt werden. Alle drei Skripte besitzen außerdem die Parameter -i bzw. --input und -o bzw. --output. Bei diesen Parametern sollte dann zusätzlich <PFAD> (das hier in runden Klammern steht) durch den Pfad der Datei, aus der der Text extrahiert bzw. in den der Text geschrieben werden soll, ersetzt werden. Außerdem ist zwingend darauf zu achten, dass das Modul mit python3 aufgerufen wird.

Parameter (kurz)	Parameter (lang)	Beschreibung
-h	--help	Hilfetext wird angezeigt
-i	--input	führt das Skript aus
-o	--output	Parameter für die Ausgabedatei. Nur anwendbar mit Parameter -i bzw. –input
-r	————————	fügt Wiederholungen an das Ende einer Datei an
-v	————————	Verbose-Mode: gibt den Text auf Konsole aus.

Tabelle 3: Parameterübersicht von checker.py

Parameter (kurz)	Parameter (lang)	Beschreibung
-h	--help	Hilfetext wird angezeigt
-i	--input	führt das Skript aus
-o	--output	Parameter für die Ausgabedatei. Nur anwendbar mit Parameter -i bzw. –input
-v	————————	Verbose-Mode: gibt den Text auf Konsole aus.

Tabelle 4: Parameterübersicht von seperator.py

Parameter (kurz)	Parameter (lang)	Beschreibung
-h	--help	Hilfetext wird angezeigt
-i	--input	führt das Skript aus
-o	--output	Parameter für die Ausgabedatei. Nur anwendbar mit Parameter -i bzw. –input

Tabelle 5: Parameterübersicht von trainy.py

Beispielaufrufe finden Sie in den weiteren Kapiteln.

2.4.2 Entfernen von Kopf- und Fußzeile

Zum Entfernen von Kopf- und Fußzeile tippen Sie einfach

```
python3 checker.py -i <DATEIPFAD> -v
```

oder

```
python3 checker.py --input <DATEIPFAD> -v
```

Als Beispiel sehen Sie im Folgenden wie die Kopf- und Fußzeile aus einer Textdatei entfernt wird:

```
python3 checker.py -i test_bsp.txt -v
```

oder

```
python3 checker.py --input test_bsp.txt -v
```

2.4.3 Entfernen von Kopf- und Fußzeile und anschließendes schreiben in eine Datei ohne Konsolenausgabe

Zum Entfernen von Kopf- und Fußzeile aus einer Datei, ohne dabei den Text auf die Konsole auszugeben, tippen Sie einfach

```
python3 checker.py -i <DATEIPFAD> -o <AUSGABEDATEI>
```

oder

```
python3 checker.py --input <DATEIPFAD> --output <
   AUSGABEDATEI>
```

Es ist zu beachten, dass wenn die angegebene Ausgabedatei bereits existiert, der Inhalt durch den Text der im Moment durch Kopf- und Fußzeilen entfernt wird, überschrieben wird. Beispiel:

```
python3 checker.py -i test_bsp.txt -o ohne.txt
```

oder

```
python3 checker.py --input test_bsp.txt --output ohne.txt
```

2.4.4 Entfernen von Kopf- und Fußzeile und anschließendes schreiben in eine Datei mit Konsolenausgabe

Zum Entfernen von Kopf- und Fußzeile aus einer Datei, mit dabei den Text auf die Konsole auszugeben, tippen Sie einfach

```
python3 checker.py -i <DATEIPFAD> -o <AUSGABEDATEI> -v
```

oder

```
python3 convertToTxt.py --input <DATEIPFAD> --output <
   AUSGABEDATEI> -v
```

Es ist zu beachten, dass wenn die angegebene Ausgabedatei bereits existiert, der Inhalt durch den Text der im Moment durch Kopf- und Fußzeilen entfernt wird, überschrieben wird. Beispiel:

```
python3 checker.py -i test_bsp.txt -o ohne.txt -v
```

oder

```
python3 checker.py --input test_bsp.txt --output ohne.txt -v
```

2.4.5 Entfernen von Kopf- und Fußzeile und anschließendes schreiben von Text mit Wiederholungen in eine Datei ohne Konsolenausgabe

Zum Entfernen von Kopf- und Fußzeile aus einer Datei, ohne dabei den Text auf die Konsole auszugeben und Wiederholungen in die Ausgabedatei zu schreiben, tippen Sie einfach

```
python3 checker.py -i <DATEIPFAD> -o <AUSGABEDATEI> -r
```

oder

```
python3 convertToTxt.py --input <DATEIPFAD> --output <
   AUSGABEDATEI> -r
```

Es ist zu beachten, dass wenn die angegebene Ausgabedatei bereits existiert, der Inhalt durch den Text der im Moment durch Kopf- und Fußzeilen entfernt wird, überschrieben wird. Beispiel:

```
python3 checker.py -i test_bsp.txt -o ohne.txt -r
```

oder

```
python3 checker.py --input test_bsp.txt --output ohne.txt -r
```

2.4.6 Hilfe aufrufen

Zum Aufrufen der Hilfe einfach wie im folgenden Bild den Parameter -h bzw. --help eingeben.

```
python3 checker.py -h
```

oder

```
python3 checker.py --help
```

Die Ausgabe sollte wie folgt aussehen:

```
arguments
-h,                        --help                             show help message and exit
-i [path to file]          --input [path to file]             to run the program
-o [path to outputfile] --output [path to outputfile] to extract text into file
-r                                                            add repetitions at the end
-v                                                            verbose-Mode
```

2.4.7 Trennen von Sätzen

Zum Trennen von Sätzen tippen Sie einfach

```
python3 seperator.py -i <DATEIPFAD> -v
```

oder

```
python3 seperator.py --input <DATEIPFAD> -v
```

Als Beispiel sehen Sie im Folgenden wie die Sätze aus einer Textdatei getrennt werden:

```
python3 seperator.py -i test_bsp.txt -v
```

oder

```
python3 seperator.py --input test_bsp.txt -v
```

2.4.8 Trennen von Sätzen und anschließendes schreiben in eine Datei ohne Konsolenausgabe

Zum Trennen von Sätzen aus einer Datei, ohne dabei den Text auf die Konsole auszugeben, tippen Sie einfach

```
python3 seperator.py -i <DATEIPFAD> -o <AUSGABEDATEI>
```

oder

```
python3 seperator.py --input <DATEIPFAD> --output <
   AUSGABEDATEI>
```

Es ist zu beachten, dass wenn die angegebene Ausgabedatei bereits existiert, überschrieben wird. Beispiel:

```
python3 seperator.py -i test_bsp.txt -o out.txt
```

oder

```
python3 seperator.py --input test_bsp.txt --output out.txt
```

2.4.9 Trennen von Sätzen und anschließendes schreiben in eine Datei mit Konsolenausgabe

Zum Trennen von Sätzen aus einer Datei, mit dabei den Text auf die Konsole auszugeben, tippen Sie einfach

```
python3 seperator.py -i <DATEIPFAD> -o <AUSGABEDATEI> -v
```

oder

```
python3 seperator.py --input <DATEIPFAD> --output <
   AUSGABEDATEI> -v
```

Es ist zu beachten, dass wenn die angegebene Ausgabedatei bereits existiert, überschrieben wird. Beispiel:

```
python3 seperator.py -i test_bsp.txt -o out.txt -v
```

oder

```
python3 seperator.py --input test_bsp.txt --output out.txt -v
```

2.4.10 Hilfe aufrufen

Zum Aufrufen der Hilfe einfach wie im folgenden Bild den Parameter -h bzw. --help eingeben.

```
python3 seperator.py -h
```

oder

```
python3 seperator.py --help
```

Die Ausgabe sollte wie folgt aussehen:

```
arguments
-h,                        --help                              show help message and exit
-i [path to file]          --input  [path to file]             to run the program
-o [path to outputfile]    --output [path to outputfile]       to extract text into file
-v                                                             verbose-Mode
```

2.4.11 Erstellen der Satztrennungs-Erkennungs-Datei *.pickle

Zum Erstellen der Satztrennungs-Erkennungs-Datei tippen sie einfach:

```
python3 trainy.py -i <DATEIPFAD> -o <AUSGABEDATEI>
```

oder

```
python3 trainy.py --input <DATEIPFAD> --output <AUSGABEDATEI
   >
```

Es ist zu beachten, dass wenn die angegebene Ausgabedatei bereits existiert, überschrieben wird. Beispiel:

```
python3 trainy.py -i master_bp.txt -o master.pickle
```

oder

```
python3 trainy.py --input master_bp.txt --output master.pickle
```

2.4.12 Hilfe aufrufen

Zum Aufrufen der Hilfe einfach wie im folgenden Bild den Parameter -h bzw. --help eingeben.

```
python3 seperator.py -h
```

oder

```
python3 seperator.py --help
```

Die Ausgabe sollte wie folgt aussehen:

```
arguments
-h,                        --help                              show help message and exit
-i [path to file]          --input  [path to file]             to run the program
-o [path to outputfile]    --output [path to outputfile]       to extract text into file
-v                                                             verbose-Mode
```

3 Technischer Hintergrund

3.1 Verwendete Fremdsoftware

Datiename	verwendete Zusatzsoftware	Entwicklerwebseite
docTxt.py	catdoc	http://freecode.com/projects/catdoc
docxTxt.py	Python-Modul docx2txt	http://docx2txt.sourceforge.net/
pdfTxt.py	pdftotext	https://poppler.freedesktop.org/
odtTxt.py	odt2txt	https://github.com/dstosberg/odt2txt
rtfTxt.py	unrtf	https://www.gnu.org/software/unrtf/unrtf.html

Tabelle 6: Auflistung der verwendeten Software

3.2 Aufbau

Wie schon aus der Tabelle in Kapitel 2.3 zu sehen ist, besteht das Projekt aus einer Hauptdatei convertToTxt.py die aufgerufen wird, einer Hilfsdatei extractTxt.py in die die Programmlogik ausgelagert wurde und den einzelnen Modulen. Im weiteren werden die einzelnen Dateien technisch erläutert.

3.2.1 convertToTxt.py

Die convertToTxt.py nimmt alle Anfragen entgegen und beinhaltet die einzelnen Argumente sowie die Hilfe-Funktion. Die Verarbeitung der Argumente und Optionen wurden mittels dem Modul getopt realisert.

3.2.2 extractTxt.py

Die Datei extractTxt.py enthält die eigentliche Logik des Extrahierungs-Skriptes. Sie enthält die Funktionen process(path) und file(text, a). Die process(path)-Funktion nimmt den Pfad aus der zu extrahierenden Datei über den Parameter path entgegen und übergibt den Pfad dem entsprechenden Modul, indem es sich die Dateiendung betrachtet. Der heraus zu extrahierende Text, der von einzelnen Modulen zurückgegeben wird, wird mit UTF-8 dekodiert und so dann endgültig zurückgegeben. file(text, a) speichert den heraus extrahierten Text in eine Datei. Dazu wird dem Parameter text der zu speichernde Text und dem Parameter a der Speicherort übergeben.

3.2.3 Die Module

Wie im Abschnitt 3.1 in der Tabelle zu sehen ist, gibt es fünf Module. Alle Module rufen, bis auf docxTxt.py, ein Konsolen-Programm mittels subprocess.Popen() auf. Es wird dazu das Modul subprocess importiert. In einer Liste, die dem subprocess.Popen() übergeben wird, steht das zu ausführende Programm und der Dateiname. Außerdem wird die Standardausgabe und die Ausgabe für die Fehler in eine Pipe umgeleitet, damit

der zu extrahierte Text später weiterverarbeitet werden kann. Am Ende wird dann process.stdout.read() zurückgeliefert und mit process.stdout.close(), wird der Lese-Stream dann wieder geschlossen. Die Fehlerbehandlung wurde mittels try-except-Anweisung realisiert und es wird bei einer auftretenden Exception ins Logfile geschrieben.
Das Modul docxTxt.py unterscheidet sich von den anderen Modulen in sofern, dass ein Modul namens docx2txt importiert wird und dieses eigene Methoden zum Aufrufen besitzt. In docxTxt.py wird lediglich die Funktion process() verwendet. An die Funktion wird der Dateiname, aus der der Text extrahiert werden soll, übergeben und process() liefert dann einen String mit dem darin extrahierten Text zurück.

4 Kontaktdaten

Name	E-Mail
Mark Unger	mrk.unger@gmail.com
Siegfried Kienzle	siegfried.kienzle@gmx.de