## Aufgabe 1

### Analyse:

1. Das Dokument ist in utf-8 Kodierung, es enthält nicht-ASCII Symbole wie z.B. chinesische Hieroglyphen.
2. Außer Links enthält das Dokument diverse undefinierte Textausschnitte, welche vom Programm ignoriert werden sollen.
3. Unterschiedliche Linkformate:
   - Allgemeine Links [[link]] und Piped links [[link|text]]. Bei allgemeinen Links ist Zielartikelname gleich dem gelinkten Text.
   - „Internal links to an anchor“ ([[#See also]]), „internal links to a subpage“ ([[/example]]): Name der Seite kann nicht ermittelt werden, daher sollen diese Links ignoriert werden.
   - Links mit „Pipe trick“ [[link|]] sollen als allgemeine Links behandelt werden
4. Geschachtelte Links: nur das interne Link soll verarbeitet werden (Verhalten der Wikipedia).
5. Links mit \n:
   - Zielartikelnamen mit einem Zeilenumbruch sollen ignoriert werden.
   - Zeilenumbruch im gelinkten Text mit Leerzeichen ersetzen (Verhalten der Wikipedia).

### Programmbeschreibung

Das Programm analysiert die Eingabedatei("data/wikilinks_en.txt") symbolweise. Für das Finden von Anfang („[[„) und Ende („]]“) von Links wird ein vorheriges Symbol gespeichert. Wenn Linkanfang gefunden wurde, werden darauffolgende Symbole als Name der gelinkten Artikel interpretiert, bis ein Linkende bzw. Pipe („|“) gefunden wird. Falls ein Pipe gefunden wurde, werden darauffolgende Symbole als Linktext interpretiert, bis ein Linkende gefunden wird. Sonst wird wie in „Analyse“ beschrieben vorgegangen.
Die Ausgabe bestehend aus Paaren Zielartikelnamen und Linktext wird unter "data/Linktext.txt" gespeichert.
Für die Perfomanceanalyse wird Laufzeit des Programms angezeigt.

## Aufgabe 2.a

Die Dateien „data/Linktext.txt“ und „data/Linkstatistik.txt“ werden geöffnet. Die Zeilen im Linktext.txt werden analysiert: falls sie nicht im Dictionary vorkommen, wird ein neuer Eintrag mit dem Wert 1 erstellt, sonst wird der Wert vom existierenden Eintrag um eins erhöht.
Das entstandene Dictionary wird im Format „Häufigkeit Artikel Linktext“ (getrennt durch Tabulatorzeichen) in Linkstatistik.txt gespeichert.

## Aufgabe 2.b

Die Datei „data/Linkstatistik.txt“ wird geöffnet, dann werden Links mit gleichen Artikel aufsummiert und in einem Dictionary gespeichert.
Dictionary wird danach sortiert und Einträge mit mehr als 200 Vorkommen werden in die Datei „data/Over200.txt“ geschrieben.

## Aufgabe 3

Öffnen der Aufgabe im batch-modus: aufgabe_3.py Linkstatistik-file Eingabe-file Ausgabe-file
→ wie in der Aufgabenstellung beschrieben

Linkstatistik-file ist ohne Eingabe standartmäßig „data/Linkstatistik.txt“.
Wenn keine Argumente gegeben werden, startet die Aufgabe im Interaktiven-modus, bei dem mehrere Strings eingegeben werden können, jeweils durch Enter getrennt, bis ein Leerzeichen eingegeben wird, dann beendet sich die Eingabe und das Programm wird Fortgesetzt.
Eingaben werden mit Linkstatistik verglichen und es wird der Artikel mit dem jeweils höchsten Vorkommen ausgegeben.
outputfile_open() methode öffnet die Ausgabedatei, wenn mehr als 3 Argumente im Bash-aufruf angegeben wurden, ansonsten erfolgt die Ausgabe am Terminal.

## Aufgabe 4

Öffnen der Aufgabe im batch-modus: „aufgabe_4.py Linkstatistik-file Eingabe-file Ausgabe-file“
→ wie in der Aufgabenstellung beschrieben
Linkstatistik-file ist ohne Eingabe standartmäßig „data/Linkstatistik.txt“.
Die Ausgabe ist gemäß Aufgabestellung die Eingabetext mit Links.
Das Verhalten von outputfile_open() entspricht gleichnamiger Funktion aus Aufgabe 3.
createSearchWordDic(inputText, tagger) erstellt ein Dictionary. Dafür wird ein eingegebener Text gesplittet mit \n. Jede so entstandene line wird tokenized und getaggt. Danach wird geschaut, ob das Token ein „O“ ist. Wenn ja wird der Text einfach als „Text“, „O“ gespeichert. Wenn der Text nicht leer und das Token nicht „O“ ist, werden Text und Token im Dictionary „searchwords“ gespeichert (falls die Wörter mit dem gleichen Token nacheinander vorkommen, werden sie zusammengefasst und als ein Eintrag im Dictionary gespeichert). So entstandene Dictionary wird zurückgegeben.
createOutList(inputText, dictionaryText, tagger) erstellt aus Eingabetext und Ranking-Dictionary eine Ausgabeliste mit Texten und eventuellen Links. Dabei entspricht ein Listeeintrag einer Zeile.
AnalyseSenteses(inputList, Linkstatistikfilename) erstellt aus Eingabetext mit Hilfe von createSearchWordDic ein Dictitonary mit gerankten Begriffen und gibt mit Hilfe von createOutList eine Ausgabeliste mit Texten und eventuellen Links zurück.

## Aufgabe 5a

Das Programm öffnet in Aufgabe 2a entstandene Linkstatistik („data/Linkstatistik.txt“) und erstellt daraus „data/idf.csv“ und „data/tf.csv“. Die Dateien enthalten jeweils idf-Werte und tf-Werte für alle in Datei „Linkstatistik.txt“ vorkommende Terms bzw. Dokumente und Terms.
Jede Zeile in der Datei „idf.csv“ enthält ein Paar Term und idf-Wert, getrennt durch ein Tabulatorzeichen.
Jede Zeile in der Datei „tf.csv“ enthält am Anfang der Zeile einen Dokumentnamen gefolgt von Tabulatorzeichen und Paaren „Term und tf-Wert“ getrennt durch ein Tabulatorzeichen für alle in Dokument vorkommende Terms.
Alle Terms werden mithilfe der von uns geschriebener Funktion „stringNormalizer“ gemäß Aufgabenstellung normalisiert. Dabei benutzen wir die Funktion „safeStemmer“. Die Funktion „safeStemmer“ fängt in PorterStemmer geworfene Exception „IndexError“ ab (die entsteht, z.B., bei Eingabe von Wörter aed, aing, aeds), um die abnormale Programmterminierung zu vermeiden.

## Aufgabe 5b

Das Programm nimmt als erstes Parameter die Datei die tf-Werte in csv-Format enthält. Wenn kein Parameter eingegeben wurde, wird die Datei „data/tf.csv“ geöffnet.

Das Programm nimmt als zweites Parameter die Datei die idf-Werte in csv-Format enthält. Wenn kein Parameter eingegeben wurde, wird die Datei „data/idf.csv“ geöffnet.
Als drittes Parameter wird die Datei mit Suchanfrage erwartet. Falls der dritte Parameter nicht eingegeben wurde, wechselt das Programm in interaktives Modus und erwartet die Angabe der Suchanfrage vom Benutzer.
Als viertes Parameter wird der Name von Ausgabedatei übergeben. Wenn kein Parameter eingegeben wurde, erfolgt die Ausgabe auf der Konsole.
Das Programm erhält eine getIdf-Funktion, die aus einer csv-Datei idf lädt und als Dictionary zurückgibt.
Das Programm erhält eine getTf-Funktion, die aus einer csv-Datei tf lädt und als zweidimensionales Dictionary zurückgibt.
Die Funktion getDocWithMaxRating gibt das Dokument mit maximalem Ranking zurück. Diese Funktion benutzt die Funktion getRatingList, die ein Dictionary mit Paaren „Dokumentnamen und Ranking“ zurückgibt.
Das Programm verwendet die Funktion stringNormalizer aus der Aufgabe 5a.
Bei der Berechnung von idf-Wert für die Suchanfrage gehen wir davon aus, dass die Anzahl von Dokumenten im Vergleich zur vorherigen Aufgabe um eins erhöht hat (die Suchanfrage selbst).

## Aufgabe 6

Das Programm nimmt als erstes Parameter die Statistik Datei. Falls kein Parameter eingegeben wurde, öffnet das Programm die Datei data/Linkstatistik.txt.
Das Programm nimmt als zweites Parameter die Datei die tf-Werte in csv-Format enthält. Wenn kein Parameter eingegeben wurde, wird die Datei „data/tf.csv“ geöffnet.
Das Programm nimmt als drittes Parameter die Datei die idf-Werte in csv-Format enthält. Wenn kein Parameter eingegeben wurde, wird die Datei „data/idf.csv“ geöffnet.
Als viertes Parameter wird die Datei mit Suchanfrage erwartet. Falls der dritte Parameter nicht eingegeben wurde, wechselt das Programm in interaktives Modus und erwartet die Angabe der Suchanfrage vom Benutzer.
Im Vergleich zu Aufgabe 4, hat die Benutzung von Rankingalgorithmus dazu geführt, dass die vorgeschlagene Links kontextabhängig sind, z.B., dem Linktext „Donald Trump“ wird der Artikel über seine Präsidentschaftskampagne zugewiesen, und nicht der Artikel über ihn selbst.

## Aufgabe 7

Das Programm implementiert eine Hilfsfunktion removeStopwordFromList, die dazu dienen soll, „Stoppwörter“ aus der Liste zu entfernen. Stoppwörter sind die Wörter, die keine Eigenbedeutung haben (z.B., Artikel, Präposition, Pronomen usw). Diese Wörter werden von allen großen Suchmaschinen, z.B. Google, von Suchanfragen entfernt, um Suchqualität zu verbessern. Ein weiterer Vorteil in unserem Fall wäre die Verringerung von Suchdateien (Linkstatistik.txt, idf.csv, tf.csv).