Valerie Claessen (2179913)
Projektseminar Angewandte Informationswissenschaft

Projektplan

Bereich	Lösung	Erläuterung	Material, Ideen, Bibliotheken
Thema	Tippfehlerkorrektur/Autokorrektur	Tippfehler in freien Texten sollen erkannt und korrigiert werden, die korrekten Wörter müssen dazu im Vorhinein bereits vorliegen/ eingelesen werden damit im Nachhinein festgestellt werden kann, wie erfolgreich die Korrektur war. Es soll ein Vergleich zwischen verschiedenen Algorithmen sattfinden.	
Benötigte Daten	Lexikon	Eigenes kleines Lexikon, dieses ist dann erweiterbar Alternative: Es muss ein Lexikon vorhanden sein, auf das zugegriffen werden kann, um die falschen Wörter zu erkennen und sie durch das (am wahrscheinlichsten) richtige Wort zu ersetzten	Methode: Lexikon in das Programm einlesen Lexikon liegt in Form einer .txt-Datei vor https://wordnet.princeton.edu/ http://elexicon.wustl.edu/ oder mittels PyEnchant Wörter als falsch erkennen lassen
	Korrekte Texte (zu Anfang sollen die Trainingstexte eher Kurz sein, also ca.100 – 200 Wörter lang)	Es muss ein korrekter Ausgangstext ohne Fehler vorliegen. Dieser wird benötigt um den Erfolg der Algorithmen zu berechnen ➔ Wie viele Fehler wurden erkannt ➔ Wie viele der fehlerhaften Wörter wurden richtig korrigiert Hierbei kann auf die Tabellen (Gold Labels) von Naive Bayes zurückgegriffen werden	Methode: Wortabgleich Textbeispiel: http://www.writerswrite.com/books/excerpts/inkheart-excerpt-400014
	Fehlerhafte Texte	Der richtig vorliegende Text muss intellektuell so verändert werden, dass Fehler enthalten sind. Hierbei	Methode: Fehlerhafte Wörter erkennen Frage: Liegt mein Wort in Lexikon vor ➔ Ja: Wort ist richtig

		muss beachtet werden, dass es verschiedene Arten von Fehlern gibt (die auch besonders Häufig sind) - Buchstabe fehlt - Buchstabe zu viel - Buchstaben verdreht - Leerzeichen falsch (zu viel oder zu wenig) - Andere Fehler	➔ Nein: Berechnung des „richtigen“ Wortes Methode: Fehlerhafte Wörter ersetzen ➔ Gebe dem Nutzer das „richtige“ Wort
Methoden → Diese können im Programm umgesetzt werden	N-Grams (z.B. Henrichs's Pentagram Index)	- Es wird kontrolliert, wie viele n-Gramme (z.B. Trigramme) von Wörtern miteinander übereinstimmen	Methode: checkNGrams
	Acquaintance: N-Grams in the Vector Space, Euklidische Distanz oder Cosinusähnlichkeit	- Ähnlichkeit von Vektoren (Ein Wort wird in Form von einem Vektor repräsentiert) - Distanz zwischen zwei Vektoren	Methode: checkVectorSpace checkVectorDistance
	Edit distance	- Misst wie unähnlich sich zwei Strings sind ➔ Es wir die Anzahl an Operationen gemessen, die benötigt wird, um den falschen String in den richtigen zu verwandeln	Methode: checkEditDistance Beispiel: http://stackoverflow.com/questions/13928155/spell-checker-for-python http://norvig.com/spell-correct.html
	Damerau–Levenshtein distance	- Misst die Anzahl der Veränderungen um einen String in einen anderen (den richtigen) zu verwandeln	Methode: checkDLDistance Library: https://pypi.python.org/pypi/pyxDamerauLevenshtein/1.4.1
	Bayes Theorem	- Wir versuchen die Ersetzung mit der höchsten Wahrscheinlichkeit (gemeint zu sein) zu finden	Methode: checkBayes $\text{argmax}_{c \in candidates} P(c \mid w)$ $\text{argmax}_{c \in candidates} P(c)\, P(w \mid c) / P(w)$
Aufgaben	Textbearbeitung	Text: - Einlesen	Beispiel:

		- Säubern - In Wörter zertrennen - In Teile Zertrennen, mit denen später gearbeitet werden kann (N-Grams, Buchstaben o.ä.)	> import string > clean = [] > with open('book.txt') as f: // .txt einlesen > data = f.readlines() > for element in data: > s = element.translate(None, string.punctuation) // Punktation entfernen > clean.append(s)
	Linguistische Methoden anwenden	Die implementierten Methoden sollen auf einen fehlerhaften Text angewendet werden. - Fehlererkennung - Fehlerkorrektur - Darstellung der Ergebnisse (Gesamt oder einzelne Wörter mit ihrer Korrektur)	Verschiedene Algorithmen, die einzeln für einen Fehler das „richtige“ Wort berechnen
	Ausgabe erstellen	- Wie viele Worte sind Falsch - Wie sehen sie (wahrscheinlich) richtig aus - Dies soll in einer kleinen GUI dargestellt werden	Darstellung jeweils gesplittet zwischen den einzelnen Methoden Vergleich zwischen den Algorithmen: - War das Wort wirklich richtig - Darstellung aller Wortvorschläge in GUI Ausgabe
	Eigene Wörter dem Lexikon hinzufügen	- Button in der Haupt-GUI, die in zweites Fenster leitet → hier ist das Hinzufügen eigener Wörter möglich	<u>Textfeld:</u> Hier kann ein neuer Lexikoneintrag hineingeschrieben werden <u>Hinzufügen – Button:</u> Hiermit wird das Wort im Textfeld in das Textdokument hinzugefügt, indem sich das bisherige Lexikon befindet
	Methoden Vergleichen/ Analysieren	Welche der Methoden ist wie erfolgreich, wie viele Wörter werden richtig korrigiert. Dies kann in Form von Graphen dargestellt werden Analyse: Werden die Methoden heute noch genutzt, ist es sinnvoll sie anzuwenden?	z.B.: allgemeiner Vergleich

	Visualisierung	z.B. mit Matplotlib direkt mit ausgeben - als fasch erkannte Wörter - Wörter die richtig als falsch erkannt wurden → Gold Labels - Vergleich der Methoden z.B. als Balkendiagramme	Library: Matplotlib
Technische Details	Programmiersprache	Python	Anaconda
	Visualisierung	Matplotlib für Graphen etc.	Library: Matplotlib
	Präsentation der Ergebnisse/ Arbeitsoberfläche	Konsole und GUI	Library: Tkinter

Valerie Claessen (2179913)
Projektseminar Angewandte Informationswissenschaft