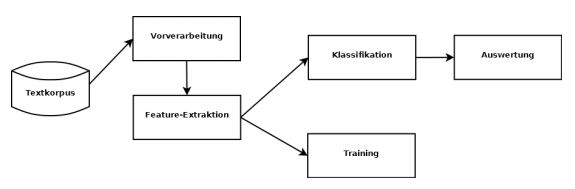
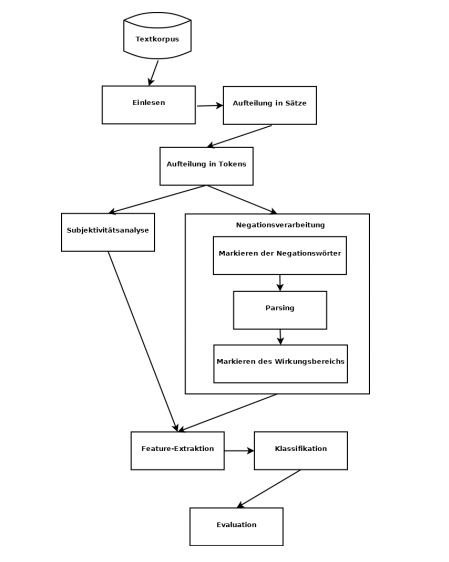
Bachelorarbeit Zusammenfassung

**1. Ablauf von Sentiment-Analysis**

****

**2. Detailierte Beschreibungen**

****

**a. Textkorpus:** Sammlung von Daten -> Trainingsdaten und Daten zur Validierung entnehmen -> bessere Analyse auf bestimmter Domäne. Insgesamt enthält der Korpus hier 1000 positive und 1000 negative Dokumente, die von einer Internet-Reviewdatenbank ausgelesen und eingeordnet wurden.

**b. Einlesen der Dokumente mit** *Collection Reader*

**c. Aufteilung der Dokumente in Sätze**

\_ Nachdem das Dokument in Sätze aufgeteilt wurde, werden diese weiter zerlegt. Dazu wird jedes Wort mit einer Token-Annotation versehen. Diese Aufgabe wird von einer OpenNLPKlasse übernommen, die über die ClearTK-Bibliothek als UIMA-Komponente zugänglich ist.

**d. Vorverarbeitung**

* *Subjektivitätsanalyse*:

|  |  |  |  |
| --- | --- | --- | --- |
| **Ansatz** | **Art der Tests** | **Bewertung** | **Mittel** |
| Naives Bayes Klassifikator | * Unigramme * Bigramme * Trigramme * Unigramme, Bigramme und Trigramme * Präsenz von Wortarten * n-Gramme und sem. Orientierung * n-Gramme, sem. Orientierung und Wortarten | Liefert die beste Ergebnisse in Precision-, Recall- und F-Score | Mallet (ClearTK) |
| Support-Vektor Maschinen | Genauso wie bei Naives-Bayes-Klassifikator | Fast genau so gut wie Naives-Bayes-Klassifikator | SVM Light (ClearTK) |

* *Negationsverarbeitung:*

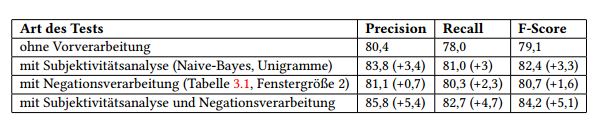
\_ Negationswörterliste erstellen

|  |  |  |  |
| --- | --- | --- | --- |
| hardly | lack | lacking | lacks |
| neither | nor | never | no |
| nobody | none | nothing | nowhere |
| not | n’t | ain‘t | can’t |
| cannot | daren’t | don’t | doesn’t |
| didn’t | hadn’t | hasn’t | haven t |
| haven’t | isn’t | mightn’t | mustn’t |
| needn’t | oughtn't | shan’t | shouldn’t |
| wasn’t | wouldn’t | without |  |

\_Negationswörter und deren Bewirkungsbereich finden und markieren

|  |  |  |  |
| --- | --- | --- | --- |
| **Ansatz** | **Art der Tests** | **Bewertung** | **Mittel** |
| Feste Fenstergröße | Wörter von der obenstehende Liste finden und den Wirkungsbereich davon durch Fenstergrößen von 1-5 bestimmen | bei allen Score liegt der Durchschnitt bei 80%. Es gibt hier fast kaum Unterschied zwischen den Größen des Fensters. |  |
| Parsing | verkleinern des Wirkungsbereich durch mehrere selbstdefinierte Regel:  - vorläufiger Wirkungsbereich  - Begrenzer  - Begrenzer und Ausnahm | genau so gut wie beim Ansatz mit Fenstergröße |  |

* **Auswirkung der Vorverarbeitungsschritte auf die Dokumentenklassifikation**

****

**c. Feature-Extraktion**

\_ Erstellung eines Featurevektors F = (f1, f2, ..., fn) aus einem Dokument. Die Wahl der Features ist ein wichtiger Teil der Textverarbeitung und kann das Ergebnis stark beeinflussen, ist abhängig von der Fragestellung und der Domäne des Textes. → keinen einheitlichen Algorithmus, nur mit Intuition, Wissen über die Domäne und Experimentieren.

* **N-Gramme**: Unigramme (n = 1): 1 Feature – 1 Wort, Bigramme (n = 2): Wortpaaren von zwei aufeinanderfolgenden Wörtern. Satzanfang und Satzende können hierbei durch n-Gramme dargestellt werden. Je größer n ist, desto unwarscheinlicher, n-Gramme, ein weiteres Vorkommen in den Texten zu finden. → Die Wahl des Wertes für n beeinflusst also die Ergebnisse.

\_ Für spezielle Anforderungen kann es sich positiv auswirken, nur bestimmte Wortarten als Features zu verwenden. (z.B: Adjektiven bei subjektiver Analyse).

\_ Dafür gibt es auf der ClearTK Webseite ein Tutorial.

**d. Klassifikation**

* **Naive-Bayes-Klassifikator**: Der Klassifikator versucht, die **Wahrscheinlichkeit** für die Zugehörigkeit eines Textes zu jeder der Klassen zu bestimmen → die Zuordnung mit höchster Warscheinlichkeit wird ausgewählt. Die Wahrscheinlichkeit, dass ein Text mit dem Featurevektor x in die Klasse C gehört, wird durch den Ausdruck P(C|x) dargestellt. P(Ci |x) = P(Ci)p(x|Ci) P(x)
* **Support-Vektor-Maschinen:** Support-Vektor-Maschinen versuchen anhand der Trainingsdaten, eine Funktion zu erlernen, die die Klassen möglichst gut voneinander trennt.Feature-Vektor wird als ein Punkt dargestellt. Jede Funktion ist eineTrennebene. Hier wird eine optimale Ebene gesucht. Support-Vektoren sind die Punkte, die den Mindestabstand von der Ebene haben.

**e. Training**

\_ wird per Hand durchgeführt. Ablauf: Vorverarbeitung → Feature-Extraktion → Featurevektoren und Klasseneinordnung .

**f. Bewertung**

\_ Es wird hier bewertet, wie gut das Verfahren arbeitet. Es wird gezählt, wie viele Dokumente das trainierte Verfahren richtig und wie viele falsch zuordnet.

\_ Es gibt hier 4 Kategorien für bereits sortierte Dokumente: „wahres Positiv“ ,wahres Negativ, falsch Positiv, falsch Negativ.

\_ Zum Bewerten gibt es:

* Precision: precision = wp/( wp + f p ). Ein Precision-Wert von 100% zeigt an, dass alle Zuordnungen von Dokumenten zu der positiven Klasse korrekt sind. Zuordnungen zu anderen Klassen werden nicht betrachtet.
* Recall: recall = wp/( wp + fn )
* F-Score: F = 2 ∗ precision ∗ recall /(precision + recall )