Projektseminar Angewandte Informationswissenschaft – Projektplan

Name: Sabrina Wirkner Matkrikelnummer: 2183349 Github: SabrinaWirkner

Thema: Sentiment Analysis mit Twitterdaten

Bachelorarbeit: Sentiment-Analyse mit ereignisspezifischen Twitterdaten – Anwendung von Methoden des Machine Learnings zum Aufbau einer Klassifikation

---

Projekt für das Projektseminar: Bau einer Klassifikation zur Sentiment Analysis mit Twitterdaten

Ziel: Bau einer Klassifikation nach Naive Bayes und MaxEnt

Vergleich der Ergebnisse mit und ohne Hilfe eines Sentiment-Lexikons

Vergleich der Ergebnisse zu den Twitterdatensätzen (vorher, währenddessen, nachher)

Am Ende des Projektes möchte eine möglichst gut funktionierende Klassifikation implementiert haben und Ergebnisse erzielen, aus denen hervorgeht, ob es sinnvoll ist ein Lexikon nur aus den 2000 Tweets zu aufzubauen oder ob das Nutzen eines vorhandenes Sentiment-Lexikons effektiver ist, um zu wissen, wie ich in der Bachelorarbeit am besten vorgehe.

Außerdem möchte ich sehen, welche Methoden die besten Ergebnisse erzielen und welche Probleme es gab bzw. ob diese direkt im Projekt gelöst worden konnte.

Als letztes möchte ich noch einen Einblick gewinnen, wie das Verhältnis zwischen den Datensätzen ist (vorher, währenddessen, nachher) und ob dieses Verhältnis beim automatischen Klassifizieren bestehen bleibt.

Material:

- Sentiment-Lexikon (positiv, neutral, negativ) mit Gewichten
  - Entweder „Subjectivity Clues Lexicon“ (8000 Wörter) oder „Opinion Lexicon” (6900 Wörter)
- Twitterdaten: ~2000 Tweets zum Thema San Diego Comic-Con 2016 (englisch)
  - Vorher 19. - 20.07.2016 ~450 Tweets
  - Währenddessen 21. - 24.07.2016 ~800 Tweets
  - Nachher 25. - 26.07.2016 ~700 Tweets
    - ➔ Manuell klassifiziert nach positiv, neutral und negativ
- Stoppwortliste

Methode: Naive Bayes (Basis: Bag-of-Words)

Maximum Entropy Model

Support Vector Machine (k-nearest neighbour)

Pointwise Mutual Information

➔ Selbst implementiert

Programmiersprache: Python

Libraries: Matplotlib, evtl. NLTK

Auswertung / Vergleich:

- Vergleich mit den manuell klassifizierten Datensätzen (gold labels)
- Auswertung der Ergebnisse mit Precision, Recall & Accuracy
- Visualisierung: Vergleich der Twitterdatensätze mit Diagrammen (Matplotlib)

Vorgehen:

1. Tweets manuell klassifizieren in CSV (Listenauswahlfeld: positiv, neutral, negativ)
2. Naive Bayes

   Bayes Theorem:

   $P(A|B) = (P(B|A) * P(A))/ P(B)$

   P(A|B) = Wahrscheinlichkeit von A (Klasse (positiv, neutral, negativ) gegeben B (Tweet)

   P(B|A) = Wahrscheinlichkeit von B (Tweet) gegeben A (Klasse)

   P(A) = Wahrscheinlichkeit von A (Klasse), unabhängig

   P(B) = Wahrscheinlichkeit von B (Tweet), unabhängig

   → konstant; kann ignoriert werden

   Also: P(Klasse|Tweet) = P(Tweet|Klasse) * P(Klasse)

   P(Tweet|Klasse) ist das Produkt aller Wahrscheinlichkeiten der Wörter des Tweets (w1, w2, …) in der Klasse zu sein

   Also: $P(Tweet|Klasse) = P(w_1|Klasse) * P(w_2|Klasse) * … * P(w_n|Klasse)$

   $P(w_1|Klasse)$ = w1 in positiven Tweets / Anzahl w in positiven Tweets

   P(Klasse) = 1/3 für jede der drei Klassen (positiv, neutral, negativ)

Laplace-Smoothing

Um ungesehene w und Klasse Kombinationen abzudecken

P(w1|Klasse) = w1 in positiven Tweets + 1 | Anzahl w in positiven Tweets + Vokabular

Ergebnisse abgleichen mit Ergebnissen, die man bekäme, wenn man das Sentiment-Lexikon verwendet

Funktionen:

NLTK: CSV Dateien einlesen, Stemming, mit Stoppwortliste abgleichen, Stoppworte löschen, Satzzeichen löschen, alles in Kleinbuchstaben (für alle Methoden)

Anzahl der Wörter in positiven, neutralen und negativen Tweets zählen lassen

Für jedes Wort P(w|Klasse) berechnen und Gewichte speichern → Eigenes Sentiment-Lexikon

Funktion, um P(Klasse|Tweet) für jeden Tweet und alle 3 Klassen zu berechnen

Für jeden Tweet P(Klasse|Tweet) mithilfe des eigenen Lexikons berechnen (für positive und negative Klasse)

Werte für die drei Klassen vergleichen → maximales P suchen

Berechnetes Sentiment in neue CSV-Datei schreiben

Funktion, um P(Klasse|Tweet) für jeden Tweet und alle 3 Klassen mithilfe des Sentiment-Lexikons zu berechnen

P(Klasse|Tweet) = Gewicht aus Lexikon für Wort * 1/3

Werte für die drei Klassen vergleichen → maximales P suchen

Berechnetes Sentiment in neue CSV-Datei schreiben

3. Maximum Entropy Model

   Gewichte aus Naive Bayes Schritt oder Sentiment-Lexikon als Features (Indicator Functions) verwenden

   P(Klasse|Tweet) berechnen nach MaxEnt

   Funktionen:

   - Funktion, um P(Klasse|Tweet) zu berechnen
     - Gewichtete Feature Summe für jede Klasse und jeden Tweet berechnen
     - Für jede Klasse auf MaxEnt-Formel anwenden
     - Maximales P suchen
     - Berechnetes Sentiment in neue CSV-Datei schreiben
   - Funktion, um dasselbe mit Sentiment-Lexikon zu berechnen

4. Support Vector Machine – k-nearest neighbour

   Für alle Tweets der originalen CSV-Datei Vektoren erstellen (jeder Wert ist die Anzahl eines Wortes des gesamten Vokabulars)

   Trainieren mit denselben Daten, indem von jedem Tweet die Ähnlichkeit seines Vektors zu den anderen Tweets berechnet wird (normalized dot product) und dann der k=3 oder k=5 k-nearest-neighbour berechnet und so die am besten geeignete Klasse (positiv, neutral, negativ) zugeordnet wird.

   Funktionen:

   - Funktion zum Berechnen der Vektoren
   - Funktion zum Vergleichen der Vektoren
     - Ähnlichste Vektoren ermitteln, Klasse anhand von k zuweisen

5. Pointwise Mutual Information

   Wort-Wort-Matrix berechnen vom gesamten Vokabular

   P(w), P(c) und P(w|c) berechnen anhand der Matrix und auf PMI-Formel anwenden → neue Gewichte

   Auf Naive-Bayes-Formel anwenden → größtes P suchen

   Funktionen:

   - Funktion zum Berechnen der Wort-Wort-Matrix

Funktion zum Berechnen von P(w), P(c) und P(w|c)

Auf PMI-Formel anwenden → neues Gewichte-Lexikon

Funktion zum Zuordnen der Klasse

Gewichte auf Naive Bayes Formel anwenden → größtes P suchen

6. Auswertung

Manuell klassifiziertes Sentiment mit automatisch berechnetem Sentiment abgleichen

➔ Gold labels
➔ Precision, Recall & Accuracy berechnen

Funktionen:

CSV-Dateien einlesen, Zellen der Matrix für die Gold Labels berechnen (positiv-positiv, positiv-negativ, positiv-neutral, …)

Funktion zur Berechnung von Precision, Recall & Accuracy

Matrix als Tabelle ausgeben für Naive Bayes und MaxEnt Ergebnisse + Precision, Recall & Accuracy (jeweils mit und ohne Verwendung des Sentiment-Lexikon)

7. Visualisierung

Funktionen:

Anzahl positiver, neutraler und negativer Tweets (intellektuell & automatisch klassifiziert nach Naive Bayes und MaxEnt) als Kreisdiagramm darstellen für jeweils vorher, währenddessen und nachher

---

Aufbau:

1. NLTK auf Testdaten anwenden

2. Austauschbar: Naive Bayes-, MEM-, SVM- oder PMI-Methode

3. Auswertung

(4. Optimierung der Methoden (Welche Probleme gab es? Können diese gelöst werden? Wie kann die Perfomance verbessert werden?) → Nur, wenn noch Zeit ist)

5. Visualisierung

Probleme:

- Verneinung
  - Lösung: „not“ und „no“ berücksichtigen und Gewicht umkehren
- Neutrale Klasse
  - Viel mehr Wörter zur neutralen Klasse
  - Lösung: Nur positive & negative Klasse berechnen, neutrale Klasse durch Nichterreichen des Schwellenwerts zuordnen