IMDB Reviews Sentiment Analysis

Fabio Anderegg
Michel Hosmann
Casimir Platzer

Inhalt

- Einleitung
- Versuche
  - Baseline & Gewichteter Baseline
  - Naive Bayes
  - Neuronales Netzwerk
- Vergleich mit Python
- Fragen

Ungewichteter Baseline

Stopwords	Stem Dataset	Stem List	Accuracy
Yes	No	No	69.85
No	No	No	69.85
Yes	Yes	Yes	68.95
No	Yes	No	66.55
Yes	Yes	No	65.8
Yes	No	Yes	63.15
No	Yes	Yes	63.05
No	No	Yes	57.6

Gewichteter Baseline

- SentiWordNet & WordNet 3.0
- Jedes Wort eine positive & negative Bewertung
- Worte in ‘Synsets’, jedes Wort hat mehrere Bedeutungen
  - Beispiel: ‘have’
    - (Positiv) have or possess, either in a concrete or an abstract sense; "She has $1,000 in the bank"; "He has got two beautiful daughters"; "She holds a Master's degree from Harvard"
    - (Neutral) have as a feature; "This restaurant features the most famous chefs in France"
- Schlussfolgerung: Gewichteter Baseline unzuverlässig
  - ‘Wortsense’ nicht bekannt aber wichtig

Quelle: http://sentiwordnet.isti.cnr.it/

Gewichteter Baseline: Ergebnisse

Ergebnisse

- Ohne Stopwords: ~54%
- Mit Stopwords: ~64%
- Ohne Stopwords & ‘erster’ Sense: ~62%

Naive Bayes

- 10 Zeilen relativ simpler Java Code
- Diverse Anpassungen der Voreinstellungen haben keine Verbesserung bewirkt
- Ergebnis: 81.25 %

Neuronales Netzwerk

- Angepasstes Beispiel aus dem deeplearning4j Projekt
- Recurrent neural network mit LSTM units
- Erster Versuch: Dataset aus Beispiel ersetzt mit unserem Dataset:

Accuracy:	0.5325
Precision:	0.7251
Recall:	0.5325

- Ziemlich schlechtes Ergebnis für eine Laufdauer von 5 Minuten
- Idee: Dataset auf dem Beispiel zum lernen verwenden (50’000 IMDB reviews)

Neuronales Netzwerk mit grossem Lern-Dataset

- Ergebnis:

Precision: 0.7323
Recall: 0.7300

Neuronales Netzwerk mit grossem Lern-Dataset

- Ergebnis:

Precision: 0.7323
Recall: 0.7300

- Ergebnis mit auf 1000 Worte begrenzte Reviews (statt 256 Worte):

Precision: 0.8531
Recall: 0.8530

- Fazit: Neuronale Netzwerke brauchen ein grosses Dataset zum Lernen

Vergleich mit Python

Normalisierung:

```
"""
This is where a lot of the magic happens. For every corpus passed to this function, it will (in this order):
- normalize accented characters
- unescape and remove html
- expand contractions according to contractions.py
- if set: lemmatize according to nltk pos-tag
- else: lowercase
- remove special characters (string.punctuation)
- remove stopwords (nltk.corpus.stopwords plus some additions)
- if set: remove all non-text characters
- if set: as tokens (words)
- else: not tokenized
```

Vergleich mit Python

Training Set:

http://ai.stanford.edu/~amaas/data/sentiment/

Feature Extraction:

- CountVectorizer (binary or by frequency)
- TfidfVectorizer (term frequency–inverse document frequency)

Vergleich mit Python

Classifier	Accuracy	Precision	Recall
SGDClassifier	0,86	0,86	0,88
MultinomialNB	0,81	0,81	0,81
BernoulliNB	0,82	0,78	0,89
LogisticRegression	0,87	0,86	0,87
LinearSVC	0,86	0,86	0,85
NuSVC	0,82	0,79	0,84

Fragen?