DATEN

Beschreibung:

- Die Daten bestehen aus 25.000 englischen Film Reviews.
- Reviews stammen von der Seite IMDb
- Entspricht einem zufällig gewählten halben Datensatz von ai.stanford
- Er wurde im Juni 2011 von Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng und Christopher Potts erhoben.

Features:

- Es gibt die Klasse Text in Form eines Strings, sie beinhaltet den Text der Review.
- Die Zielklasse ist die Bewertung: positiv (pos) oder negativ (neg)

➔ Es soll also anhand des Textes der Review vorhergesagt werden, ob die Review positiv oder negativ ist.

- Mithilfe von Java haben wir weitere Features hinzugefügt, um so die Vorhersagesicherheit zu erhöhen (siehe Merkmalauswahl)

Trainingsprozess:

- Datensatz besteht aus 25.000 einzelnen Reviews
- Daraus wurden randomisierte Trainings-, Entwicklungs-Testdaten generiert
- Trainingsdaten: 20.250 Reviews
- Entwicklungsdaten: 2.250 Reviews
- Testdaten: 2.500 Reviews

Algorithmus:

Wir haben uns auf einen Entscheidungsbaum (J48 und teilweise Random Forest) konzentriert, jedoch auch einen NaiveBayesMultinomialText angewendet. Dazu in der Abschlussevaluation mehr.
Wir haben unsere eigene Software in IntelliJ geschrieben, diese ist auch angehängt. Man kann mit dem Package „Classifier“ einen Baum trainieren, testen und abspeichern. Mit einer anderen Klasse kann man diesen Baum laden und auf neue Daten anwenden, diese müssen jedoch die gleiche Form haben, wie die zum Training benutzten Daten.
(Details zur Software in der README.md Datei)
GITHUB: https://github.com/SirJonasM/KiReviewClassifier.git

Merkmalauswahl:

Wir haben uns dazu entschieden die Anzahl der Satzzeichen und die Länge des Textes mit einzubeziehen. Mit dem Package PreProcessing kann man eine „.arff“ Datei vorverarbeiten um diese weiteren Features als Attribute zu erhalten.

Vorgehensweise

Wir hatten die Idee, dass man, um eine Review in negativ oder positiv einteilen zu können nur bestimmte Wörter des Textes benötigt. Diese Wörter sind entweder positiv oder negativ konnotiert. Im Internet haben wir Listen mit positiven englischen Wörtern (circa 2000) und negativen Wörtern (circa 4812) gefunden. Diese beinhalten auch die konjugierten Wortformen.

- positiv: https://gist.github.com/mkulakowski2/4289437
- negativ: https://gist.github.com/mkulakowski2/4289441

Zu diesen Wörtern haben wir noch ein paar hinzugefügt, z.B. „*“, „**“,… ,“1/10“, „2/10“,…, „10/10“, „1/5“, „2/5“, …, „5/5“ und noch andere.

Jedoch muss man auch die verneinten Wörter beachten. In unserer Implementation behalten wir das Wort not und schieben es mit dem nächsten Wort aus einer der Listen zusammen. Aus: „not […] good“ wird: „notgood“. Will man diese Einstellung nicht, kann man einen AlphabeticTokenizer für den StringToWordVector nutzen.

Mit einem StringToWordVector Filter und AttributeSelection werden diese Texte dann weiterverarbeitet. Es gibt eigene Methoden zum Konfigurieren der Filter:

- configureStringToWordVector()
- configureAttributeSelection()

Nach vor Verarbeitung der Daten gibt es ungefähr 8000 verschiedene Attribute, die der StringToWordVector findet, demnach ungefähr 8000 verschiedene Wörter.

Man sollte den Threshold des AttributeSelection Filter auf 0.0 setzen und auch nicht zu wenige Attribute im Vergleich zu der Anzahl der Attribute des StringToWordVector Filter zulassen (bei der Nutzung eines InfoGainAttributeEval()). Wählt man z.B., dass der StringToWordVector 1000 Wörter auswählt, sind darunter auch sehr seltene Wörter, diese können aber auch sehr Ausdrucks stark sein und bekommen damit einen hohen Wert bei der Attribute Selection. Wenn man dabei also wenige Wörter auswählen lässt, kann es sein, dass nur diese seltenen Wörter ausgewählt werden. Da diese selten sind kann es passieren, dass eine Review keine dieser Wörter enthält und somit der Baum keine sinnvolle Entscheidung treffen kann, für diesen ist es dann quasi 50/50.

Als nächstes hatten wir die Idee, die Anzahl der negativen und der positiven Wörter zu zählen und als Attribut zu nutzen und zudem eine Informationsdichte einzuführen, damit ist gemeint, wie viele relevante Wörter im Vergleich zur insgesamten Text Länge enthalten sind.

Tests

Zum Trainieren nutzten wir die Trainingsdaten, zum Testen die Entwicklungsdaten (Devdata)

Wir setzen die Anzahl der Wörter zu behalten nach dem StringToWordVector Filter auf wordsToKeep = 500 und die zu selektierenden Wörter des AttributeSelection Filters auf numToSelect = 250.

Nach Erstellen eines Entscheidungsbaums, sind…

… Anzahl negativer Wörter,
… Anzahl positiver Wörter,
… Anzahl Satzzeichen,
… Länge des Textes

im Baum zu finden. Die Informationsdichte wird jedoch nicht berücksichtigt. Das ist ein Indiz dafür, dass die Informationsdichte vermutlich kein gut gewähltes Attribut ist.

Danach testeten wir, ob es einen positiven Einfluss hat, die negativen/positiven Wörter zu zählen:

```
=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0,810    0,192    0,796      0,810    0,803      0,618    0,834     0,776     pos
                0,808    0,190    0,822      0,808    0,815      0,618    0,834     0,800     neg
Weighted Avg.   0,809    0,191    0,810      0,809    0,809      0,618    0,834     0,788
```

---- Mit Anzahl positiv/negativ

```
=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0,795    0,173    0,810      0,795    0,802      0,622    0,854     0,804     pos
                0,827    0,205    0,813      0,827    0,820      0,622    0,854     0,830     neg
Weighted Avg.   0,812    0,190    0,812      0,812    0,811      0,622    0,854     0,818
```

---- Ohne Anzahl positiv/negativ

➔ Es hat demnach einen negativen, wenn auch sehr geringen Einfluss.

Aus diesem Grund werden wir diese Attribute nicht mehr weiter berücksichtigen.

Wir setzen nun wordsToKeep auf 500 und variieren numToSelect:

80:

```
=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0,831    0,251    0,753      0,831    0,790      0,580    0,840     0,779     pos
                0,749    0,169    0,827      0,749    0,786      0,580    0,840     0,833     neg
Weighted Avg.   0,788    0,209    0,792      0,788    0,788      0,580    0,840     0,807
```

100:

```
=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,827    0,246    0,756      0,827    0,790      0,581    0,835     0,773     pos
               0,754    0,173    0,825      0,754    0,788      0,581    0,835     0,824     neg
Weighted Avg.  0,789    0,208    0,792      0,789    0,789      0,581    0,835     0,800
```

150:

```
=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,836    0,230    0,771      0,836    0,802      0,606    0,843     0,792     pos
               0,770    0,164    0,836      0,770    0,801      0,606    0,843     0,819     neg
Weighted Avg.  0,802    0,196    0,804      0,802    0,802      0,606    0,843     0,806
```

250:

```
=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,795    0,173    0,810      0,795    0,802      0,622    0,854     0,804     pos
               0,827    0,205    0,813      0,827    0,820      0,622    0,854     0,830     neg
Weighted Avg.  0,812    0,190    0,812      0,812    0,811      0,622    0,854     0,818
```

350:

```
=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,822    0,200    0,792      0,822    0,807      0,622    0,845     0,773     pos
               0,800    0,178    0,830      0,800    0,814      0,622    0,845     0,836     neg
Weighted Avg.  0,811    0,188    0,811      0,811    0,811      0,622    0,845     0,806
```

450:

```
=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,821    0,191    0,799      0,821    0,810      0,630    0,841     0,783     pos
               0,809    0,179    0,831      0,809    0,820      0,630    0,841     0,819     neg
Weighted Avg.  0,815    0,184    0,816      0,815    0,815      0,630    0,841     0,802
```

480:

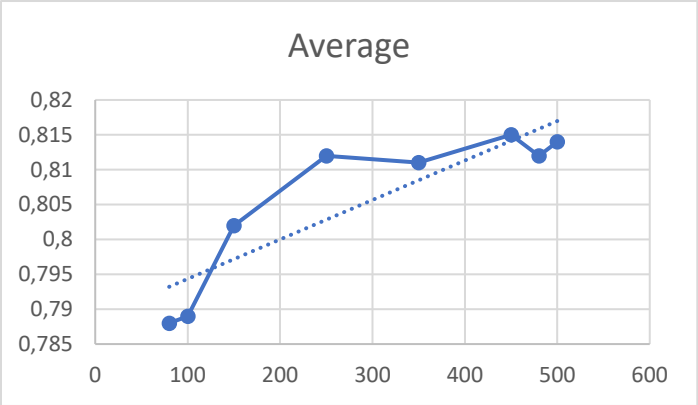
```
=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,822    0,197    0,794      0,822    0,808      0,625    0,849     0,793     pos
               0,803    0,178    0,830      0,803    0,817      0,625    0,849     0,832     neg
Weighted Avg.  0,812    0,187    0,813      0,812    0,813      0,625    0,849     0,813
```

500:

```
=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,828    0,198    0,794      0,828    0,811      0,629    0,851     0,798     pos
               0,802    0,172    0,834      0,802    0,818      0,629    0,851     0,833     neg
Weighted Avg.  0,814    0,185    0,815      0,814    0,814      0,629    0,851     0,816
```



Aus diesen Daten haben wir dieses Diagramm erstellt. Man kann einen klaren Trend erkenne, je mehr Attribute, desto genauer kann der Klassifikationsalgorithmus Vorhersagen.

Man sollte jedoch auch beachten, dass je mehr Attribute zugelassen sind die Berechnungszeit stark zunimmt. So betrug sie für numToSelect = 80 circa 34 s und für numToSelect = 500 circa 200 s. Also bei 5-mal so vielen Attributen braucht das Klassifizieren rund 6-mal so lange.

Abschlussevaluation

Entscheidungsbaum (J48):

Mit einem J48 Klassifikationsalgorithmus erhielten wir meistens ein Ergebnis von circa 81% Genauigkeit. Die Differenz der Vorhersagbarkeit von negativen und positiven Reviews, betrug meist nicht mehr als ±10%. Es ist auffällig, dass die Vorhersage bei Positiven Reviews genauer ist. Dies könnte an der vor Verarbeitung liegen, welche „not“ besonders behandelt.

Die Einstellungen wordsToKeep = 500; numToSelect = 250, sind ein guter Kompromiss aus Effektivität und Aufwand.

Negative und Positive Wörter zu zählen, scheint auch nicht stark ins Gewicht zu fallen.

Entscheidungsbaum (Random Forest):

Mit einem Random Forest Klassifikationsalgorithmus erhielten wir meistens ein Ergebnis um die 87,5% Genauigkeit. Somit ist er zu circa 6,5% genauer als der J48 Klassifikationsalgorithmus.

```
=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,892    0,141    0,854      0,892    0,872      0,750    0,948     0,944     pos
               0,859    0,108    0,896      0,859    0,877      0,750    0,948     0,951     neg
Weighted Avg.  0,875    0,124    0,876      0,875    0,875      0,750    0,948     0,948
```

NaiveBayesMultinomialText:

Sehr interessant ist, dass wenn man einen Naive Bayes Klassifikationsalgorithmus benutzt will, man diesen mit den Rohdaten trainieren und testen sollte, da dieser (in diesem Beispiel) um rund 3,2% besser vorhersagt.

```
=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0,840    0,101    0,885      0,840    0,862      0,741    0,942     0,929     pos
                0,899    0,160    0,859      0,899    0,878      0,741    0,942     0,945     neg
Weighted Avg.   0,871    0,132    0,871      0,871    0,870      0,741    0,942     0,937
```

```
=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0,851    0,172    0,821      0,851    0,836      0,679    0,918     0,905     pos
                0,828    0,149    0,857      0,828    0,842      0,679    0,918     0,925     neg
Weighted Avg.   0,839    0,160    0,840      0,839    0,839      0,679    0,918     0,915
```

Müsste man sich für einen Klassifikationsalgorithmus entscheiden, würden wir uns für den Naive Bayes Multinomal Text entscheiden, dieser benötigt für das Erstellen eines Models die wenigste Zeit. Auch der der Random Forest Algorithmus ist zuverlässiger als der J48.

Wir haben den Naive Bayes Klassifikationsalgorithmus und den Random Forest in seinen Standardeinstellungen getestet, vielleicht sind diese mit speziellen Einstellungen noch präziser.