Künstliche Intelligenz - PVL Dokumentation

Nils Generlich (s85449)

12.06.2024

1 Einführung

Im Folgenden wird das aus der PVL-Aufgabe entstandene Programm dokumentiert und dessen Benutzung erläutert. der Erstellung eines webbasierten Lernprogramms dokumentiert. Das Programm setzt einen Bernoulli Naive Bayes Klassifikator mit Laplace-Glättung um, welcher mittels Training (durch eingelesene Trainingsdaten) Text klassifiziert und diese der Klasse A oder der Klasse B zuordnet. Die Kompilierung und das Starten, sowie das Hinzufügen von Trainingsdaten ist folgend erläutert. Außerdem wird die Klasse NaiveBayesClassifier mit ihren Methoden und Klassenvariablen erläutert. Das als PVL entstandene Programm ist in der Programmiersprache Java geschrieben.

2 Benutzung des Programms

Der Naive Bayes Klassifikator ist als Klasse in der Java-Datei "NaiveBayesClassifier.java" im src-Verzeichnis gespeichert. Er lässt sich mittels zwei übergebenen txt-Dateien trainieren und klassifiziert mit Berücksichtigung der Laplace-Glättung Dokumente (je eine Zeile) einer anderen txt-Datei zu einer der Klassen A oder B. Im Folgenden ist die Kompilierung und Ausführung des Programmes, von der Kommandoszeile aus, beschrieben.

2.1 Kompilieren

Das Java-Programm kann nach Installation des Java Development Kit (JDK) über zwei Wege kompiliert werden:

- Aufruf von Shellskript `./make.sh` (nur über Linux)
- oder dem Befehl `javac -d bin src/*.java` (ausgeführt im Verzeichnis "KI-PVL_s85449")

2.2 Ausführen

- Wechsel in das Binary-Verzeichnis: `cd bin`
- Ausführen des Programms mit:
  `java NaiveBayesClassifier <traindataA> <traindataB> <data_to_be_classified> [includeWordsNotInDataToBeClassified]`
  - **traindataA**: relative Pfadangabe (ausgehend vom Verzeichnis KI-PVL_s85449) zu einem txt-Dokument, welches alle Trainingsdaten für Klasse A beinhaltet (z.B.: data/train/A.txt)
  - **traindataB**: relative Pfadangabe (ausgehend vom Verzeichnis KI-PVL_s85449) zu einem txt-Dokument, welches alle Trainingsdaten für Klasse B beinhaltet (z.B.: data/train/B.txt)
  - **data_to_be_classified**: relative Pfadangabe (ausgehend vom Verzeichnis KI-PVL_s85449) zu einem txt-Dokument, welches die zu klassifizierenden Daten enthält (z.B.: data/test.txt)
  - **includeWordsNotInDataToBeClassified**: *(optional)* Boolean-Wert, welcher bestimmt, ob alle Wörter des Trainingsvokabulars, auch wenn sie nicht in dem zu klassifizierendem Dokument vorkommen, in der zu berechnenden Wahrscheinlichkeit (als Gegenwahrscheinlichkeiten) berücksichtigt werden sollen. (**true** = Berücksichtigung nicht im zu klassifizierenden Dokument vorkommender Wörter, **false** = nur Berücksichtigung von Wörtern, welche im klassifizierenden Dokument vorkommen). Falls kein 4. Kommandozeilenparameter eingegeben wird, gilt hierfür standardmäßig false.
- Aufruf des Programms mit gegebenen Beispieldatensätzen ist damit möglich mit Berücksichtigung aller Wörter des Vokabulars bei der Klassifikation:
  `java NaiveBayesClassifier data/train/A.txt data/train/B.txt data/test.txt true`

  ...und lediglich nur mit der Berücksichtigung der Wörter, die im zu klassifizierenden Dokument vorkommen:
  `java NaiveBayesClassifier data/train/A.txt data/train/B.txt data/test.txt`

2.3 Aufbau der Dateien mit Trainings- oder zu klassifizierenden Daten

2.3.1 Trainingsdaten

- Die beiden Dateien für Klasse A und B müssen eine txt-Datei (Endung .txt) sein.
- Jede Datei steht für eine Klasse und enthält pro Zeile ein Trainingsdatum/-dokument
- Zeilen werden durch einen Zeilenumbruch (\n für Linux/MacOS oder \r\n für Windows) getrennt
- Die Wörter sind sowohl in Groß- und Kleinschreibung erfassbar. Erfasste Wörter sind nicht case sensitive, wodurch es egal ist ob sie groß- oder kleingeschrieben werden. (z.B. gilt "Hallo" als das gleiche Wort wie "hallo")

- Es werden nur Wörter erfasst, welche aus Buchstaben, Ziffern oder Untersrichen bestehen. Gleichsam werden alle weiteren Zeichen, wie bspw. Satzzeichen oder Leerzeichen als Trennzeichen der Wörter verwendet.

2.3.2 Klassifizierende Daten

- Die Datei muss eine txt-Datei (Endung .txt) sein.
- Diese Datei beinhaltet beliebige Zeilen an zu klassifizierenden Dokumenten
- Dabei wird jede Zeile als einzelnes Dokument klassifiziert
- Zeilen werden durch einen Zeilenumbruch (\n für Linux/MacOS oder \r\n für Windows) getrennt
- Die Wörter sind sowohl in Groß- und Kleinschreibung erfassbar. Erfasste Wörter sind nicht case sensitive, wodurch es egal ist ob sie groß- oder kleingeschrieben werden. (z.B. gilt "Hallo" als das gleiche Wort wie "hallo")
- Es werden nur Wörter erfasst, welche aus Buchstaben, Ziffern oder Untersrichen bestehen. Gleichsam werden alle weiteren Zeichen, wie bspw. Satzzeichen oder Leerzeichen als Trennzeichen der Wörter verwendet.

2.3.3 Beispieldaten

Im Ordner *data* befinden sich Beispieldateien mit Beispieldaten aus Aufgabe 25 des KI-Praktikums.

- Trainingsdaten: *data/train/A.txt* oder *data/train/B.txt*
- Dokumente, die klassifiziert werden sollen: *data/test.txt*

3 Verzeichnisstruktur von "KI-PVL_s85449"

- `docs`: Hier befindet sich diese Dokumentation der PVL, in welcher ich auf die Benutzung des Programms und die Klasse inkl. ihrer Methoden (Architektur) eingehe.
- `src`: Hier befindet sich das gesamte Programm in Form von Java-Dateien, welches als Bernoulli Naive Bayes Klassifikator mit Laplace-Glättung fungiert. Mit übergebenenen Trainingsdaten können weitere Dokumente durch dieses Programm klassifiziert werden.
- `bin`: Enthält die kompilierte Klassendatei (.class) der Java-Quellcodedatei (.java), die sich im src-Ordner befindet. Diese ist mittels *java Klassendateiname [Kommandozeilenparameter]* ausführbar.
- `make.sh`: Kompiliert die Java-Dateien aus dem "src"-Verzeichnis zu einem im Verzeichnis "bin" ausführbaren Programm (nur bei Linux).

4 Architektur

Die Klasse NaiveBayesClassifier implementiert einen Naive Bayes Klassifikator für Textklassifikation, der Dokumente basierend auf Trainingsdaten in eine von zwei Klassen kategorisiert.
Als Funktionen des Programm zählen:

- Trainieren des Klassifikators mit Dokumenten aus zwei Klassen (Klasse A und Klasse B).
- Klassifizieren einzelner Dokumente oder mehrerer Dokumente in einer Datei.
- Option, Wörter, die nicht in den Testdaten vorkommen, während der Klassifikation einzubeziehen.

4.1 Klassenvariablen

- **Set<String> vocabulary:**Menge aller Wörter, die in den Trainingsdaten von Klasse A oder B vorkommen
- **Map<String, Integer> wordCountA/wordCountB:**Mapping mit Wort als Schlüssel und die Anzahl, wie oft das Wort in Klasse A/ B vorkommt als Integer-Wert
- **int totalLinesA/totalLinesB:**Anzahl der Dokumente in Klasse A/ B
- **int totalWordsA/totalWordsB:**Anzahl der Wörter in Klasse A/ B
- **int vocabSize:** Anzahl der Wörter im Vokabular
- **String fpA/fpB:** Dateipfad der Trainingsdaten von Klasse A/ B

4.2 Methoden

4.2.1 Konstruktor:

```
public NaiveBayesClassifier()
```

Initialisiert den Klassifikator mit leerem Vokabular und Wortzählungen.

4.2.2 train:

```
public void train(String filePathA, String filePathB) throws IOException
```

Trainiert den Naive Bayes Klassifikator mit den bereitgestellten Trainingsdateien für Klasse A und Klasse B.

Parameter:

- filePathA: Der Dateipfad zu den Trainingsdaten für Klasse A.
- filePathB: Der Dateipfad zu den Trainingsdaten für Klasse B.

4.2.3 trainClass:

```
private boolean trainClass(String filePath, Map<String, Integer> wordCount)
throws IOException
```

Hilfsmethode zum Trainieren des Klassifikators für eine spezifische Klasse.

Parameter:

- filePath: Der Dateipfad zu den Trainingsdaten für die Klasse.
- wordCount: Die Map, um die Wortzählungen für die Klasse zu speichern.

Rückgabewert:

- boolean: true, wenn das Training erfolgreich ist, ansonsten false.

4.2.4 classify_all:

```
public void classify_all(String filePath, boolean includeWordsNotInTestData)
throws IOException
```

Klassifiziert alle Dokumente in der angegebenen Datei und gibt das Klassifizierungsergebnis für jedes Dokument aus.

Parameter:

- filePath: Der Dateipfad zu den Testdaten, die mehrere Dokumente enthalten.
- includeWordsNotInTestData: Ein boolean-Flag, das angibt, ob Wörter, die nicht in den Testdaten vorkommen, während der Klassifikation einbezogen werden sollen.

4.2.5 classify:

```
public String classify(String line, boolean includeWordsNotInTestData)
throws IOException
```

Klassifiziert ein einzelnes Dokument basierend auf den trainierten Daten.

Parameter:

- line: Der Text des zu klassifizierenden Dokuments.
- includeWordsNotInTestData: Ein boolean-Flag, das angibt, ob Wörter, die nicht in den Testdaten vorkommen, während der Klassifikation einbezogen werden sollen.

Rückgabewert:

- String: Das Klassifizierungsergebnis.

4.2.6 main:

```
public static void main(String[] args) throws IOException
```

Hauptmethode, um den Klassifikator von der Kommandozeile auszuführen.

Parameter: args: Kommandozeilenargumente. Erwartete Argumente sind:

- args[0]: Der Dateipfad zu den Trainingsdaten für Klasse A.
- args[1]: Der Dateipfad zu den Trainingsdaten für Klasse B.
- args[2]: Der Dateipfad zu den Testdaten, die mehrere Dokumente enthalten.
- args[3] (optional): Ein boolean-Flag (true oder false), das angibt, ob Wörter, die nicht in den Testdaten vorkommen, während der Klassifikation einbezogen werden sollen.

5 Ergebnisse

5.1 Bernoulli Naiver Bayes Klassifikator

Dieser Bernoulli Naiver Bayes Klassifikator arbeitet mit binären Merkmalen, d.h. er betrachtet nur das Vorhandensein oder Nichtvorhandensein von Wörtern in einem Dokument (welches sich in einer Datei befindet). Das Programm implementiert dies, indem es für jedes Dokument nur die einzigartigen Wörter betrachtet, die in dem Dokument vorkommen.

5.2 Laplace-Glättung

Die Laplace-Glättung wird verwendet, um zu verhindern, dass Wahrscheinlichkeiten auf Null gesetzt werden, wenn ein Wort in den Trainingsdaten fehlt. Dies wird durch Hinzufügen von 1 zur Zähleranzahl (Anzahl der Vorkommen eines Wortes) und Hinzufügen von 2 zum Nenner (Gesamtzahl der Dokumente in der Klasse plus 2) erreicht. Im Code erfolgt dies durch folgende Zeilen:

Listing 1: Java: Laplace-Glättung

```
probA *= (countA + 1.0) / (totalLinesA + 2.0);
probB *= (countB + 1.0) / (totalLinesB + 2.0);
```

5.3 Anwendung der Klassifizierung von Beispiel von Praktikumsaufgabe 25

Hinzufügen von Bildern und Testweise Nachrechnen zum Approven