Handbuch: Face Recognition und Matching

Jakob Kusnick Christopher Schott Hans Dieter Pogrzeba

1 Programmversionen und Installation

Für das Projekt werden verschiedene Bibliotheken benötigt. Die angegebenen Tutorials verwenden ggf. andere Programmversionen.

Java 1.8	aus Ubuntu Paketquellen (OpenJDK 1.8.0_91)
Scala 2.10.6	Download
Hadoop 2.7.2	Download Tutorial
Spark 1.6.1	Download (Hadoop und Scala Version beachten) Tutorial Tutorial
OpenCV 3.1.0	Download Download (mit contrib Erweiterung) Tutorial Tutorial contrib

2 Projektstruktur

Das Projekt ist ein IntelliJ IDEA Projekt, basierend auf einer Gradle Datei. Gradle stellt hierbei externe Bibliotheken, wie JUnit, zur Verfügung. Wir haben anfangs versucht, auch Spark über Gradle einzubinden, das hat jedoch nicht funtkioniert.
In der Entwicklungsumgebung (IntelliJ IDEA) müssen die lokal kompilierten *.jar Dateien von Spark und OpenCV eingebunden werden.

Beim Erstellen neuer Klassen mit Main-Methode, muss darauf geachtet werden, dass (für IntelliJ unter *Run/Debug Configurations/Application/Klassenname*) folgende Optionen gesetzt sind:

VM Options:

```
-Djava.library.path=/home/bduser/opencv/src/lib
```

Environment Variables:

```
HADOOP_USER_NAME=hduser
```

3 Bedienungshinweise

3.1 Hinweise zur Arbeit mit der VM

In der Virtuellen Maschine ist Xubuntu in der Version 16.04 installiert. Das standardmäßige Passwort für die Nutzer „bduser“ und „hduser“ ist „bdpassword“. Nach dem starten des Betriebssystems benötigt man die Kommandozeile, um das HDFS zu starten. Hierzu sind folgende Befehle einzugeben:

```
su - hduser
```

Um in die Sicht des Nutzers „hduser“zu wechseln.

```
cd /usr/local/hadoop/sbin
```

Um anschließend in das nötige Verzeichnis zu wechseln.

```
./start-all.sh
```

Um die verantwortlichen Prozesse zu starten.

Anschließend ist es möglich, innerhalb der IntelliJ IDEA das Projekt zu öffnen und das Programm im GUI-Betrieb zu starten. Hierzu muss die main-Klasse des Frontends (src.main.java.org.bdprak.frontend.Frontend) gestartet werden.

3.2 Bedienung der GUI

Das Programm unterteilt sich in vier unterschiedliche Tabs:

Video Preprocessing In diesem Reiter werden sowohl einzulesende Videos sowie deren Speicherorte im HDFS eingetragen. Dabei werden die Videos automatisch in einzelne Frames unterteilt.

Training Hier werden Ordner mit Bildern zu Personen angegeben sowie deren lokalen Ausgabepfad nach der Vorverarbeitung. Dabei ist darauf zu achten, dass auf den Bildern nur die jeweils gewünschte Person und keine weitere Person abgebildet ist. Außerdem sollte, um den Personennamen eindeutig den Gesichtern zuzuordnen, der entsprechende Name in das zugehörige Feld eingetragen werden.

Recognition In diesem Reiter muss der HDFS-Pfad zum Video angegeben werden, in dem nach den gelernten Gesichtern gesucht werden soll. Sollte bis zum aktuellen Zeitpunkt die Reihenfolge der Reiter eingehalten worden sein, so sollte dieser Pfad automatisch ausgefüllt worden sein. Es ist jedoch auch möglich ein anderes Video, welches sich ebenfalls im HDFS befindet, zu nutzen.

Result Um die Ergebnisse anzuzeigen ist es notwendig erneut einen HDFS-Pfad zu den Einzelbildern eines Videos, in dem nach Gesichtern gesucht worden ist, anzugeben. Auch hier wird bei Einhaltung der Reihenfolge ein Pfad eingetragen. Unser Testvideo haben wir unter *hdfs://localhost:54310/testLoadVideo/* gespeichert.

Reiterübergreifend Im unteren Teil der GUI ist eine Optionsübersicht. Hier sind reiterübergreifende Optionen anzugeben. Hierzu gehören der HDFS-ROOT-Pfad, der HDFS-Pfad zum Modell, welches im Reiter Training erstellt wird, der lokale Pfad zum Modell sowie der Pfad zur Personenmap, die Personen im Modell den Namen zuordnet.

4 Hinweise zum Deployment auf einem echten Cluster

Für die Entwicklung des Programms stand keine Clusterumgebung zur Verfügung. Aus diesem Grund sind für den Betrieb auf einem echtem Cluster eventuell Anpassungen vorzunehmen.

Zum einen könnte die Verteilung des Training Modells problematisch sein, da dieses auf jedem Spark Worker lokal vorliegen muss, jedoch nur auf einem davon erstellt wurde. Für diese Verbreitung existieren bereits Codeabschnitte, welche lokal funktionieren, jedoch bisher auf verteilten Systemen ungetestet sind. Zu finden sind diese in der Datei „FolderSelectPanel.java“ in Zeile 136-148.

Aus dem genannten Grund, war es ebenfalls nicht möglich, Aussagen über die Lastverteilung bzw. Partitionierung der Bilddaten während der parallelen Erkennung auf den einzelnen Workern zu testen. Die Verteilung der Bilder basiert zurzeit auf einer Zuordung der Dateinamen. Gegebenenfalls sollten die jeweiligen Bilder dann auch dem entsprechenden Worker bereits vorliegen.

Ein weiteres Problem könnte die Verfügbarkeit der OpenCV-Bibliotheken auf den einzelnen Spark Workern sein. Diese sind nötig um die Gesichtserkennung parallel auf den separaten Systemen durchzuführen.