Konzept: Face Recognition und Matching

Jakob Kusnick Christopher Schott Hans Dieter Pogrzeba

Zusammenfassung

Es existieren mittlerweile einige Technologien, die eine Gesichtserkennung auf Bildern mittels maschinellen Lernens ermöglichen. Auf großen Datenmengen benötigen diese sehr viel Zeit, da herkömmliche Datensysteme nicht die nötige Skalierbarkeit und Performanz bieten. An dieser Stelle kommen Big Data Technologien zum Einsatz. Diese können große Datenmengen aufnehmen und den Zugriff auf diese Daten hinreichend schnell gewährleisten. Eine Parallelisierung der Gesichtserkennung könnte die nötige Performanz liefern, um den Prozess in Echtzeit zu realisieren.

1 Zielstellung

Ziel unserer Arbeit ist es Bibliotheken zur Gesichtserkennung in eine Big Data Umbegbung zu integrieren. Dabei soll dem Nutzer eine Oberfläche zugänglich gemacht werden, über die er Bilder zu bestimmten Personen und zu analysierendes Material zur Verfügung stellen kann. Danach kann der Nutzer über die Oberfläche den Prozess zum Erlernen der Bilder bzw. der Modellbildung und anschließend den parallelisierten Analyseprozess starten. Am Ende kann der Nutzer z.B. ein analysiertes Video mit graphischen Markierungen für die erkannten Gesichter abspielen. Zudem sollen Analyseergebnisse mit Hilfe des Big Data Frameworks Spark erstellt werden.

2 Technologien

2.1 Hadoop

Für die zugrundeliegende Datenstruktur wird *Apache Hadoop* in Version 2.7.2 genutzt. Hadoop ist ein Opensource Framework von Apache, das sich besonders durch seine gute Skalierbarkeit, aufgrund der Nutzung verteilter Systeme, auszeichnet. Eine der wichtigsten Eigenschaften von Hadoop, die für dieses Projekt besonders wichtig ist, ist die Möglichkeit, mittels Hadoop intensive Rechenprozesse und Analysen von großen Datenmengen auf Computerclustern durchzuführen. Dies ermöglicht eine effizientere Nutzung von Ressourcen und somit einen performanteren Machine-Learning-Prozess. Dazu muss ein *HDFS* (*Hadoop Distributed File System*) erstellt und die Bilder für das maschinelle Lernen darin gespeichert werden. Somit können die Daten auf verteilten Systemen gespeichert und parallel genutzt werden.

2.2 Spark

Um auf die im HDFS verteilt vorliegenden Daten effizient zugreifen und diese parallel verarbeiten zu können, wird *Apache Spark* in Version 1.6.1 benutzt. Dabei handelt es sich um ein Open Source Framework für *Cluster Computing*. Auf diese Weise kann der Programmablauf parallelisiert bzw. verteilt werden, was insbesondere die Verarbeitung von Videodaten beschleunigen sollte.

2.3 OpenCV

Als Bibliothek für die Bilderkennung und Bearbeitung soll *OpenCV* in der Version 3.1.0 dienen. Dabei handelt es sich ebenfalls um eine freie Bibliothek, die Algorithmen für Bildverarbeitung liefert. In unserem Projekt kommt *OpenCV* zur Vorverarbeitung der Bilder zum Einsatz und liefert zudem die notwendigen Algorithmen zur Face Detection und Face Recognition.

3 Arbeitsschritte

Abbildung 1 stellt den Ablauf der verschiedenen Teilschritte des Prozesses dar. Im folgenden werden diese näher beschrieben.

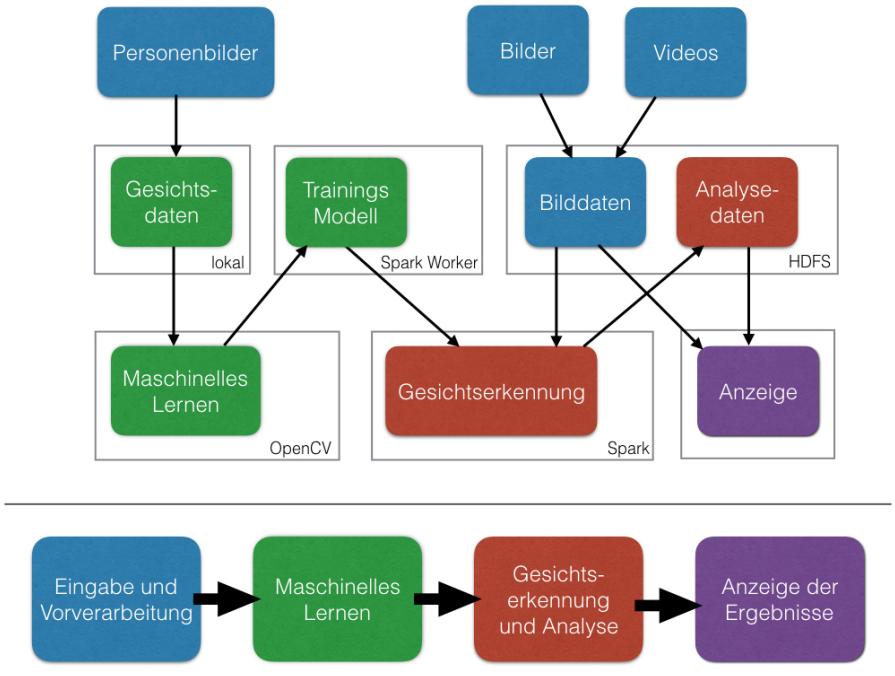

Abbildung 1: Programmablauf und Komponentenübersicht

3.1 Eingabe und Vorverarbeitung

3.1.1 Personenbilder

Der Nutzer liefert Ordner mit Bildern von jeweils einer Person. Aus diesen soll später das Modell zur Face Recognition erstellt werden. Zur Verwendung mit den von *OpenCV* zur Verfügung gestellten Algorithmen, müssen die Bilder bestimmte Eigenschaften erfüllen, die durch eine Vorverarbeitung sichergestellt werden.
Die Bilder müssen ein Gesicht frontal und gerade, sowie mit einem einheitlichen Randabstand abbilden. Dazu wird mit Hilfe von *OpenCV* die Position der Augen erkannt. Relativ dazu wird das Bild gedreht und ausgeschnitten. Anschließend werden die Bilder auf eine einheitliche Größe skaliert und als Schwarz-Weiß-Bilder gespeichert.
Da der Vorverarbeitungsprozess auf den Erkennungsfunktionen von *OpenCV* basiert, werden Bilder, auf denen mehr als ein Gesicht bzw. nicht genau zwei Augen erkannt wurden, aussortiert. Gegebenenfalls muss im Anschluss noch manuell geprüft werden, ob alle Bilder im Einzelfall richtig verarbeitet wurden.

3.1.2 Medien zur Analyse

Des Weiteren gibt der Nutzer Medien an, auf denen später nach Gesichtern gesucht werden soll. Videos werden dabei in Einzelbilder, nummeriert nach der Framezahl, umgewandelt. Zur anschließenden parallelen Bearbeitung werden die Bilder ins *HDFS* geladen.

3.2 Machine Learning

Mit Hilfe von *OpenCV* wird aus den vorverarbeiteten Bildern ein Modell zur Face Recognition erstellt. Dieses muss anschließend an alle, an der Analyse beteiligten *Spark Worker*, verteilt werden.

3.3 Face Recognition und Analyse

Die Analyse der Eingabemedien soll parallelisiert als Spark Prozess erfolgen. Dabei werden als Grundlage alle zu analysierenden Bild- bzw. Framedateien in einem *JavaPairRDD* zusammengefasst. Mit Hilfe einer *map*-Funktion wird für jeden Frame die Erkennung durchgeführt. Diese transformiert die Bilder bzw. Bildpfade in ein Ergebnistupel aus Framezahl und der Menge der erkannten GesichtsIDs, welches für weitere Auswertungen mit Spark genutzt werden kann. Außerdem werden die Erkennungsergebnisse als Textdatei im JSON Format im *HDFS* gespeichert.

Die eigentliche Gesichtserkennung erfolgt in zwei Schritten. Zuerst werden mit Hilfe eines von *OpenCV* bereitgestellten *CascadeClassifier* alle Gesichter in einem Frame gesucht. Für alle Bereiche, in denen ein Gesicht erkannt wurde,

versucht das aus den Trainingsbildern erstellte Modell die entsprechende Person zu identifizieren.

Als Beispiele für weitere Analysen mit den Ergebnistupeln sollen zunächst Übersichten über die erkannten Gesichter in bestimmten Videobereichen und eine Zählung der Frames, in denen ein jeweiliges Gesicht erkannt wurde, dienen.

3.4 Anzeige der Ergebnisse

Neben einer Anzeige der Spark Analysen auf der Konsole, kann der Nutzer ein analysiertes Video abspielen, wobei zu jedem Frame die JSON Informationen aus dem HDFS geladen und die Gesichter mit einem farbigen Rahmen hervorgehoben werden. Der bei der Vorverarbeitung angegebene Name zu der entsprechenden Person wird ebenfalls angezeigt.