HUGO

Highly Useful Graphical Output

Durchgeführt im Rahmen des Moduls „Big Data Engineering“ des Master Studiengangs Wirtschaftsinformatik an der Fachhochschule Münster.

Gruppenmitlieder:

Ben Lohrengel Matr. Nr.:

Sebastian Manns Matr. Nr.:

André Meier Matr. Nr.:

Prüfer

Erstprüfer: Lars George

Zweitprüfer:

Abgabedatum: 25.02.2016

Inhaltsverzeichnis

[Einleitung 3](#_Toc443831954)

[Idee 3](#_Toc443831955)

[Input 3](#_Toc443831956)

[Wikimedia 3](#_Toc443831957)

[Verarbeitung 3](#_Toc443831958)

[Output 3](#_Toc443831959)

[Verwendete Tools 3](#_Toc443831960)

[HBase 3](#_Toc443831961)

[Map Reduce 3](#_Toc443831962)

[Hadoop File System 3](#_Toc443831963)

[Oozie 3](#_Toc443831964)

[Ergebnisse 4](#_Toc443831965)

[Visionen 4](#_Toc443831966)

[Fazit 4](#_Toc443831967)

# Einleitung

## Problemstellung

Das Internet enthält heute eine unvorstellbare Menge an Videos. Im Dezember 2014 wurden alleine auf der Videoplattform YouTube, pro Minute, 12,5 Tage Material hochgeladen. All diese Videos sind einmalig und identifizieren sich aus einer bestimmten Kombination von Pixeln. Die darin enthaltenen Farbwerte machen den gesamten Videostream unverwechselbar.

Es ergibt sich nun die Fragestellung inwieweit das Bildmaterial der Videos auf seine Farbwerte hin analysiert und ausgewertet werden können. Als Ergebnis sollen zwei Farbwerte visualisiert werden. Zum einen ein durchschnittlicher Farbwert, basierend auf allen im Stream vorkommenden Pixel und zum anderen der am häufigsten vorkommende, also dominierende Farbwert. Das Ziel ist es eine Plattform für Nutzer zu schaffen, auf der sie die Ermittlung durchführen lassen können. Dazu ist es zunächst nötig den Link einer, prinzipiell beliebigen, Videoquelle einzugeben. Nach der Betätigung eines Buttons, soll sich das System eigenständig den Viedostream herunter laden und ihn lokal ablegen. Die darauffolgende interne Verarbeitung soll mit verschiedenen Tools aus dem Bereich des Big Data Engineering erfolgen. Zum Abschluss des Analyseprozesses soll dem Nutzer das Ergebnis in Form einer Durchschnittsfarbe sowie die ermittelte dominierende Farbe auf der Plattform präsentiert werden.

Ziel dieses Projektes namens “Highly Useful Graphical Output” kurz HUGO, ist es nun, die zuvor beschriebene Vorgehensweise zu realisieren.

## Zielsetzung und Aufbau

### Input

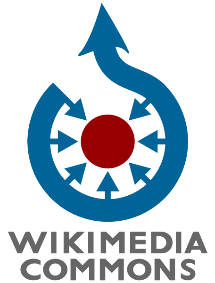
Zu Beginn des Prozesses wird ein Videostream benötigt, den ein Nutzer auf seine dominante bzw. Durchschnittsfarbe analysieren lassen möchte. Dabei müssen jedoch zwei Voraussetzungen erfüllt sein.

1. Der Stream muss online verfügbar und über einen direkten Downloadlink erreichbar sein.
2. Die Onlineplattform auf der sich der Stream befindet muss über eine frei zugängliche API erreichbar sein.

Ist beides erfüllt, so kann der Nutzer den http-Link zum Video kopieren und in der Eingabemaske des HUGO einfügen.

Jedoch wird bei den meisten Videoplattformen der zuvor erwähnte Punkt 2 leider nicht erfüllt. In der Regel sind es datenschutzrechtliche Gründe die große Anbieter wie „Youtube“ dazu bewegen keine freie API mehr anzubieten. Die Wikimedia Foundation hingegen bietet auf ihrer Website „wikimedia.org“ Videostreams an die basierend auf freien Lizenzen genutzt und vor allem mittels ihrer API „MediaWiki“ von Drittsystemen verwendet werden können.

**Wikimedia Commons**

Wikimedia definiert sich als weltweite Bewegung für freies Wissens. Unter dem Dach der gemeinnützigen Wikimedia Foundation werden verschiedenste Wikimedia-Projekte wie Wikipedias angeboten. Darunter befindet sich ebenfalls Wikimedia Commons, welche zum Ziel hat, als zentrales Medienarchiv in verschiedenen Sprachen für die Wikipedias zu dienen. Die Medienplattform stellt somit gemeinfreie und frei-lizensierte Medieninhalte wie Bilder, Audio- und Videodateien bereit. Derzeit werden über 30 Millionen Dateien zur Verfügung gestellt.

Die Gemeinfreiheit sowie das freie Lizensierungsmodell gestatten es Videoframes ohne die Verletzung von Persönlichkeits- und Datenschutzrechten sowie des Urheberrechts herunterzuladen und nicht-kommerziell zu nutzen. Die Videoinhalte können dabei über einen direkten Link erreicht werden. Das bedeutet, dass der Link nicht, wie bei anderen Plattformen, auf die Webseite des Videos führt, sondern direkt zum Download verwendet werden kann. Somit erfüllt Wikimedia Commons die beiden zuvor erwähnten Voraussetzungen für eine verwendbare Videoquelle. Dadurch wird diese Plattform im Folgenden als Videoherkunft für das HUGO-Projekt verwendet.

Abbildung Logo Wikimedia Commons

### User Interface

Als zentrale Anlaufstelle zur Interaktion mit dem HUGO-Projekt, dient eine einzige Webseite. Hier soll zunächst eine Eingabemaske zu sehen sein. Der zuvor kopierte Link zum Videostream ist dort einzufügen. Im nächsten Schritt ist der Nutzer dazu angehalten anzugeben welches Ergebnis er haben möchte. Es kann entweder die dominierenden Farbe oder die Durchschnittsfarbe ermittelt werden. Nach der Betätigung eines Buttons zur Bestätigung, wird das entsprechende Video im Hintergrund vom System heruntergeladen und lokal gespeichert. Anschließend soll die interne Verarbeitung ebenfalls unsichtbar durchgeführt werden. Für den Nutzer ist auf der Webseite ein Lade- bzw. Fortschrittsbalken zu visualisieren, anhand dieser den Prozessfortschritt erkennet. Zudem kann die Beendigung einzelner Prozessabschnitte, wie bspw. der erfolgreiche Download durch Ausgabe eines Textes dem Nutzer erscheinen. Zum Abschluss der Verarbeitungskette sind die Ergebnisse in Form einer Grafik wieder auf der Webseite angezeigt.

### Output

In den vorherigen Abschnitten erfolgte die Definition des erwarteten Inputs, der zukünftig verwendeten Quelle sowie der graphischen Oberfläche. In diesem Abschnitt soll es nun kurz um eine Beschreibung des erwarteten/möglichen Ergebnisses bzw. Outputs gehen.

Nachdem eine Videoquelle erfolgreich eingelesen, verarbeitet und analysiert wurde soll auf der zentralen Weboberfläche die Ausgabe erfolgen. Abhängig von der zuvor getroffen Wahl des Nutzer, sind die Farbwerte hinsichtlich eines Ergebnisses zu ermitteln und in Form einer Grafik zu visualisieren. Dabei stehen folgende Analysemöglichkeiten zu verfügung.

1. **Durchschnittsfarbe** (average Color)

Ziel bei der Ergebnisform „Durchschnittsfarbe“ ist es den durchschnittlichen Farbwert pro Frame zu ermitteln.

1. **Dominierende Farbe** (dominant Color)

Bei der Ergebnisform „Dominierende Farbe“ handelt sich um die Ermittlung jenes Farbwertes pro Frame, der am häufigsten vorkommt.

Unabhängig von dem gewünschten Ergebnis, sind die ermittelten RGB-Werte eines jeden Frames abzulegen. Zur abschließenden Darstellung müssen die einzelnen Durchschnitts- bzw. dominierenden Farbwerte nacheinander als Pixel ausgegeben und in einer Grafik zusammengefasst werden. Die so entstandene Grafik bildet somit eine chronologische Aneinanderreihung der Farbwerte eines Videostreams. Zuletzt ist dann die so entstandene Grafik auf der Weboberfläche auszugeben. Der Nutzer erhält damit das von ihm gewünschte Ergebnis und die Arbeit des HUGO-Projektes ist beendet.

* Webseite
* Video
* Blocksize
* Analyseverfahren
  + Durchschnittsfarbe
  + Dominante Farbe

# Verwendete Tools

## HBase

## Map Reduce

* Pixel von Bildern verarbeiten
* Durchschnittswerte

## Hadoop File System (HDFS)

* Für Videodateien und Frames

## Oozie

* Workflow
* Tracking um den Status des Jobs an die Webseite zurückzugeben

## Xuggler

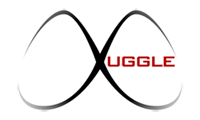
Xuggler definiert sich selber als einfachen Weg Videostreams zu dekomprimieren, zu manipulieren und wieder zu komprimieren. Es handelt sich dabei um eine Java-Bibliothek und ist unter der GPL Version 3 Lizenz frei verfügbar. Verwendet wird die leistungsstarke FFmpeg media handling librarie.

Abbildung Xuggle Logo

Im Rahmen des HUGO-Projektes wird Xuggler dazu verwendet, um einen zuvor geladenen und abgespeicherten Videostream in seine einzelnen Frames zu zerlegen. Anschliessend werden diese Frames im HDFS angelegt und mit MarReduce analysiert.

# Implementierung

In den vorherigen Kapiteln haben wir zunächst die Idee des HUGO-Projektes zusammen mit dem gelieferten Input sowie den zu erwartenden Output kennengelernt. Darauf folgte eine Einführung in die Tools die im Rahmen dieses Projektes zum Einsatz kommen werden.

Im Folgenden Kapitel soll es nun um die konkrete Implementierung von Hugo gehen. Zunächst wird dabei eine Prozessübersicht gezeigt. Mit ihr sollen vorab die Zusammenhänge zwischen den einzelnen Modulen bzw. Prozessschritten visuell dargestellt werden. Darauf erfolgt dann eine detaillierte Beschreibung der einzelnen Prozessphasen. Beginnend mit dem einlesen des Videostreams, folgt das Zerlegen der Videostreams sowie die Analyse der so ermittelten Frames und schließt mit der Ergebnispräsentation.

## Prozessübersicht

Abbildung 3 zeigt die Prozessübersicht des gesamten HUGO-Projekts. Es verdeutlicht die Zusammenhänge zwischen den im Anschluss beschriebenen einzelnen Phasen.

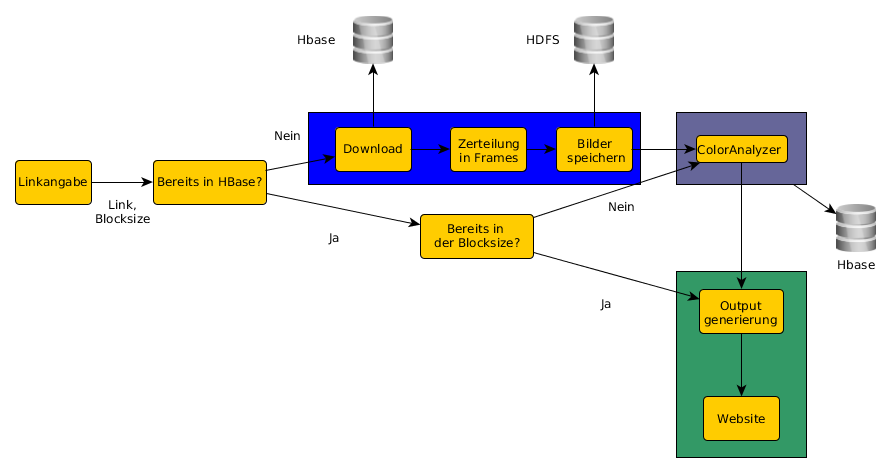


Abbildung Prozessübersicht des HUGO-Projektes

## Prozessphasen

### Einlesen und Zerlegen der Videostreams

Wenn der auf der Website angegebene Link nicht als Key in der HBase Datenbank zu finden ist, wird der Download gestartet. Dies geschieht über einen FSDataOutputStream, der das Video in das festgelegte Verzeichnis "hugo/Videos/filesName" schreibt, wobei FilesNames aus dem Hyperlink extrahiert wird.  
  
<TODO: Welches Ablageformat der Videos?> <@André: TODO Offset beschreibung>  
  
Um das Video in Frames zu zerlegen, wird das Framework Xuggler genutzt. Auf das Framework wird in Kapitel <TBD> eingegangen.  
Der Verweis auf das Video erfolgt als Inputargument in der Mainmethode. Das Video wird temporär zwischengespeichert, um das Extrahieren aus der Offsetdatei zu vereinfachen.  
Xuggler zerlegt daraufhin das Video in einzelne Frames, für die Bearbeitung in BufferedImages vom Typ 3Byte BGR, also ein 8-Bit RGB BufferedImage ohne Alphachannel. Über die Systemzeit wird dann geprüft, ob eine festgelegte Zeit zwischen den Frames verstrichen ist, sodass ein neues Bild erstellt wird.  
Die Bilder werden im Datenformat ".png" unter Angabe einer fortlaufenden Nummer in dem festgelegten Ordner "hugo/Frames/NameDesVideos" gespeichert, wobei NameDesVideos sich aus dem Namen der Videodatei sowie (bereinigtem) Timestamp zusammensetzt. Diese Kombination garantiert die Eindeutigkeit des Pfades, wobei zur Absicherung im Code noch das Vorhandensein des Pfades abgefragt und nötigenfalls gelöscht wird.  
Die Speicherung der Bilder erfolgt im .png Dateiformat, da die Komprimierung verlustfrei erfolgt und durch Xuggler problemfrei umgesetzt wird.  
  
Final wird eine "Links.txt" im HDFS-Verzeichnis der Frames erstellt. In dieser wird, durch einen Zeilenumbruch getrennt, für jedes einzelne Bild der entsprechende Link gespeichert. Diese Datei wird dem MapReduceJob als Input zur Verfügung gestellt.

### Analyse der Frames

#### MapReduce Jobs

Das HUGO-Projekt enthält zwei MapReduce Jobs, einen, der die durchschnittliche Farbe eines Bildes ermittelt (im folgenden MR1 genannt), während der andere dessen dominante Farbe bestimmt (im folgenden MR2 genannt). Der grundlegende Aufbau ist bei beiden gleich: Jeder hat eine Main- und Run-Methode, einen Mapper und einen Reducer. Es gibt jedoch die Ausnahme, dass der MR1 eine Combinerklasse besitzt.

Die Mainmethode beider Jobs erhält als Input eine Datei. Diese enthält die, zu Workflowbeginn angegebene Blockgröße und stellt diese per Parameterübergabe dem Mapper zur Verfügung. Zudem wählt die Mainmethode die letzte im HDFS modifizierte Datei in einem festgelegten Verzeichnis aus. Dies stellt sicher, dass automatisch die zuvor beschrieben "Links.txt" ausgewählt wird. Gemäß dem Workflow bildet ihre Erstellung den Abschluss des Zerlegungsprozesses.

Für jeden zu analysierenden Frame ist es notwendig ausschließlich einen MapReduceJob zu starten. Andernfalls würde das verwendete Rechenschema die Ergebnisse derart verfälschen, das sie unbrauchbar werden. Aus diesem Grund wird aus der "Links.txt" jede Zeile einzeln ausgelesen und ein exklusiver MapReduceJob gestartet. Zu beachten ist, dass die erste Zeile immer den Key für die Datenbank enthält, welcher per Parameterübergabe an den Reducer übergegeben wird.

Obligatorisch setzt die Run-Methode die Configuration, den Job, das FileInputFormat, die Mapperklasse und dessen Outputkeyklasse und -valueklasse sowie die Reduceklasse. Bei MR1 wird ebenso die Combinerklasse festgelegt.

Im Folgenden wird auf die einzelnen MapReduceJobs eingegangen.

#### Ermittlung des durchschnittlichen Werts – MR1

**Rechenschema am Beispiel**

Als Beispiel dient in folgenden ein Bild mit der beispielhaften URL http://shorturl.de/vid. Es besteht aus 4 Pixeln, wobei zwei Pixel den Farbwert R100 G0 B0 haben, ein Pixel den Wert R180 G0 B0 sowie ein Pixel den Wert R255 G0 B0 hat. In diesem Beispiel behandeln wir nur den Rot Wert, da Blau und Gelb beide 0 sind.

Zunächst wird gezählt, dass R100 zweimal vorhanden ist, R180 und R255 nur einmal. Danach wird der relative Anteil der Farbe an der Gesamtpixelzahl ermittelt, für R100 ist es 2/4 = 0.5, für R180 und R255 1/4= 0.25. Danach wird der relative Anteil mit dem Farbcode multipliziert, sodass man folgende Zahlen erhält: Für R100 50, für R180 45 sowie R255 abgerundet 63. Summiert man diese Zahlen, erhält man das Gesamtergebnis der durchschnittlichen Farbe für alle R-Werte des Bildes, also gerundet 158. Als Gegenprobe kann können die Farbwerte summiert und durch die gesamte Pixelanzahl dividiert werden, was zum gleichen (abgerundeten) Ergebnis führt ((100+100+180+255) / 4) = 158).

Zusammenfassend ergeben sich also folgende Prozessschritte:

1. Häufigkeit der einzelnen Farbwerte ermitteln
2. Ermittlung des relativen Anteils des Farbcodes an der gesamten Pixelzahl
3. Zusammensetzung des Wertes durch Summierung der Anteil

**Mapper**

Der Mapper erhält als Parameter die angegebene Blockgröße sowie einen Verweis auf das zu analysierende Bild. Dieses wird aus dem HDFS geladen und für jeden Pixel via java.awt.Color.getRGB() jeder einzelner R, G und B Wert ermittelt. Der Mapperoutput wird als Key, bestehend aus dem Codetyp (R, G oder B), dem Wert (je nach Blockgröße gerundet) sowie der gesamten Pixelanzahl des Bildes, und als Value 1 übergeben. Das Output beim o.g. Beispiel wäre <R100,4, 1> <R100,4, 1> <R255,4, 1> <R180,4, 1>.

**Combiner**

Der Combiner aggregiert die Häufigkeit eines "gesehenen" Farbcodes, sortiert nach Grundfarbe. Das heißt, er zählt wie häufig beispielsweise der Wert R10 gesehen wurde. Dies geschieht für alle im Mapper erkannten R, G und B Werte. Danach wird der relative Pixelanteil an der Gesamtpixelzahl ermittelt und anschließend mit dem Farbcode multipliziert. Dieser Wert wird als Value, zusammen mit dem Key R, G oder B an den Reducer weitergegeben. Ein Beispieloutput wäre <R, 50> <R, 45> <R, 63>.

**Reducer**

Der Reducer erhält alle Farbwerte des Durchschnitts, gruppiert nach R, G bzw. B. Anschließend werden die Werte gruppiert aufsummiert, sodass nur noch ein einziger Farbwert bleibt. Dieser finale Wert wird dann in HBase gespeichert, mit dem Hyperlink des Videos als Key, averageColor als Columfamily, der Grundfarbe als Qualifier sowie dem Farbwert dann als Value. Im Beispiel bleibend wäre dies <http://shorturl.de/vid, averageColor, R, 159>.

#### Ermittlung der dominanten Farbe – MR2

**Mapper**

Der Mapper verhält sich nahezu identisch zu MR1. Das heißt, es wird ermittelt, dass ein gesamter Farbwert "gesehen" wurde. Allerdings wird diesmal als Value der gesamte Farbcode ausgegeben, sodass das Output beim o.g. Beispiel <R100 G0 B0, 1> <R100 G0 B0, 1> <R255 G0 B0, 1> <R180 G0 B0, 1> ist.

**Reducer**

Der Reducer ermittelt für jeden Key (also Farbwert), dessen Häufigkeit des Auftretens. Dabei wird gegen eine globale Variable maxValue geprüft, ob das Ergebnis größer ist. Falls dies der Fall ist, wird der zugehörige Farbwert, getrennt nach R, G oder B Wert in die HBase Table geschrieben. Am o.g. Beispiel orientiert, wäre dann der Output <http://shorturl.de/vid, dominantColor, R, 100> <http://shorturl.de/vid, dominantColor, G, 0> <http://shorturl.de/vid, dominantColor, B, 0>.

### Ergebnispräsentation

* Prüfen
* Ablaufmodell
* Download
* Zerlegen
* MapReduce Job

Out Image generieren und abspeichern

# Ergebnisse

# Visionen

* Was kann man noch machen
* Der Nutzer kann die Abtastrate selber wählen
* Evtl übertragbar auf den sound der videos

# Fazit

http://de.statista.com/statistik/daten/studie/207321/umfrage/upload-von-videomaterial-bei-youtube-pro-minute-zeitreihe/