# Projektdokumentation Datenbanksysteme

## **Erfolgsfaktoren auf YouTube**

Adrian Franz Willi
Frederico Fischer
Nico Iseli
Rahul Pucadyil

Abgabedatum: 25. Juni 2020
Dozent: Prof. Dr. Michael Kaufmann
Eingereicht im Rahmen des Studiengangs: Informatik

**Hochschule Luzern**
Departement Informatik

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# 1 Einleitung

## 1.1 Ausgangslage und Problemstellung

Im Rahmen des Moduls Datenbanksysteme wird den Studierenden des Studiengangs Informatik der Hochschule Luzern die Aufgabe gestellt, ein Projekt durchzuführen, das passend zu den theoretischen Inputs ebenfalls die Praxis abdeckt. Die Studierenden bilden dafür Projektgruppen und definieren dabei einen Use-Case, den sie behandeln möchten. In der Themenwahl sind sie frei.

Das Projekt ist insofern relevant, als dass es Theorie sowie Praxis kombiniert. Ferner haben die Studierenden die Möglichkeit, sich in einer spezifischen Domäne zu spezialisieren (Technologie, Datenbanken etc.).

## 1.2 Aufbau und Methodik

Der Aufbau erfolgt anhand der Vorlesungen. Daher sollen die wöchentlichen Inputs jeweils in die Arbeit einfliessen, damit die Aufgaben über das Semester verteilt werden.

## 1.3 Zielsetzung

Ziel des Projekts ist es, dass sich die Studierenden in einer spezifischen Datenbanktechnologie vertiefen und dadurch neben den Inputs durch die Vorlesungen von weiterem Wissenserwerb profitieren. Ein weiterer Anspruch an das Projekt ist das Umsetzen eines sinnvollen Use Cases. Dadurch wird neben den Technologieaspekten auch der wirtschaftliche und wissenschaftliche Aspekt abgedeckt.

# 2 Datenmanagement

## 2.1 Datennutzung

### 2.1.1 Use Case

Für das vorliegende Projekt hat sich das Projektteam dazu entschieden, Daten von Youtube-Videos zu analysieren. Youtube hat seit ihrer Gründung im Jahr 2005 kontinuierlich an Popularität gewonnen und ermöglicht heutzutage sogar, die Plattform als Vollzeitbeschäftigung zu nutzen. Durch die zunehmende Bedeutung von Werbekampagnen und „Affiliate Marketing“ etabliert sich der Beruf „Youtuber“ und „Influencer“ immer wie mehr als Berufsbezeichnung. Für Leute, die einem solchen Beruf nachgehen möchten, ist es zunächst wichtig, Erfolgsfaktoren zu erkennen und ihre Community stets auszubauen. Die Plattform ist jedoch nicht nur für Erwerbstätige in diesem Bereich interessant. Aufmerksamkeit oder Informationsverbreitung sind weitere treibende Faktoren, die Plattform zu nutzen. Auch für diese Anspruchsgruppe ist das Wissen über erfolgsversprechende Aspekte wichtig.

Erfolg ist sehr individuell interpretierbar. Für die einen sind viele Likes wichtig, andere wollen eine möglichst hohe Anzahl an Views und wiederum andere möchten sich mit ihrer Konkurrenz vergleichen. Daher hat das Projekt zum Ziel, über eine Webseite möglichst viele Anfragen zu ermöglichen, so dass der Youtuber (also der Kunde) seinen Account optimal analysieren kann. Die Auswertungen sollen im Anschluss graphisch dargestellt werden, so dass diese auch für den Laien verständlich sind.

Der genutzte Datensatz ist auf Kaggle unter dem Namen „Trending YouTube Video Statistics“ verfügbar (Mitchell J, 2019). Kaggle ist eine Online-Community, die sich an Datenwissenschaftler richtet und eine Menge an freien Daten zur Verfügung stellt.

### 2.1.2 Entscheidungsgrundlage

Wie bereits erwähnt, sind Ziele und der Erfolg auf Youtube sehr individuell zu betrachten. Eine hohe Anzahl Views, viele Like oder eine umfangreiche Diskussion mit der Community (Kommentare) sind nur einige Beispiele davon. Die vorliegende Arbeit hat daher zum Ziel, dass jeder Youtuber seinen Account analysieren kann und dabei die Möglichkeit hat, die für ihn wichtigen Zahlen zu beurteilen. Damit kann er sich mit wenigen Mausklicks sofort einordnen und abschätzen, wie es um seine Erfolgschancen steht.

## 2.2 Datenarchitektur

Um das Ziel des Use Cases zu erreichen, wurden folgende Daten verwendet:

- Video
  - ID
  - Titel
  - Veröffentlicht (Datum)
  - Trends erreicht (Datum)
  - Herkunft (Land)
- Channel
  - ID
  - Titel
- Interaktionen
  - Likes / Dislikes
  - Views
  - Anzahl Kommentare
  - Ratings
- Kategorien
  - ID
  - Name

Die Eigenschaften dieser Daten werden im Kapitel Datenmodellierung (vgl. Abschnitt 3 auf Seite 10) genauer erläutert. Diese Daten wurden von Mitchell Jolly auf Kaggle (Mitchell J, 2019) bezogen. Auf die Herkunft dieser Daten wird im Kapitel Quelle und Import (vgl. Abschnitt 5.2 auf Seite 18) genauer eingegangen.

## 2.3 Datenadministration

Mit dem Use Case sollen Personen angesprochen werden, die Youtube erfolgreich nutzen möchten. Die Datenauswertung soll ermöglichen, den eigenen Account analysieren zu können. Die Ergebnisse werden über ein Website zur Verfügung gestellt, die auf Anfrage individuelle Resultate liefert. Die Resultate sind somit für alle zugänglich.

## 2.4 Datentechnik

Als Datentechnik wurde aus Interessensgründen entschieden, eine NoSQL-Technik zu verwenden. Hierfür ist MongoDB angedacht, die Daten im JSON-Format verwaltet. Grund hierfür ist, dass MongoDB eine weit verbreitete NoSQL-Technologie ist und somit eine grosse Community aufweist. Dadurch lässt sich bei Problemen schneller eine sichere Quelle finden. Für den Server wird auf die Ressourcen vom EnterpriseLab zurückgegriffen. Dafür wurde eine Virtuelle Maschine mit Ubuntu reserviert. Da die Ressource extern bezogen wird, ist die Katastrophenvorsorge sowie die Replikation durch das EnterpriseLab sichergestellt. Die Daten werden im CSV-Format bezogen. MongoDB enthält Funktionen, die das Importieren von CSV-Daten in das Datenbankmanagementsystem ermöglichen.

## 2.5 Übersicht Architektur des Projektes

Um die nachfolgenden Erläuterung besser im Kontext des vorliegenden Projektes zu verstehen, werden hier die verwendeten Technologien sowie die gesamte Architektur (vgl. Abbildung 1 auf Seite 9) erläutert. Die Architektur lässt sich in drei Schichten (Daten, Logik und Präsentation) unterteilen. In der Datenschicht wird, wie bereits erwähnt, eine MongoDB verwendet, um die Daten zu speichern. In der darüberliegenden Logikschicht erfolgen die Abfragen sowie deren Verarbeitung. Die gesamte Logik ist in der Programmiersprache Python geschrieben. Damit die User Abfragen machen können, wird mit Hilfe des Webframeworks Flask eine Website generiert. Die Webseite selbst nutzt die gängigen Webtechnologien wie HTML, CSS und JavaScript.

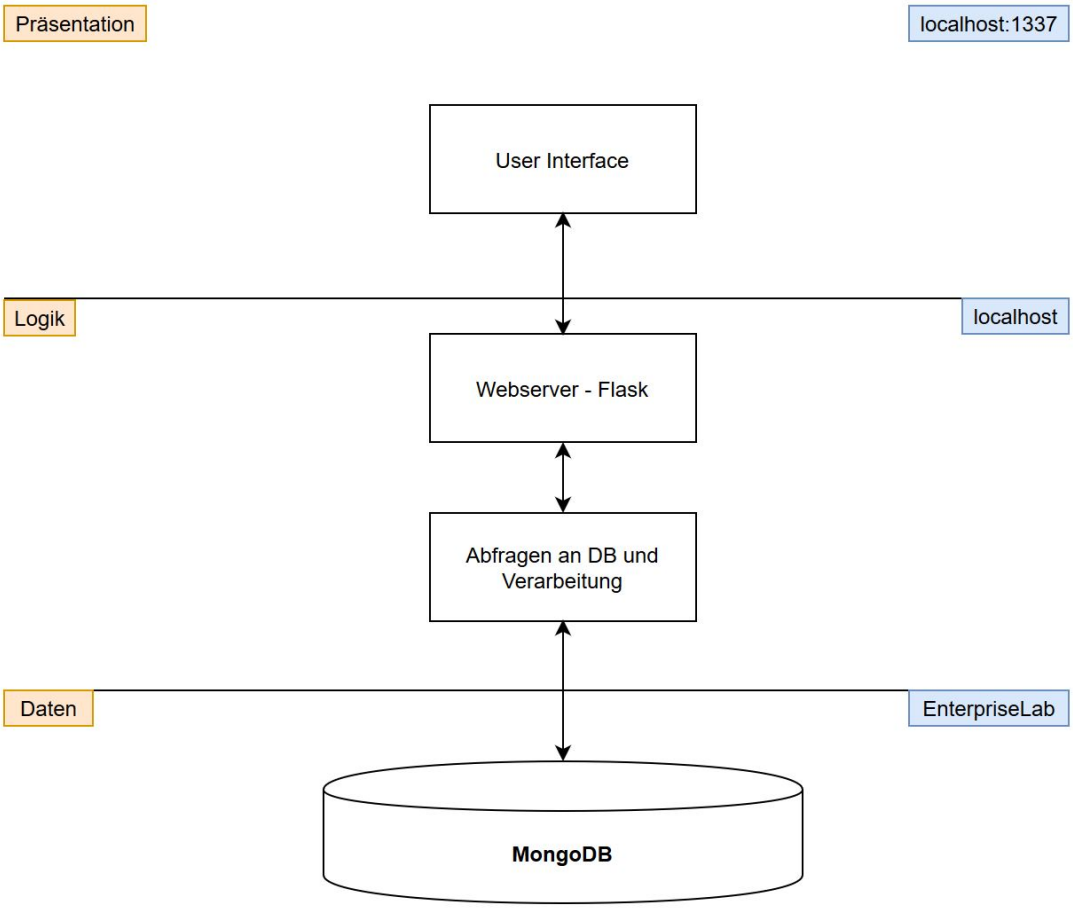



Abbildung 1: Architektur des Projektes,
Quelle: Autoren

# 3 Datenmodellierung

## 3.1 Datenmodell

Wie bereits erwähnt, wird im vorliegenden Projekt auf eine NoSql-Lösung mit MongoDB zurückgegriffen. Die Verwaltung der Daten erfolgt daher im JSON-Format. Grundsätzlich umfasst das vorliegende Projekt zwei Collections (vgl. Abbildung 2 auf Seite 10). Das ist zum einen „videos“ und zum anderen „categories“. Die Collection „videos“ umfasst entsprechende Daten von Trendvideos, während „categories“ die von YouTube vorgegebenen Kategorien enthält. Da die Collection „videos“ aus sehr vielen Attributen besteht, wurde entschieden, diese innerhalb der JSON-Struktur zu verschachteln. Deshalb sind nun zusätzlich die Objekte „channel“, „settings“ und „interactions“ der Collection „videos“ untergeordnet. Dadurch erhält das JSON-Dokument eine bessere Struktur.

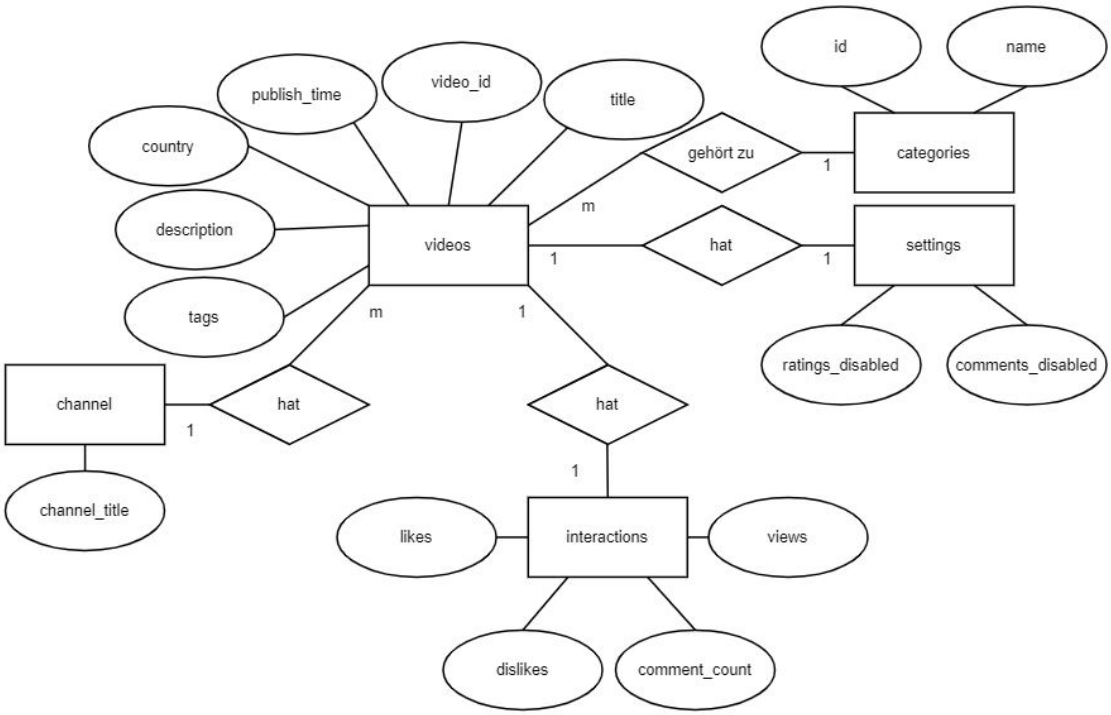



Abbildung 2: Modellierung UML,
Quelle: Autoren

Zusätzlich erstellte das Projektteam in Anschluss an den Input zu Dokumentdatenbanken noch ein UML-Diagramm (vgl. Abbildung 3 auf Seite 11), um die Collections sowie die Beziehung der untergeordneten Objekte zu modellieren.

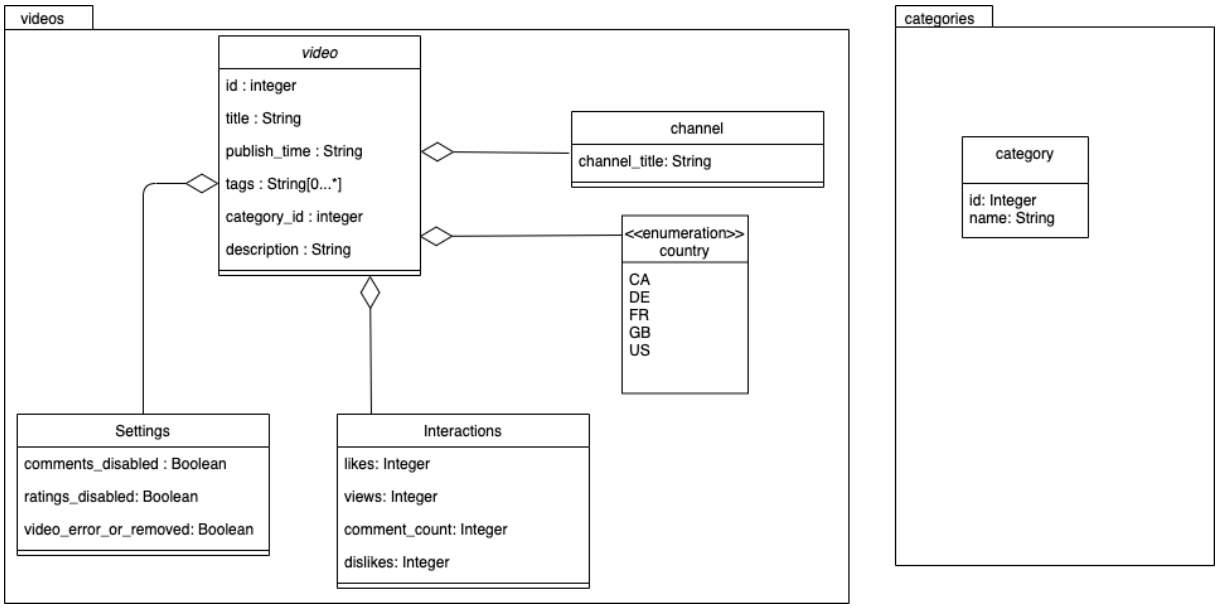



Abbildung 3: UML-Diagramm,
Quelle: Autoren

## 3.2 Datenbankschema

Zur Strukturvalidierung des JSON-Dokuments dient ein durch das Projektteam erstelltes JSON-Schema. Dabei erfolgte eine Übersetzung des Datenmodells in das entsprechende Schema. Dieses ist in der folgenden Abbildung (vgl. Abbildung 4 auf Seite 12) ersichtlich.

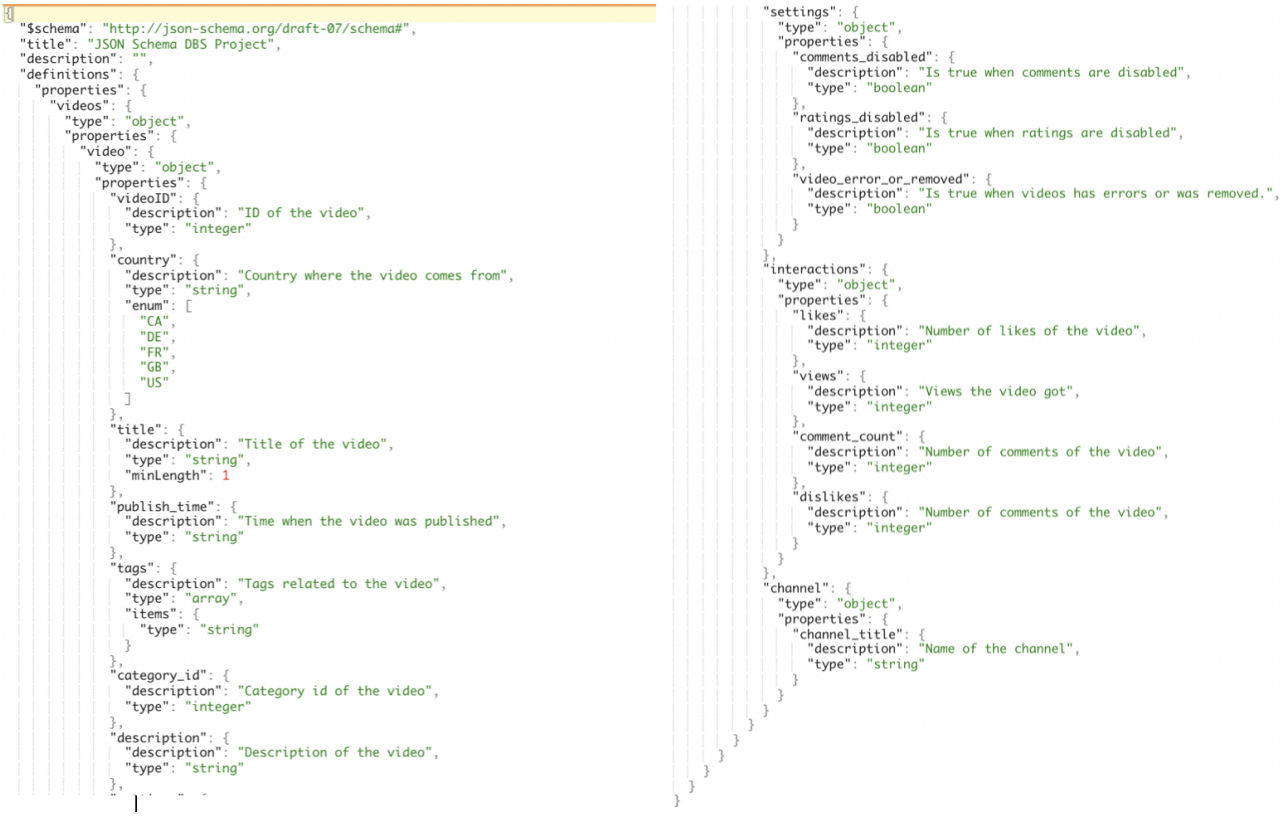



Abbildung 4: JSON-Schema,
Quelle: Autoren

# 4 Datenbanksprache

Das Projektteam entschied sich, als Abfragesprache Python zu nutzen. Denn Python bietet eine umgängliche API namens Pymongo für die Kommunikation mit MongoDB an.

In diesem Abschnitt werden die Abfragen, deren Logik und die eingesetzte Sprache behandelt. Zusätzlich wird gezeigt, wie die Anforderungen gemäss dem Projektauftrag (Selektion, Projektion, Gruppierung, Aggregation etc.) umgesetzt wurden. Die Resultate und Rückgaben der Abfragen werden jedoch erst im Abschnitt Resultate (vgl. Abschnitt 6 auf Seite 22) erläutert.

## 4.1 Verbindungsaufbau

Der Verbindungsaufbau zur MongoDB in Python (vgl. Abbildung 5 auf Seite 13) gestaltet sich einfach. Zunächst gilt es, von Pymongo das Modul „MongoClient“ zu importieren. Über „MongoClient“ lässt sich schliesslich unter der Angabe vom Protokoll (mongodb), der IP-Adresse und dem Port ein Client generieren. Von diesem kann schliesslich die Datenbank (hier „DBSFS2020“) sowie die Collection (hier „videos“) angesprochen werden.

```
from pymongo import MongoClient
from pprint import pprint
import logging

class QueryLogic:

    def __init__(self):
        remote_db = ConnectToDB()

        try:
            client = MongoClient('mongodb://10.177.122.15:27017/')
            client.server_info()
        except:
            print('Connection Error')

        db = client['DBFS2020']
        client.server_info()
        self.collection = db['videos']
```

Abbildung 5: Aufbau Datenbankverbindung,
Quelle: Autoren

## 4.2 Join

In relationalen Datenbanken ist „Join“ dafür zuständig, Tabellen zusammenzufügen. Im Fall von MongoDB geht es darum, Collections miteinander zu verbinden. In PyMongo wird dies über folgenden Befehl erreicht:

collection.aggregate([
{’$lookup’ : {...}}
])

Innerhalb der geschweiften Klammern nach „lookup“ findet schliesslich das Zusammenfügen der Collections statt. Im vorliegenden Projekt finden „Joins“ ausschliess-

lich bei der Gruppierung nach Kategorien Anwendung. Denn die durch die Datenquelle (Kaggle) offerierten Daten enthalten lediglich die ID der Kategorie, jedoch nicht deren Namen. Eine ID der Kategorie alleine ist nicht sehr aussagekräftig. Der Anbieter der Daten hat die zugehörigen Namen auf Kaggle leider nicht veröffentlicht. Daher wurden diese im Internet (Dinesh Prathap, 2015) gesucht, auf deren Richtigkeit überprüft und schliesslich in einer separaten Collection abgespeichert. Damit ist es möglich, mittels eines „lookup“ bei der Gruppierung nach Kategorie den Namen der Kategorie anstelle der ID auszugeben.

Da das Projekt generell sehr viele Abfragen enthält, wird nicht jede im Quellcode grafisch dargestellt. Daher ist untenstehend lediglich eine davon visualisiert (vgl. Abbildung 6 auf Seite 14):

```
# WHICH CATEGORY HAS THE MOST LIKES COMPARED TO VIEWS?
def calculate_categoryMostLikesComparedToViews(self):
    Category_MostLikesComparedToViews = self.categories.aggregate([
        {'$match': {}},
        {'$lookup':
            {
                'from': 'videos', 'localField': '_id', 'foreignField': 'category_id', 'as': 'category'
            }
        },
        {'$unwind':
            {
                'path': '$category',
                'preserveNullAndEmptyArrays': False
            }
        },
        {'$group': {
            '_id': '$name',
            'Likes': {'$sum': '$category.interactions.likes'},
            'Views': {'$sum': '$category.interactions.views'}
        }
        },
        {'$project':
            {
                'Ratio': {'$cond': [{'$eq': ['$Views', 0]}, 'N/A', {
                    '$multiply': [{'$divide': ['$Likes', '$Views']}, 100]}]},
                'Likes': 1,
                'Views': 1,
            }
        },
        {'$sort': {'Ratio': -1}}
    ])
```

Abbildung 6: Umfangreiche Abfrage,
Quelle: Autoren

## 4.3 Aggregation

Aggregationen sind dafür da, Auswertungen basierend auf spezifischen Attributen vorzunehmen. Beispiele sind arithmetische Operationen, der Durchschnitt, das Maximum oder das Minimum. Es gibt natürlich noch viele weitere. Die folgende Auflistung (vgl. Tabelle 1 auf Seite 15) zeigt, wie einige solcher Aggregationen in PyMongo aussehen. Aggregationen sind ein wichtiger und grosser Bestandteil der Abfragen im Projekt. In vielen Abfragen ist ein Verhältnis zweier Attribute als Resultat vorgesehen. Verhältnisse sind insofern relevant, dass sie sich besser dazu eignen Resultate zu vergleichen. Möchte man beispielsweise wissen, welche Youtube-Kategorie die meisten Likes hat, ist es ungenau, lediglich das entsprechende Attribut Likes für eine Kategorie aufzusummieren. Angenommen in der einen Kategorie existieren zehn

| Befehl | Bedeutung |
|---|---|
| $sum | Summe |
| $divide | Division |
| $multiply | Multiplikation |
| $avg | Durchschnitt |
| $max | Maximum |
| etc. | ... |

Tabelle 1: Aggregation: Beispiele

mal so viele Videos wie in einer anderen Kategorie, dann ist vermutlich deren Anzahl Likes auch höher. Deshalb ist ein Verhältnis zwischen den Attributen wichtig. Beispielsweise wäre ein Ansatz, die aufsummierte Anzahl Likes durch die Anzahl existierender Videos in einer Kategorie auszuwerten. Dann wäre die Kennzahl: „Anzahl Likes pro Video innerhalb einer Kategorie“. Durch solche Verhältnisse sind schliesslich Vergleiche von Kategorien viel aussagekräftiger. Die Aggregation ist eine wichtige Funktion, Verhältnisse zu ermitteln. Um auf das Beispiel „Anzahl Likes pro Video innerhalb einer Kategorie“ zurückzukommen: Hierfür ist zunächst die Summe aller Likes sowie die Menge an Videos innerhalb einer Kategorie zu ermitteln. Dafür eignet sich die Aggregation „$sum“. Das Verhältnis (prozentual) lässt sich nun noch wie folgt mit einem einfachen Dreisatz berechnen:

$$\text{Verhältnis} = \frac{SummeVideos}{SummeLikes} \cdot 100$$

Da der Dreisatz lediglich aus Multiplikations- und Divisions-Operatoren besteht, hält die Aggregation auch hierfür eine Lösung bereit. Für die Division ist „$divide“ und für die Multiplikation „$multiply“ zu verwenden. Ein ähnliches Beispiel ist im Source-Code abgebildet(vgl. Abbildung 6 auf Seite 14). Dies ist nur ein Beispiel wie eine Aggregation umgesetzt wurde. Das Projekt umfasst jedoch weitaus mehr. Diese sind im Github-Repository ersichtlich (Projektteam, 2020).

## 4.4 Gruppierung

Gruppierungen dienen dazu, Attribute mit gleichem Wert zusammenzufassen. In Bezug auf das Projekt wäre dies beispielsweise eine spezifische Kategorie (bsp. „Film & Animation“). Gruppierungen sind in Pymongo über folgenden Befehl ausführbar:

```
collection.aggregate([
{'$group' : {...}}
])
```

Auf einer Gruppieren lassen sich gut Aggregationen anwenden (beispielsweise die Summe aller Likes innerhalb einer Kategorie). Gruppierung finden im Projekt primär

bei Abfragen auf Kategorien Anwendung. Im abgebildeten Beispiel (vgl. Abbildung 6 auf Seite 14) ist eine solche Gruppierung ersichtlich. Dort wird zunächst „self.collection.aggregate“ aufgerufen. Wie bereits beim Abschnitt Aggregation (vgl. Abschnitt 4.3 auf Seite 14) gezeigt, ist dies relevant, damit Daten entsprechend aggregiert werden können. Anschliessend folgt die Gruppierung („$group“). Darin wird mittels „id“ bestimmt, welches Attribut für die Gruppierung relevant ist (hier „category_id“). Im Anschluss folgen die Aggregationen („Summe der Likes“ sowie „Summe der Views“) basierend auf der Gruppierung. Anschliessend erfolgt die Projektion der Daten, die im nächsten Abschnitt (vgl. Abschnitt 4.5 auf Seite 16) vertieft wird. Um die Gruppierungen noch genauer zu sehen, wird auch hier ein Einblick in das Github-Repository empfohlen (Projektteam, 2020).

## 4.5 Selektion und Projektion

Im relationalen Datenbankumfeld steht die Selektion für das Auswählen spezifischer Tabellenzeilen, während die Projektion die gewünschten Spalten abbildet. Solche Funktionen lassen sich auch durch Pymongo in den Collections erreichen.

Die Projektion erfolgt durch den Befehl „$project“ und filtert die gewünschten Attribute aus der Collection. Das Beispiel (vgl. Abbildung 6 auf Seite 14) zeigt die Projektion auf drei Attribute (Ratio, Likes, Views). Dabei ist auch gut ersichtlich das „Ratio“ neu durch eine Rechnung generiert wird. Die Rechnung besagt grundsätzlich, dass das Verhältnis (prozentual) zwischen „Views“ und „Likes“ zu berechnen ist. Falls das Attribut „Views“ null ist, ist „N/A“ auszugeben, da „durch-null-teilen“ mathematisch nicht definiert ist. Man sieht also, dass Attribute auch während der Abfrage dynamisch erstellbar sind.

Die Selektion nutzt den Befehl „$match“ und gibt lediglich die Objekte aus, die eine bestimmte Bedingung erfüllen. Auch hierfür ist ein Beispiel abgebildet (vgl. Abbildung 7 auf Seite 16). Dieser Befehl sagt aus, dass alle Videos auszugeben sind, die in ihren Einstellungen spezifische Werte (true, false) haben.

```
        pipe_cfrt = [{'$match': {'settings.comments_disabled': False, 'settings.ratings_disabled': True
}},
                     {'$group': {'_id': None, 'Views': {'$sum': '$interactions.views'}}}]
        pipe_ctrf = [{'$match': {'settings.comments_disabled': True, 'settings.ratings_disabled': False
}},
                     {'$group': {'_id': None, 'Views': {'$sum': '$interactions.views'}}}]
        pipe_ctrt = [{'$match': {'settings.comments_disabled': True, 'settings.ratings_disabled': True
}},
                     {'$group': {'_id': None, 'Views': {'$sum': '$interactions.views'}}}]
```

Abbildung 7: Selektion,
Quelle: Autoren

## 4.6 Verschachtelung von Queries

Auch die Verschachtelung der Queries (vgl. Abbildung 6 auf Seite 14) wurde in den meisten Abfragen genutzt. In der referenzierten Abbildung sieht man beispielsweise, dass innerhalb der Projektion, die entsprechende Ausgabe noch einmal manipuliert wird. Das Resultat verändert sich also insofern, dass von den gruppierten und zusammengefassten Resultaten das prozentuale Verhältnis ermittelt wird. Auch hier

ist lediglich ein Beispiel abgebildet. Weitere Verschachtelungen sind ebenfalls im Github-Repository (Projektteam, 2020) ersichtlich.

# 5 Systemarchitektur

## 5.1 Aufbau, Installation Server

Damit alle Teammitglieder mit der Datenbank arbeiten können, wurde hierfür eine Ressource auf dem Enterpriselab bezogen. Die Ressource ist unter folgender URL über SSH erreichbar, wobei über den angegebenen Port auf die Datenbank zugegriffen werden kann:

- URL: dbs-f20-iffische.el.eee.intern
- IP-Adresse: 10.177.122.15
- Port: 27017

Bei der Installation hat sich das Projektteam an die Anleitung der offiziellen Webseite von MongoDB gehalten (mongoDB, 2020b). Dafür wurde die aktuellste Version „MongoDB 4.2 Community Edition“ installiert. Nach der erfolgreichen Installation ist der Zugriff auf die Datenbank in der Shell über den Befehl „mongo“ möglich.

Auf die Erstellung und Einrichtung der Datenbankinstanz wird im nächsten Kapitel (vgl. Abschnitt 5.2 auf Seite 18) genauer eingegangen.

## 5.2 Quelle und Import der Daten

Wie bereits im Use Case erwähnt, wurde als Quelle ein Datensatz genommen, welcher durch Mitchell Jolly (Mitchell J, 2020) durch ein eigens erstelltes Tool gesammelt wurde. Er stellt diesen (Mitchell J, 2019) frei zur Verfügung und baut ihn von Zeit zu Zeit mit neuen Daten aus. Dieser Datensatz beinhaltet Daten aus USA, Grossbritannien, Deutschland, Kanada, Frankreich, Russland, Mexiko, Süd-Korea, Japan und Indien und ist insgesamt 514 MB gross. In diesem Projekt hat sich das Projektteam jedoch auf die Daten aus USA, Kanada, Grossbritannien, Deutschland und Frankreich eingeschränkt. Die Daten sind online im CSV-Format erhältlich und enthalten folgende Attribute (vgl. Tabelle 2 auf Seite 18): Um die Daten vom CSV-

| | | | |
|---|---|---|---|
| video_id | trending_date | title | channel_title |
| category_id | publish_time | tags | views |
| likes | dislikes | comment_count | thumbnail_link |
| comments_disabled | ratings_disabled | video_error_or_removed | description |

Tabelle 2: Attribute Video-Collection

Format in das JSON-Format zu konvertieren, wurde in Python ein Parser geschrieben. Der Parser wandelt die Daten Schema-konform (vgl. Abschnitt 3.2 auf Seite 11) um und speichert diese als JSON-Datei. Auf der MongoDB-Instanz wurde dann eine Datenbankinstanz namens „DBSF20“ mit einer entsprechenden Collection „videos“ erstellt. Darin wurden die einzelnen Objekte als Dokumente angelegt. Durch Eigenrecherche erstellte das Projektteam zusätzlich eine Collection namens „categories“, um die Kategorien von Youtube-Videos festzuhalten.

## 5.3 Optimierung Performance

### 5.3.1 Schema

Eine erste Art, wie sich eine Steigerung aus Sicht der Performance erreichen liesse, wäre das Ändern des vorhandenen Schemas und das Einlesen der Daten. Bisher werden die Videodaten sowie die Kategorien jeweils über ein „$lookup“ zusammengeführt. Wie bereits erwähnt, ist das „$lookup“ mit einem „Join“ in SQL zu vergleichen. Es ist bekannt, dass solche Ausführungen generell viel Zeit beanspruchen. Man könnte also Kategorien jeweils direkt in der Collection „videos“ festhalten. Dadurch müsste man kein „$lookup“ mehr durchführen. Auf der Gegenseite hätte man mehr Redundanz und würde mehr Speicher benötigen. Da gemäss Aufgabenstellung jedoch ein „Join“ verlangt wird, wurde bislang auf die Umstellung auf eine Collection verzichtet. Zudem liegt die Performance aus Sicht des Projektteams aktuell noch im grünen Bereich. Würden mehr Daten verarbeitet werden müssen, sollten die Collections „videos“ und „categories“ zusammengeführt werden. Dafür wäre dann aber auch eine Anpassung des Parsers notwendig.

### 5.3.2 Queries

Bei den Queries fand das Projektteam eine Lösung, wie diese beim Aufruf von „$lookup“ zu optimieren sind. Dafür bietet MongoDB die Aggregation „$unwind“ (vgl. Abbildung 6 auf Seite 14) an. Die Theorie besagt (mongoDB, 2020a), dass wenn ein „$unwind“ unmittelbar auf ein anderes „$lookup“ folgt und das „$unwind“ auf dem „as-Feld“ des „$lookup“ ausgeführt wird, kann der Optimierer das „$unwind“ in der „$lookup“-Phase zusammenführen. Dadurch wird vermieden, dass grosse Zwischendokumente erstellt werden welche den Arbeitsspeicher unnötig auslasten. Würde man kein „$unwind“ nutzen, müsste man den Ausdruck „{allowDiskUse: True}“ verwenden, was die Performance beeinträchtigen würde.

Es gäbe noch weitere Arten wie sich Queries generell optimieren lassen (mongoDB, 2020a). Im Kontext der Queries im Projekt konnten diese jedoch nicht genutzt werden.

### 5.3.3 Indexing

Wie bei SQL ist das Setzen von Indexen auch bei MongoDB möglich (Data Pilot, 2019). Dafür gibt es verschiedene Arten wie „Single-Field Index“, „Multikey Index“, „Hashed Indexes“ etc., die verwendet werden könnten. Wie bereits erwähnt, bewegt sich die aktuelle Zeit für die Datenabfragen in einem akzeptablen Rahmen und bedarf vorerst keiner besonderen Optimierung, zumal in den Queries (vgl. Abschnitt 5.3.2 auf Seite 19) bereits eine Verbesserung vorgenommen wurde. Zudem ist der Datensatz nicht statisch, sondern wird hin und wieder ergänzt. Falls das Projekt weiterverfolgt würde, müsste man somit das Indexing mit jedem neuen Datenimport erneut ausführen, was ebenfalls ein wenig Zeit kostet. Deshalb wird vorgeschlagen, Indexe erst zu verwenden, wenn diese aus Performancegründen nötig sind.

### 5.3.4 Hardware

Eine letzte Möglichkeit, die für die Optimierung der Performance im vorliegenden Projekt in Betracht gezogen wird, sind Anpassungen an den Hardwareressourcen. Dafür wäre eine Kontaktaufnahme mit dem Enterpriselab nötig, da die Daten dort abgespeichert sind.

## 5.4 Sicherheitsaspekte

Die Realisierung bzw. Umsetzung der Sicherheitsaspekte ist nicht Bestandteil dieses Projektes und wurde auch nicht verlangt. Trotzdem wurde sich darüber Gedanken gemacht und in der Theorie ein Konzept erarbeitet sowie diese gar auch teilweise umgesetzt. In diesem Kapitel wird auf die einzelnen Sicherheitskomponenten eingegangen und erläutert, wie diese effektiv umgesetzt oder in der Theorie konzeptionell erarbeitet wurden. Wichtig zu erwähnen ist, dass der Zugriff auf die Daten eher uninteressant für einen Angreifer ist, da diese von einer öffentlichen Datenquelle bezogen wurden, welche wir auch beim Webauftritt entsprechend referenziert haben.

### 5.4.1 Netzwerk

Die Datenbank wurde auf einer VM installiert, welche sich in der DMZ des Enterpriselabs der Hochschule Luzern befindet. Somit ist diese bis und mit dem OSI Layer 6 bereits, in Bezug auf allen entsprechenden kritischen Angriffspunkten, abgedeckt und die Verantwortung diesbezüglich dem Enterpriselab übergeben. Aus diesem Grund wurden auf der Netzwerkebene keine weiteren Einschränkungen eingerichtet. Jeder, der sich in der DMZ befindet, hat Zugriff auf diesen. Des Weiteren wurde der MongoDB-Server mit dem Standard-Port 27017 eingerichtet, was grundsätzlich auch nicht in Frage käme, da diese bei einem Angriff immer als erstes ausprobiert werden.

Ziel des Projektes ist es, die evaluierten Daten der Öffentlichkeit zur Verfügung zu stellen. Um dies zu ermöglichen, müsste der Server ausserhalb der DMZ zugänglich sein. Hierfür würde man bei einem entsprechenden Cloud-Anbieter (Platform as a Service) eine entsprechende Dienstleistung beziehen, um den Datenbankserver darauf zu betreiben.

### 5.4.2 Usermanagement

Auf dem Server wurde kein Usermanagement realisiert. Darauf wurde aus zeitlichen Gründen verzichtet. In der Theorie wurde jedoch folgendes überlegt: Für alle Besucher des Webauftrittes, die Abfragen auf der Datenbank machen wollen, besteht lediglich ein lesender Zugriff (kein Delete, Update, etc.). Dies würde mit einem entsprechenden User ermöglicht werden. Rechte für Änderungen werden lediglich einem Administratorkonto erteilt. Mit dem Administratorkonto kann man sich hingegen nicht über den Webauftritt einloggen, sondern lediglich über die zur Verfügung gestellte Plattform, auf dem sich der Datenbankserver befindet (bsp. über SSH). Auf die technischen Aspekte im OSI Layer 7 wird im darauffolgenden Kapitel weiter eingegangen.

### 5.4.3 Applicationlayer

Da bis zum OSI Layer 6 durch den Plattformbetreiber alles abgedeckt wird, bleibt nur noch der letzte OSI Layer, auf dem auch das HTTP-Protokoll läuft, übrig. Wie bereits erwähnt, besteht über den Webauftritt lediglich ein lesender Zugriff auf die Datenbank. Es steht somit kein Loginfenster zur Verfügung. Die gewünschten Datenbankabfragen werden beim Webauftritt über Dropdown-Menüs als Auswahl zur Verfügung gestellt. Auch hier wird auf ein Eingabefeld verzichtet. Nichtsdestotrotz wäre es aber auch hier noch möglich den HTTP-Header zu verändern und so weiteren Schadcode einzufügen. Da aber im Backend lediglich bestimmte Funktionen, ohne Parameterübergabe, aufgerufen werden, besteht hierfür keine Gefahr für eine Injection. Bei der Recherche über Injections auf MongoDB-Servern stellte sich des Weiteren heraus, dass sogenannte SQL-Injections in dieser Technologie kein Problem darstellen, da die Queries als JSON-Objekte und nicht als Strings angelegt werden. Das bedeutet aber nicht, dass die MongoDB immun gegen solche Arten von Angriffen ist. Es ist immer noch möglich bspw. JavaScript-Code auf diesem einzuschleusen und auszuführen. Auch hierfür kann man auf dem Server einfach JavaScript ausschalten, was in diesem Projekt möglich wäre, da im Backend nicht damit gearbeitet wird.

# 6 Resultate

Die Resultate sind in zwei Abschnitte gegliedert. Zunächst wird der Mehrwert für den Nutzer (Youtuber) aufgezeigt und mitgeteilt, wie die Webseite genutzt werden kann. Anschliessend folgen noch Auswertungen und Erkenntnisse des Projektteams.

## 6.1 User-Story

Für das Projekt ist grundsätzlich angedacht, dass der Youtuber die Möglichkeit hat, seinen Account über eine durch das Projekt erstellte Webseite zu analysieren. Damit ergibt sich die Möglichkeit, den eigenen Account mit entsprechenden Richtwerten zu vergleichen und Empfehlungen zu erhalten. Die Analyse ist grob in generelle und kategoriespezifische Resultate unterteilt. Bei den generellen Resultaten werden unterschiedlichste Attribute (Views, Likes, Comments etc.) miteinander verglichen und ausgewertet. Dadurch erhält man eine Übersicht über die gesamte Spannweite. Kategoriespezifisch sind ebenfalls Richtwerte abfragbar. Diese sollen dazu dienen, seinen Account noch konkreter einzuschätzen. Denn zwischen den Kategorien gibt es bei gewissen Kennzahlen teilweise grosse Abweichungen. Kategorienspezifische Abfragen eignen sich aber auch für Einsteiger, die nicht wissen, in welchem Bereich sie Videos machen möchten. Denn durch gewisse Abfragen (beispielsweise: „welche Kategorie erhält die meisten Views pro Video?“) geben auch Auskunft über die Lukrativität der Kategorien.

In der folgenden Aufzählung sind alle Anfragen aufgelistet:

1. **Do dislikes have a negative impact on views? (Sorted by Ratio)**
2. **Do dislikes have a negative impact on views? (Sorted by Views)**
3. **Do dislikes have a negative impact on views? (Sorted by Dislikes)**
4. **Are likes relevant for views? (Sorted by Ratio)**
5. **Are likes relevant for views? (Sorted by Views)**
6. **Are likes relevant for views? (Sorted by Likes)**
7. **Are comments relevant for views? (Sorted by Ratio**)
8. **Are comments relevant for views? (Sorted by Views)**
9. **Are comments relevant for views? (Sorted by Comment)**
10. **Which category has the most dislikes compared to views?**
11. **Which category has the most likes compared to views?**
12. **Which category has the most comments compared to views?**
13. **Which category gets the most views per video?**
14. **Which category has the best ratio between likes and dislikes?**
15. **Which category has the highest amount of uploads?**

16. **What is the general ratio between likes and dislikes?**

Aus den genannten Fragen ergeben sich nun endlose Szenarien, wie diese nützlich sein können. Nimmt man als Beispiel eine Neueinsteigerin (junge Frau) bei Youtube, die noch nicht weiss, in welchem Bereich sie Videos machen möchte. Daher evaluiert sie mit Abfrage 13, in welcher Kategorie sie statistisch gesehen am meisten Views generiert. Anschliessend prüft Sie mit der Abfrage 16, wie hoch die Konkurrenz unter den jeweiligen Kategorien ist. Dann entscheidet sie sich für eine Kategorie und beginnt, Videos zu veröffentlichen. Nach einem halben Jahr ist sie entmutigt und überlegt, den Kanal wieder abzubrechen. Sie hat das Gefühl, dass sie sehr viele Dislikes erhält und daher zu viele Leute ihr Videomaterial nicht mögen. Bevor sie sich aber definitiv beschliesst aufzugeben, besucht sie erneut die Webseite und analysiert ihren Account. Zunächst nutzt sie die Abfrage 1, um die Verhältnisse zwischen Dislikes und Views sortiert nach dem Verhältnis zu sehen. Sie merkt, dass sie etwa im Durchschnitt liegt. Das Resultat dürfte also besser sein, ist jedoch lange nicht so schlecht, wie sie es angedacht hat. Sie führt nun noch die Abfrage 2 aus, um zu sehen, was für ein Verhältnis die meistgesehenen Videos haben. Hier macht sich bemerkbar, dass auch solche Videos mit negativer Kritik umgehen müssen, obwohl sie viel Erfolg haben. Nun wird sie langsam erleichtert und ist froh, dass sie die Webseite aufgerufen hat. Ihr wird nun bewusst, dass Zuschauer, dazu tendieren mehr negatives Feedback zu geben, als sie es gedacht hätte. Um sich aber vollständig abzusichern, nutzt sie nun noch Abfrage 10, um kategoriespezifisch die Dislikes zu analysieren. Hier merkt sie nun, dass sie sogar über dem Durchschnitt liegt und ihr Verhältnis daher positiv ist. Sie weiss nun, dass ihre Kategorie verglichen zu anderen mit mehr negativer Kritik rechnen muss. Sie ist nun überzeugt, weiterzumachen.

Damit wurde aufgezeigt, wie die Webseite genutzt werden könnte. Dabei liessen sich noch unzählige weitere Szenarien ausdenken. Ein Beispiel wie das Resultat aussehen könnte ist im Anhang (vgl. Abbildung 16 auf Seite 34) ersichtlich. Dabei wird die Abfrage 1 visualisiert.

## 6.2 Auswertung Projektteam

Wie im vorherigen Abschnitt erwähnt, liegt der grosse Mehrwert an der Arbeit daran, dass jeder seinen Account selber analysieren kann. Der Youtuber kann somit selbst entscheiden, was für ihn interessant und wichtig ist. Das Projektteam hat die Daten jedoch selbst noch analysiert und dabei einige Erkenntnisse festgehalten. Dafür werden zunächst generelle Resultate und dann kategoriespezifische Resultate aufgezeigt.

### 6.2.1 Generelle Erkenntnisse

**Verhältnis von Likes zu Dislikes**
Likes und Dislikes sind in der Welt von Social Media Plattformen omnipräsent. Der Tenor hört sich dann oft wie folgt an. Möglichst viele Likes und keine Dislikes. Jedem ist klar, dass das Wunschdenken ist. Interessanter wird es, wenn das Verhältnis von Likes zu Dislikes von allen Videos aus dem Datensatz betrachtet wird. Durchschnittlich sind 5% (vgl. Abbildung 8 auf Seite 24) aller Bewertungen Dislikes also negativ.

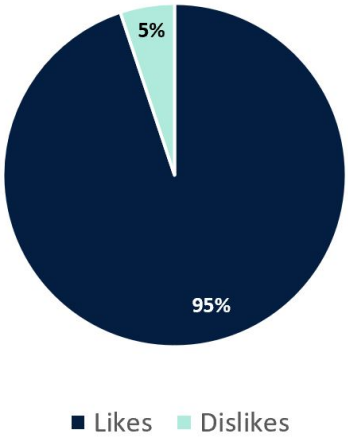



Abbildung 8: Verhältnis Likes zu Dislikes,
Quelle: Autoren

**Verhältnis von Views zu Dislikes**
Man stellt sich des öfteren die Frage, ob die Anzahl Dislikes, nebst den Likes, eine Auswirkung auf die Anzahl Views hat. Diesen Zusammenhang wollte das Projektteam weiter untersuchen und hat dabei die entsprechenden Verhätnisse nach Dislikes, Views und der Ratio (Verhältnis zwischen Views und Dislikes) sortiert und ausgewertet. Jede Sortierung sagt etwas anderes aus. Bei der Sortierung nach Views sieht man beispielsweise, wie das Verhältnis bei den meistgesehenen Videos ist. Bei der Sortierung nach Ratio erkennt man wiederum was die besten und schlechtesten Verhältnisse sind, während man bei der Sortierung nach Dislikes noch sieht, ob Videos mit vielen Dislikes effektiv auch ein hohes Verhältnis haben oder nicht. Um zusätzlich einen groben Richtwert zu erhalten, wurde jeweils der Durchschnitt sowie der Median berechnet. Der Median ist sehr tief mit 0.09% Dislikes pro Anzahl Views (vgl. Abbildung 9 auf Seite 24). Das heisst es fallen im Allgemeinen nicht sehr viele Dislikes an.

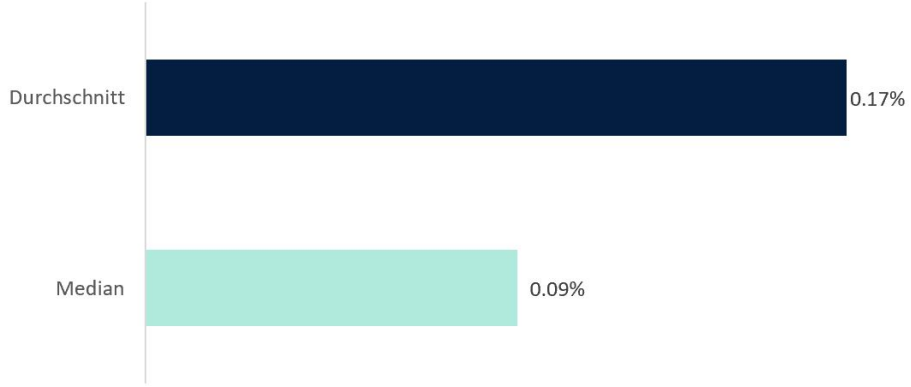



Abbildung 9: Dislikes pro Views,
Quelle: Autoren

- **Sortierung nach Ratio:** Hier ist die Spannweite der Erstplatzierten sehr gross. Auf dem ersten Platz landet ein Video mit gerade einmal 3000 Views

währenddem der Viertplatzierte über 600'000 Views enthält. Aus diesem Grund ist es hierbei schwierig eine signifikante Schlussfolgerung zu ziehen.

- **Sortierung nach Views:** Videos mit einer hohen Anzahl Views zeigen eine sehr tiefe Anzahl Dislikes auf, was auch zu erwarten war. Wenn ein Video viele Views hat, kann man schlussfolgern, dass das Resultat eher positiv daherkommt.
- **Sortierung nach Dislikes**: Videos mit den höchsten Anzahl Dislikes zeigen einen geringen, aber trotzdem auffäligen Ratio auf. Dieser befindet sich bei allen ca. um 4.3% wobei die Videos aber auch eine sehr hohe Anzahl Views aufweisen. Sie liegen also massiv über dem Wert des Medians und es handelt sich klar um Ausreisser.

**Verhältnis von Views zu Likes**
Hier wurden die entsprechenden Verhältnisse zwischen Views und Likes berechnet und ebenfalls nach den drei Eigenschaften Views, Likes und Ratio sortiert und ausgewertet. Zudem wurde erneut der Median und Durchschnitt berechnet. Der Median liegt hier bei 2.49% (vgl. Abbildung 10 auf Seite 25). Man erkennt also sofort, dass mehr Likes als Dislikes anfallen, was auch bereits die Betrachtung des Verhältnisses von Likes zu Dislikes gezeigt hat.

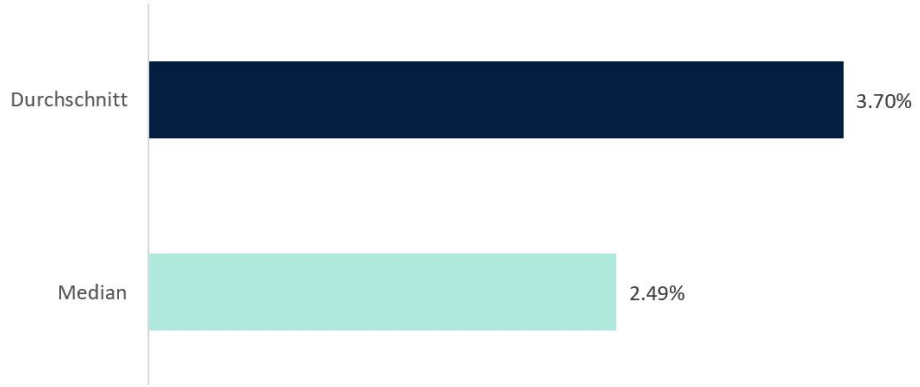



Abbildung 10: Likes pro Views,
Quelle: Autoren

- **Sortierung nach Ratio:** Die Spannweite zwischen den Views hierbei ist gross (zwischen 1'000 und 250'000). Aus diesem Grund konnte nicht direkt eine Schlussfolgerung daraus gezogen werden, da sich kein Muster erkennen lässt.
- **Sortierung nach Views:** Hier verhalten sich alle Ratios ungefähr gleich (ca. 0.6). Das ist verhältnismässig tief. Bei einer sehr hohen Anzahl Views kann man also nicht zwangsweise schlussfolgern, dass Videos auch eine hohe Anzahl Likes besitzen.
- **Sortierung nach Likes:** Hier schwanken die Ratios zwischen 4 und 5%. Dieses Verhalten war etwas unerwartet, da die Videos mit den meisten Dislikes auch sehr viele Views hatten. Das ist hier nicht der Fall.

Bei dem Vergleich dieser Sortierungen wurde bestätigt, dass die Anzahl Likes eine positive Auswirkung auf die Anzahl Views haben kann, aber nicht zwingend muss. Die Videos mit den meisten Likes gehören nämlich nicht zu diesen mit den meisten Views. Somit sollte sich ein Youtuber nicht zu sehr auf seine Likes fokussieren, wenn er viele Views als Ziel hat.

**Verhältnis von Anzahl Kommentaren und Views**
Hier war nun noch das Verhältnis zwischen den Views und der Anzahl Kommentare relevant. Auch hier wurden die Resultate entsprechend nach Views, Anzahl Kommentaren und den Verhältnissen sortiert. Zudem wurde erneut der Median und der Durchschnitt berechnet, um einen groben Richtwert zu erhalten. In diesem Fall liegt der Median bei 0.31% (vgl. Abbildung 11 auf Seite 26) und ist somit um einiges höher als derjenige von den Dislikes pro Views. Es fallen also im Normalfall mehr Kommentare als Dislikes an.

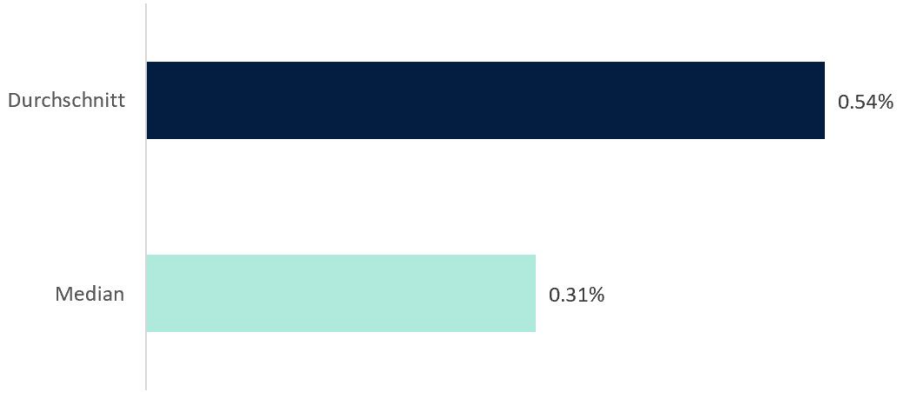



Abbildung 11: Kommentare pro Views,
Quelle: Autoren

- **Sortierung nach Ratio:** Bei den Videos mit den höchsten Verhältnissen sind erstaunlicherweise nur Videos mit einer Anzahl Views zwischen 20'000 und 46'000 vertreten. Das sind interessanterweise Videos mit sehr wenigen Views.
- **Sortierung nach Views:** Bei den Videos mit der höchsten Anzahl Views sind die entsprechenden Verhältnise ebenfalls sehr tief (um die 0.2%). Hier zeigte sich ebenfalls kein direkter Zusammenhang zwischen der Anzahl Kommentare und der Anzahl Views.
- **Sortierung nach Kommentaren:** Hier fielen diejenigen Videos mit einer sehr hohen Anzahl Views auf. Dennoch haben diese aber auch lediglich ein Verhältnis von ca. 3.6%, was auch hier erstaunlicherweise tief ausfiel. Denn es handelt sich immerhin um die Videos mit den meisten Kommentaren.

Es lässt sich hier nicht eindeutig zeigen, dass Kommentare eine sehr wichtige Rollen spielen, um viele Views zu erhalten. Dennoch sind viele Kommentare besser als keine. Dennoch ist es interessant zu erfahren, wie viel Interaktion man mit der Community etwa erreichen kann.

### 6.2.2 Kategorien

Viele Fragestellungen bezogen sich auf die Kategorien der Videos. Diese wurden durch diverse Abfragen analysiert und ausgewertet.

**Welche Kategorien haben die meisten Likes, Dislikes und Kommentare verglichen mit der Anzahl Views?**
Es ist sofort ersichtlich, dass am meisten Likes anfallen und etwas mehr Kommentare als Dislikes (vgl. Abbildung 12 auf Seite 27). Dies ist in dieser Form keine Überraschung. Interessanter wird es, wenn die prozentualen Werte etwas genauer betrachtet werden. Hier sticht ganz klar die Kategorie „Nonprofit & Activism“ heraus, deren Werte bei allen drei Parametern mit Abstand am höchsten sind. Dies lässt darauf schliessen, dass in dieser Kategorie ein reger und vor allem auch kontroverser Austausch stattfindet. Das konkrete Verhältnis von Likes und Dislikes wird in der nächsten Auswertung noch etwas vertieft. Ansonsten muss jeweils die Kategorie, an der man interessiert ist, betrachtet werden, um die richtigen Schlüsse ziehen zu können.

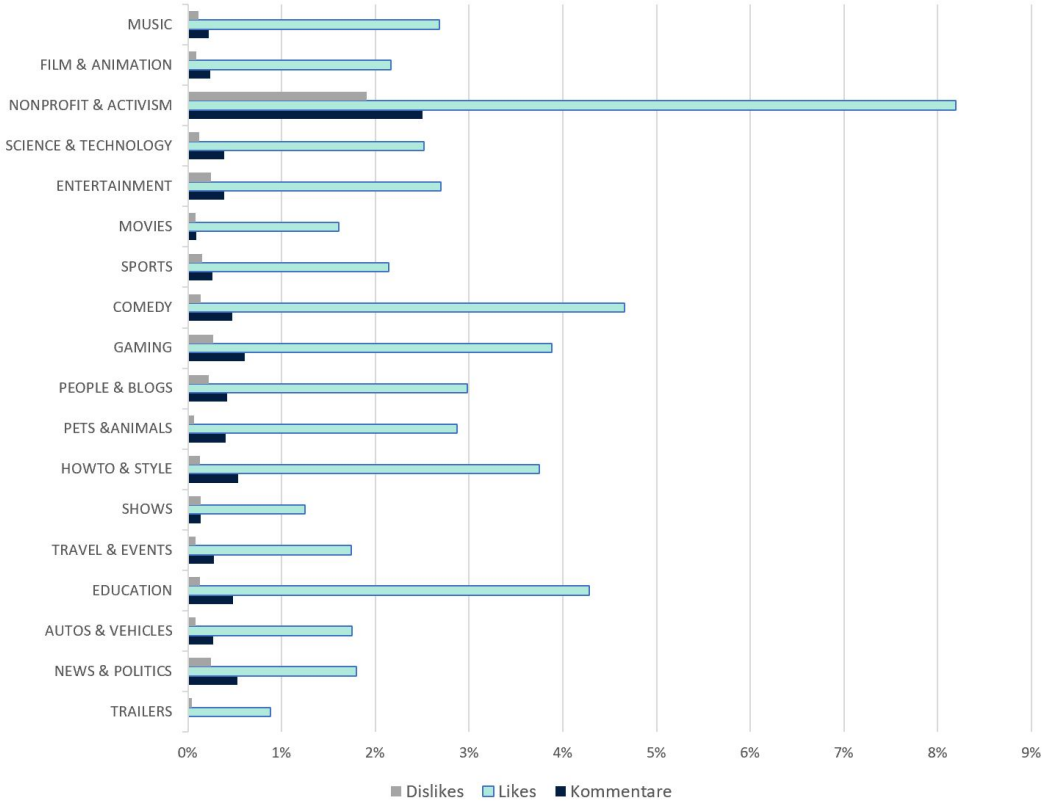



Abbildung 12: Likes, Dislikes und Kommentare pro Views nach Kategorien, Quelle: Autoren

**Welche Kategorien haben die höchste Anzahl Views pro Videos?**
Die Kategorie „Music“ belegt hier deutlich den ersten Platz (vgl. Abbildung 13 auf Seite 29). Die restlichen Werte nehmen laufend ab und die Kategorie „Trailers“ bildet das Schlusslicht mit den wenigsten Views pro Video. Zu beachten ist noch, dass in diesem Datensatz die Kategorie „Music“ eine sehr hohe Anzahl Videos beinhaltet (lediglich die Kategorie „Entertainment“ beinhaltet noch mehr Videos). Das heisst,

dass die Kategorie enorm viele Views generiert, trotz den sehr vielen Uploads. Weiter unten ist die Uploadmenge der einzelnen Kategorien noch aufgeschlüsselt und erläutert.

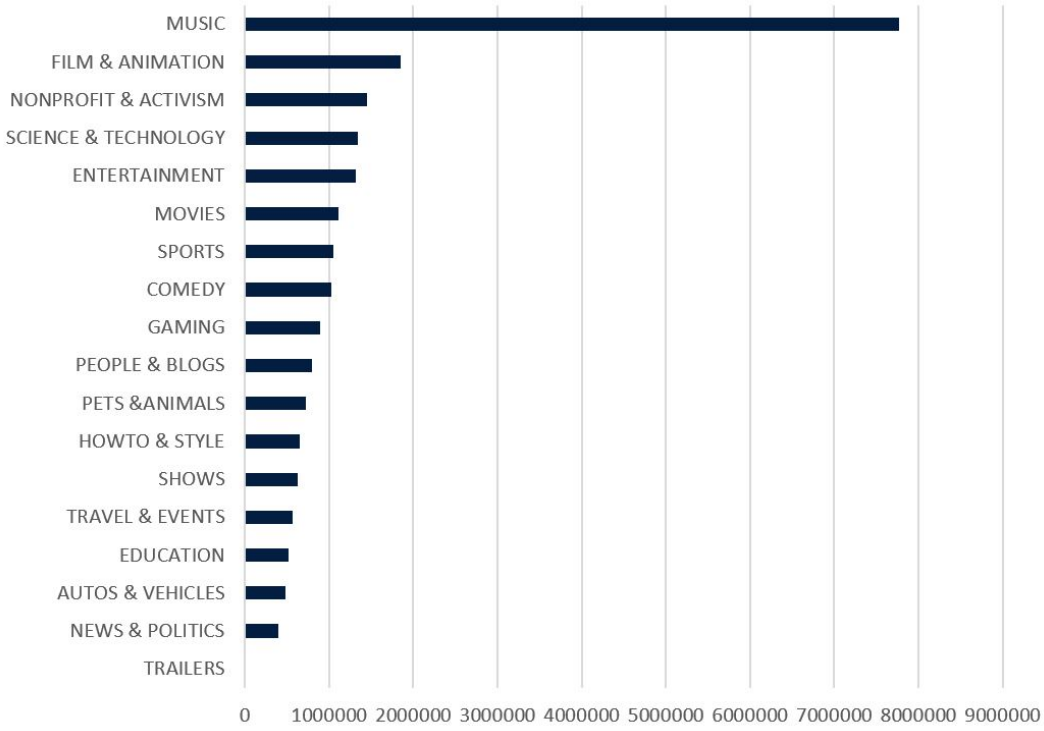



Abbildung 13: Views pro Video nach Kategorien,
Quelle: Autoren

**Welche Kategorie hat das beste Verhältnis zwischen Likes und Dislikes?** Hier erreicht die Kategorie „Pets and Animals“ den besten Wert. Nur gerade 2.5% (vgl. Abbildung 14 auf Seite 29) aller Bewertungen sind negativ. Auch hier gilt zu beachten, dass die ersten zehn Kategorien dicht aneinander gereiht sind und sich die Werte nur leicht unterscheiden. Bei der Kategorie „Nonprofit & Activism“ sind dann fast ein Viertel der Bewertungen negativ. Spannend ist an dieser Stelle auch noch einen Blick auf die allgemeineren Auswertungen zu werfen. Dort hat sich gezeigt, dass über alle Videos gesehen 5% aller Bewertungen Dislikes sind. Nun ist gut ersichtlich, dass zwischen den Kategorien doch grosse Unterschiede bestehen.

**Unter welcher Kategorie wurde die höchste Anzahl Videos hochgeladen?** Hier hat sich die Kategorie „Entertainment“ an die Spitze gesetzt (vgl. Abbildung 15 auf Seite 29), wobei auf dem zweiten Platz die Kategorie „Music“ ist. Signifikant ist hierbei der rapide Rückgang von der ersten zur zweiten Kategorie. Anschliessend sind die Werte ziemlich gleichmässig abnehmend. Es ist davon auszugehen, dass in Kategorien mit sehr vielen Uploads tendenziell eine grössere Konkurrenzsituation herrscht und es somit schwieriger ist, sich durchzusetzen.

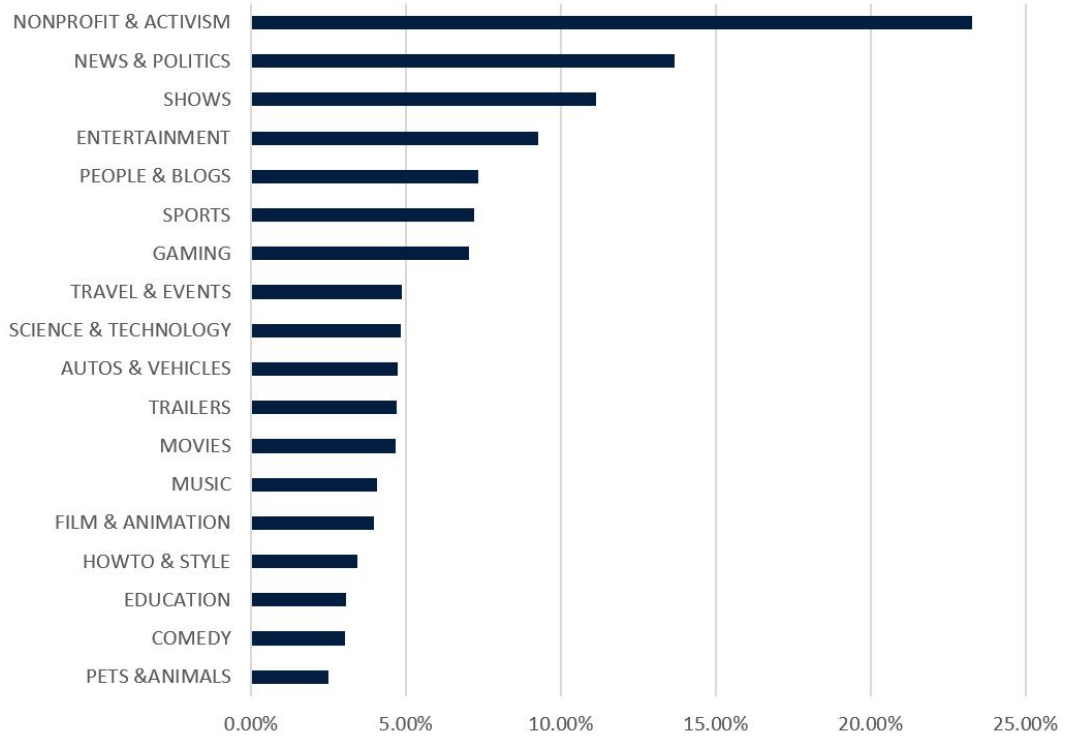



Abbildung 14: Verhältnis Likes zu Dislikes pro Kategorie,
Quelle: Autoren

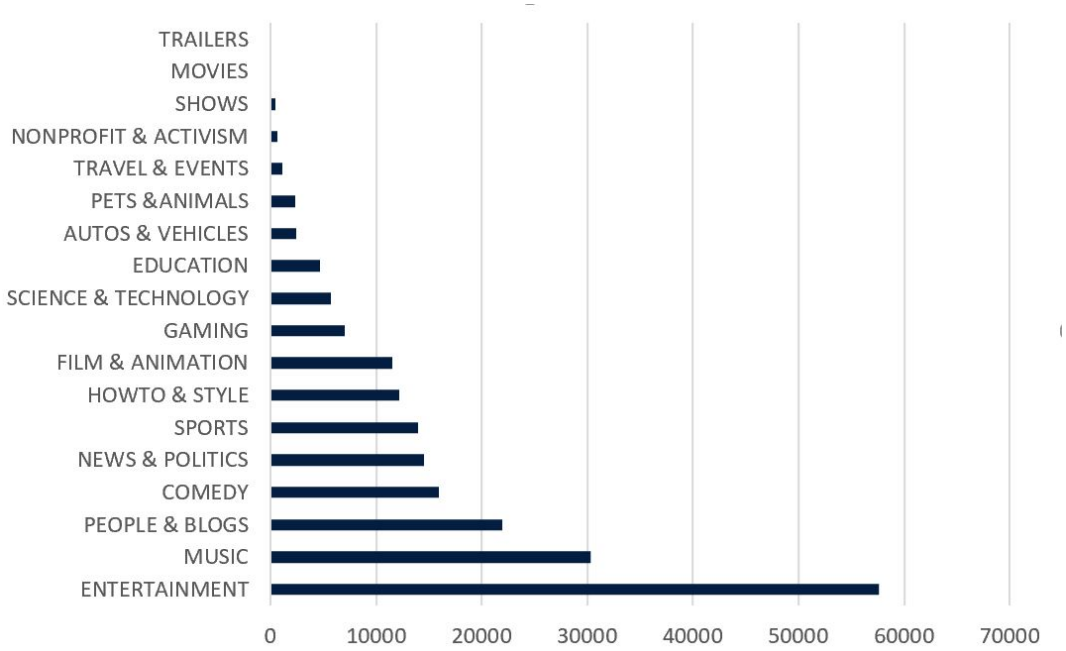



Abbildung 15: Kategorien sortiert nach Uploadmenge,
Quelle: Autoren

## 6.3 Zugriff Webseite

Aus zeitlichen und technischen Gründen hat es für das Deployment der Anwendung des Projekts nicht mehr gereicht. Dennoch soll das Projekt auf der erstellten Webseite für die Bewertung erreichbar sein. Daher wird in den folgenden Schritten erläutert, wie sich die Webseite begutachten lässt:

- Python muss hierfür installiert werden. Dazu müssen folgende Packete importiert werden: flask, flask_wtf, wtforms, json, pymongo
- Klonen des Githubprojekts mit dem entsprechenden Code. Das Repository ist öffentlich und unter folgendem Link abrufbar:
  https://github.com/NicoIseli/Datenbanksysteme
- Verbindungsaufbau mit dem VPN-Service der Hochschule Luzern
- Start der Klasse Main.py. Die Klasse ist im Projekt unter folgendem Pfad zu finden: Datenbanksysteme/02_Code
- Browser öffnen und folgende URL aufrufen: http://localhost:1337

# 7 Diskussion

Grundsätzlich besteht bei Kennzahlen, wie jenen von einem YouTube-Kanal, oft die Gefahr, dass diese vollkommen falsch interpretiert werden, da keine Referenz zur Verfügung steht. Durch das vorliegende Projekt wurde nun genau diese Möglichkeit geschaffen und jeder Kanal kann sich und seine Leistung nun besser beurteilen.

Daher ganz klar die Empfehlung: Man sollte sich besser die Zahlen anschauen anstatt selbst etwas zu interpretieren, denn möglicherweise ist jemand gerade in einer Kategorie aktiv, wo es grundsätzlich sehr wenige Likes gibt und die vermeintlich wenigen Likes normal sind. Falls jemand möglichst viele Views pro Video erreichen möchte, dann würde sich die Kategorie „Music“anbieten, die bei Weitem am meisten Views pro Video erreicht. Dabei gilt aber zu beachten, dass in dieser Kategorie auch sehr viele Videos hochgeladen werden. Hat hingegen jemand anderes Interesse an kontroversen Diskussionen, dann wäre die Kategorie „Nonprofit & Activism“ eher spannend, da dort sehr viele Kommentare pro Video abgegeben werden und zudem ein hohes Verhältnis von Dislikes zu Likes besteht. Eine weitere sehr wichtige Empfehlung ist, dass die Kategorien auch alleine betrachtet werden sollten, da zwischen ihnen teilweise grosse Unterschiede herrschen. Generell lassen sich aus dem Kapitel „Resultate“sehr viele entsprechende Empfehlungen ableiten. Bezüglich den Interaktionen (Likes, Dislikes und Kommentaren) lässt sich festhalten, dass diese eine Rolle spielen, aber nicht einzig für den Erfolg eines Videos verantwortlich sind. Jedoch fallen dabei weitaus am meisten Likes an und daher ist deren Einfluss auch am höchsten.

# 8 Rückblick

## 8.1 Beiträge

Dieses Projekt hat gesamthaft gesehen einen sehr guten Beitrag an den Lernprozess der einzelnen Teammitglieder geleistet. Durch die Anwendung der gelernten Theorie wurden einem diverse Prozesse viel klarer, auch wenn bspw. die Datenbanksicherheit oder die Optimierung der Abfragen lediglich in der Theorie miteinbezogen wurden. Die Highlights sind jedoch die einzelnen Abfragen, welche erarbeitet wurden, die verwendeten Technologien wie Flask und MongoDB und nicht zuletzt das Zusammenspiel zwischen dem Back- und Frontend. Dieses Zusammenspiel war deutlich aufwändiger als gedacht.

## 8.2 Erkentnisse

Da dieses Projekt mit einer NoSQL-Technologie realisert wurde, musste anfangs einiges an Wissen selbst angeeignet werden, da bspw. die Abfragen, Sprachkonzepte wie auch Optimierungen auf SQL aufgebaut waren bzw. die Theorie dazu erst gegen Ende des Semesters vorgestellt wurde. Nichtsdestotrotz wurde ein UseCase erarbeitet, das Datenmanagement konzipiert, ein Datenmodell erstellt, ein Schema definiert, möglichst selektive und umfangreiche Abfragen erarbeitet und schlussendlich die Resultate auch visualisiert, um einfacher entsprechende Erkentnisse daraus zu schliessen. Betreffend der Realisierung war es teilweise schwierig die gelernte SQL-Theorie mit einem NoSQL-Lösungsansatz zu praktizieren. Während das Datenmanagement noch sehr verallgemeinert war, fing es bereits mit der Datenmodellierung an, sich genauer Gedanken darüber zu machen, wie das Ganze realisiert werden soll. Da mit Dokumenten, und nicht mit Tabellen, gearbeitet wurde, musste man dabei etwas umdenken. Die Dokumentendatenbanken wurden dann erst in der letzten Vorlesungswoche vorgestellt, wobei einem dann vieles klarer wurde. Bei der Realisierung der Abfragen musste man sich ebenfalls an das Internet wenden. Es wurden diverse SQL-Abfragen kennengelernt, welche aber mit der MongoDB unterschiedlich realisert werden. Ein interessanter Punkt wurde bei der Abfrage gefunden, bei der man sehr gut erkennen konnte, wo Optimierungsbedarf herrscht, wodurch man sich nachher überlegte, wie man dies verbessern könnte. Des Weiteren war die Datenvisualisierung ebenfalls ein Knackpunkt. Da jede Abfrage die Daten unterschiedlich zurückgibt, musste die Visualisierung für jede einzelne Abfrage individuell implementiert werden. Aus diesem Grund wurde die Idee mit Abfragen über Eingabefelder schnell verworfen, da sich die Visualisierung als komplexer darstellte als zuerst gedacht.

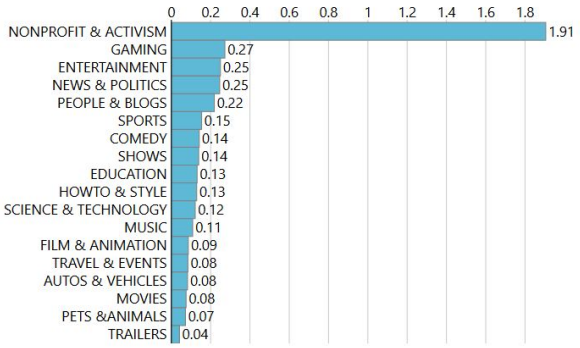
## 8.3 Lessons Learned

Alle Teammitglieder konnten durch das Projekt sehr viel lernen, da beispielsweise zuvor keines der Mitglieder eine vertiefte Erfahrung mit NoSql-Datenabanken hatte. Durch das Projekt wurde nun das Wissen im Bereich von MongoDB aber auch NoSql-Technologien im Allgemeinen aufgebaut, wovon alle in Zukunft profitieren werden. Nebst der reinen Technologie wurde auch das Fachwissen im Bereich des Datenmanagements und Datenmodellierung vertieft. Speziell Teammitglieder, welche bereits die Blockwoche XML/JSON gemacht haben, brachten in diesem Projekt

ihr Wissen über JSON ein und hatten nun die Möglichkeit, dieses im Kontext von der vorliegenden Arbeit weiter zu verbessern. Auf der anderen Seite wurde viel über den entsprechenden Use Case gelernt. Eine der spannendsten Feststellung war, dass sich die Kategorien untereinander sehr stark unterscheiden und es unabdingbar ist, diese für sich separiert anzuschauen, um gute und aussagekräftige Resultate zu erhalten. Wenn nun jemand aus dem Team ein YouTube-Kanal starten möchte, dann stehen durch diese Arbeit alle Informationen bereit, um die Kennzahlen des Kanals besser einzuordnen und damit die richtigen Entscheidungen zu treffen, um erfolgreicher auf YouTube zu sein. Nichtsdestotrotz ist im nachhinein aufgefallen, dass dieser Use Case doch nicht so einfach zu behandeln war, wie anfangs gedacht. Da des Weiteren während der Vorlesung viel auf SQL aufgebaut war, musste man die entsprechenden Aspekte in der NoSQL-Technologie selbst erarbeiten. Deshalb hätte man im Nachhinein eventuell doch mit einer SQL-Lösung das Projekt realisieren sollen, wobei aber nicht jedes Teammitglied diese Meinung teilt. Zudem war die Suche nach einem Use Case als auch nach den entsprechenden Daten nicht besonders einfach, da das in dieser Form noch nie gemacht werden musste. Man stellte sich die Frage, ob man sich zuerst für einen Use Case und folglich einen entsprechenden Datensatz erarbeiten oder einfach nach Datensätzen suchen und darauf einen Use Case erarbeiten soll. Im Nachhinein wäre das letztere wohl der einfachere Weg gewesen. Dennoch ist das ganze Team sehr zufrieden mit dem vorliegenden Projekt und dankbar für die vielen Lessons Learned aus diesem Projekt.

# 9 Anhang

| _id | Dislikes | Views | Ratio |
|---|---|---|---|
| NONPROFIT & ACTIVISM | 16310647 | 855993541 | 1.9054637936806582 |
| GAMING | 17017722 | 6245886771 | 0.2724628643448074 |
| ENTERTAINMENT | 189722509 | 75944025668 | 0.24981887295440283 |
| NEWS & POLITICS | 14369894 | 5838490573 | 0.24612344270029876 |
| PEOPLE & BLOGS | 37976135 | 17374164282 | 0.21857819681919363 |
| SPORTS | 22634186 | 14678848372 | 0.15419592481910815 |
| COMEDY | 23182576 | 16326369939 | 0.1419946754031469 |
| SHOWS | 353969 | 254301867 | 0.1391924503645111 |
| EDUCATION | 3174100 | 2421304968 | 0.1310904674111254 |
| HOWTO & STYLE | 10194442 | 7897214690 | 0.1290890826725137 3 |
| SCIENCE & TECHNOLOGY | 9211627 | 7588688940 | 0.12138627729811784 |
| MUSIC | 255404940 | 235183925321 | 0.10859795781170017 |
| FILM & ANIMATION | 18336961 | 21261626545 | 0.08624439414919655 |
| TRAVEL & EVENTS | 546486 | 649051781 | 0.08419759655508903 |
| AUTOS & VEHICLES | 956940 | 1160818428 | 0.082436665107801 |
| MOVIES | 15922 | 21219708 | 0.07503402026078776 |
| PETS &ANIMALS | 1204952 | 1677002906 | 0.07185151532468484 |
| TRAILERS | 9 | 21744 | 0.041390728476821195 |




Projekt im Modul Datenbanksysteme der Hochschule Luzern - Gruppe 1

Abbildung 16: Beispiel Auswertung,
Quelle: Autoren

# Literatur


Data Pilot. (2019). *Aggregation Pipeline Optimization.* https://kb.objectrocket.com/mongo-db/how-to-create-an-index-for-a-mongodb-collection-in-python-371. (Online; accessed 10-Mai-2020)

Dinesh Prathap. (2015). *Youtube API Video Category Id List.* https://gist.github.com/dgp/1b24bf2961521bd75d6c. (Online; accessed 10-April-2020)

Mitchell J. (2019). *Trending YouTube Video Statistics.* https://www.kaggle.com/datasnaek/youtube-new/data. (Online; accessed 25-March-2020)

Mitchell J. (2020). *Webseite Mitchell Jolly.* https://mitchelljolly.com/. (Online; accessed 30-March-2020)

mongoDB. (2020a). *Aggregation Pipeline Optimization.* https://docs.mongodb.com/manual/core/aggregation-pipeline-optimization/. (Online; accessed 10-Mai-2020)

mongoDB. (2020b). *Install MongoDB Community Edition on Ubuntu.* https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/. (Online; accessed 04-April-2020)

Projektteam. (2020). *Github Repository des Projekts.* https://github.com/NicoIseli/Datenbanksysteme.