Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

PRIVACY RANKING

Wahlprojekt SS 2017

Letztes Update: 22. August 2017

Max Mustermann

Studienbereich Informatik
Hochschule RheinMain

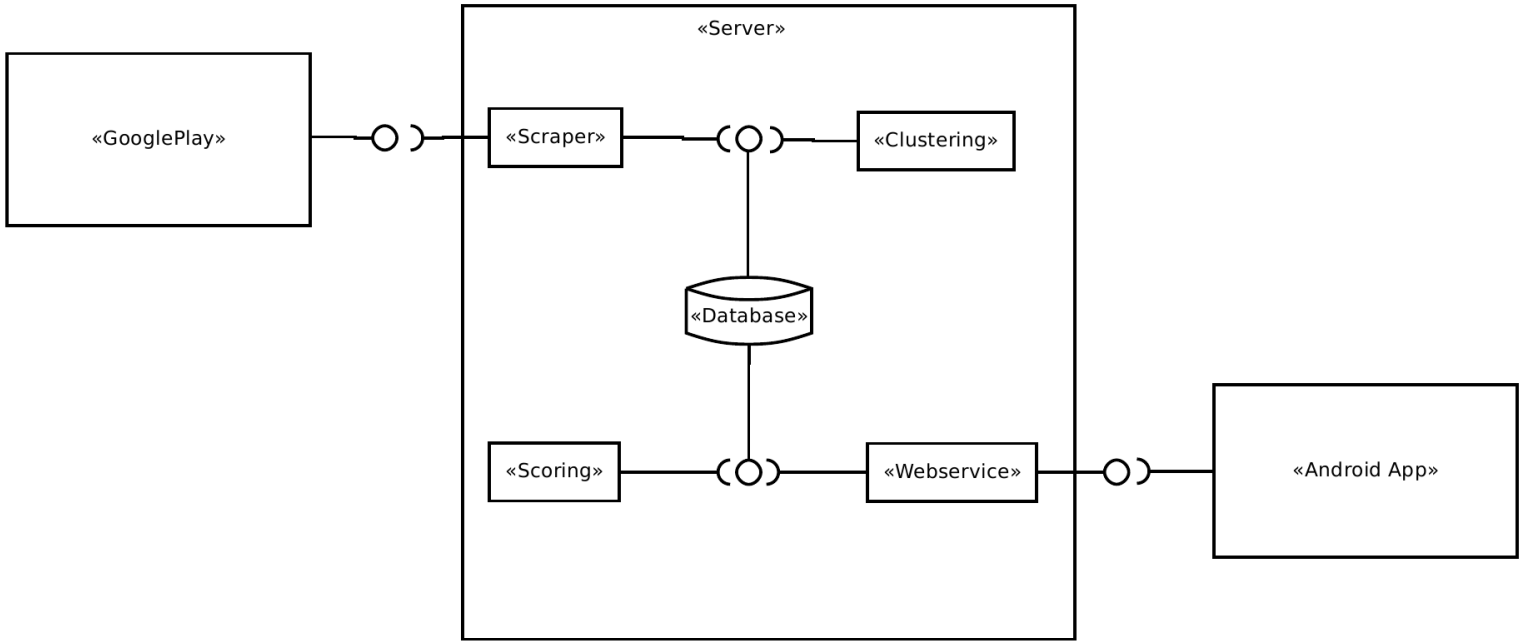
GLIEDERUNG

EINLEITUNG

EINLEITUNG

- Ziel des Projekts?
- Architektur (App - Web - Datenbank)
- Live-Demo der App

ARCHITEKTUR


«Server»
«GooglePlay»
«Scraper»
«Clustering»
«Database»
«Scoring»
«Webservice»
«Android App»

LIVE-DEMO

APP BEISPIELCODE

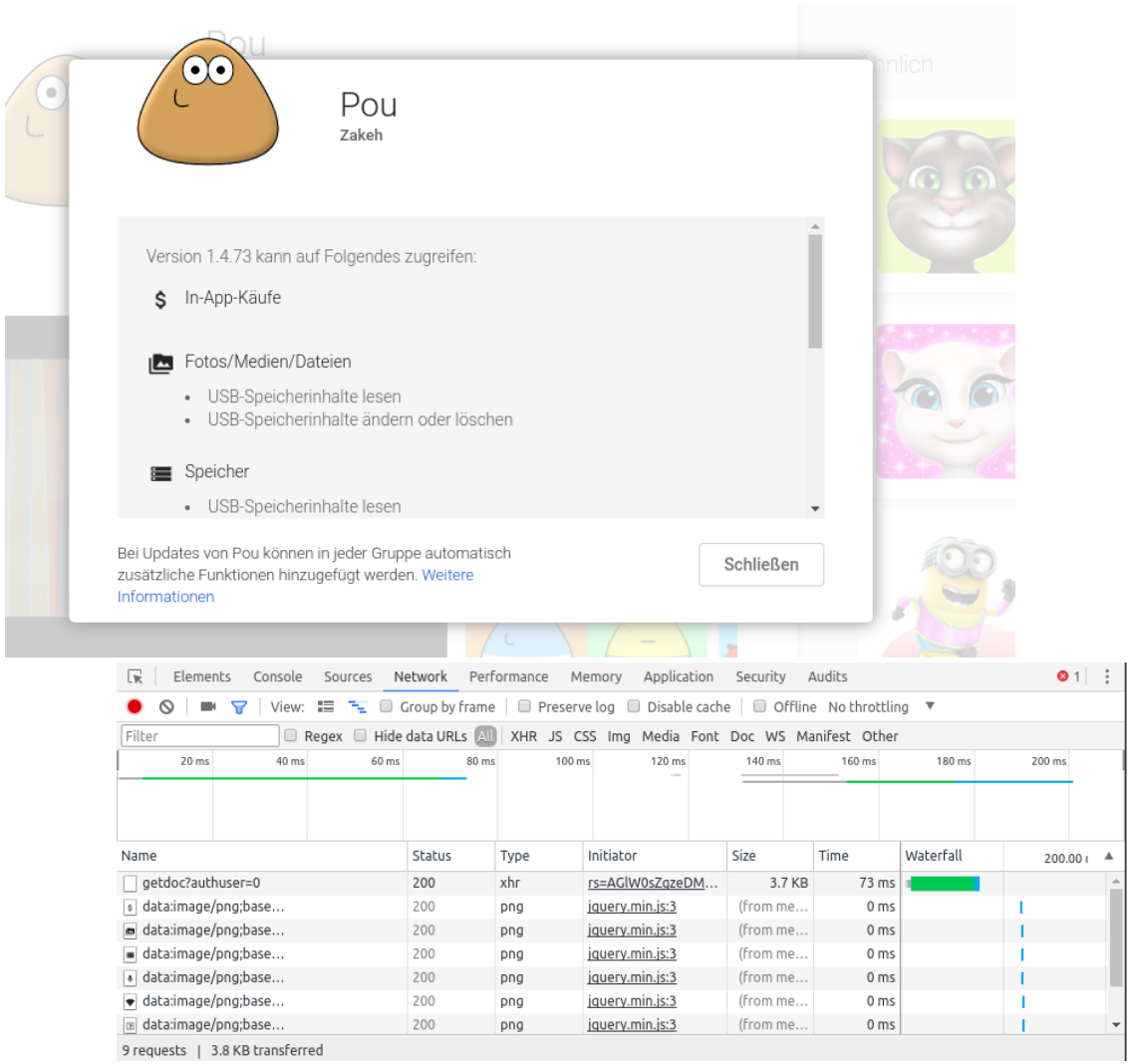
Unsere App halt

WEBSERVICE

Webservice

DATENBESCHAFFUNG UND VERARBEITUNG

WEBSITE GOOGLE PLAYSTORE



SCRAPING DER DATEN

- Zugriff auf den Webservice von Google
- https://play.google.com/store/xhr/getdoc?authuser=0
- POST (ids=app_id, xhr=1)

```
[["gdar",1,[["me.pou.app","me.pou.app",1,3,
"/store/apps/details?id\u003dme.pou.app",
"/store/apps/details?id\u003dme.pou.app",
"https://play.google.com/store/apps/details
?id\u003dme.pou.app","https://market.android
.com/details?id\u003dme.pou.app","Pou",...
```

EXTRAHIEREN DER DATEN

- Schreiben eines Wrappers in Python
- Lokalisieren der nötigen Informationen

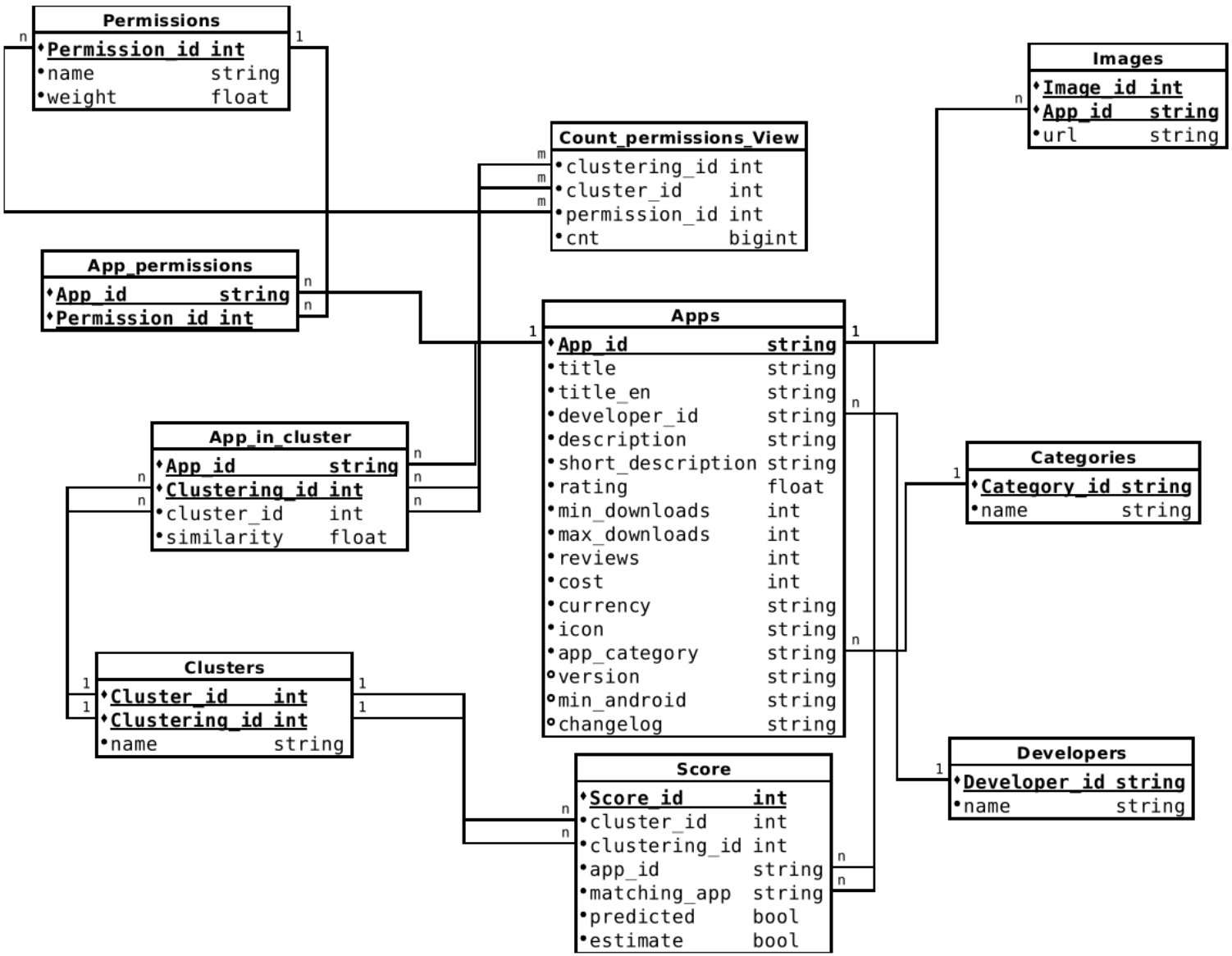
```
def extract_title(data):
    return _remove_emojis(data[0][2][0][8])

def extract_description(data):
    return _remove_emojis(data[0][2][0][9])

def extract_rating(data):
    return data[0][2][0][23]
```

DATENBANK

MARIADB DATENBANK

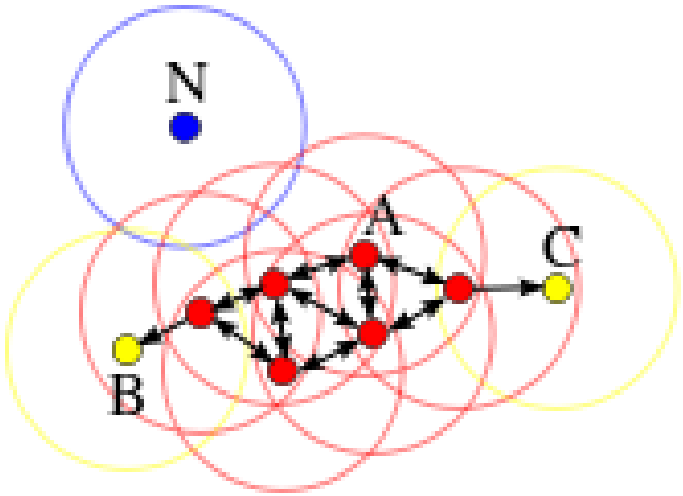


KATEGORISIERUNG UND BEWERTUNG DER APPS

DATAMINING

- Kategorisierung mithilfe von Clustering
- Auswahl zwischen den einzelnen Algorithmen
  - K-Means
    - Anzahl Cluster muss bekannt sein
  - Affinity propagation
    - Terminiert nicht
  - Mean-Shift
    - Terminiert nicht
  - Ward hierarchical clustering
    - Terminiert nicht
  - DBSCAN
    - Rauschen

DBSCAN



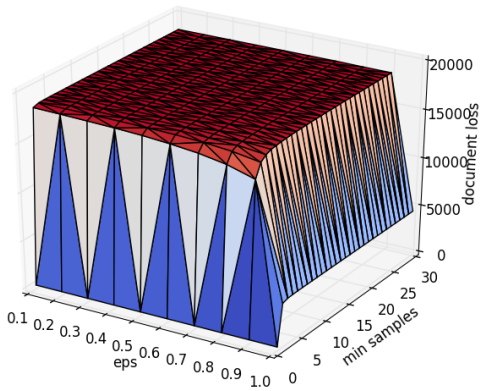
Quelle: Wikipedia

- Density-based spatial clustering of applications with noise
- Abstand (Epsilon) muss gut gewählt werden

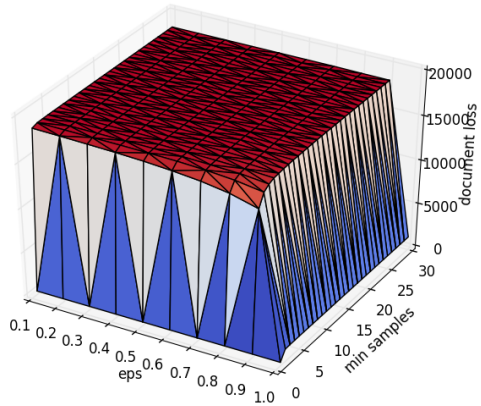
TF-IDF

- Clustering-Algorithmen funktionieren nur mit numerischen Werten
- Text frequenzy
  - Je häufiger Wort in Text enthalten $\Rightarrow$ bedeutend
  - Wert für *min-df* muss gut gewählt werden
- Inversed document frequenzy
  - Je häufiger Wort in allen Dokumenten enthalten $\Rightarrow$ unbedeutend
  - Wert für *max-df* muss gut gewählt werden
- Dadurch entsteht Documents $\times$ Features Matrix
- Max. Feautures werden bestimmt.

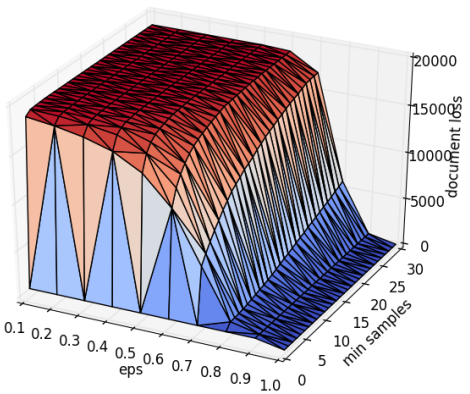
GUTE METRIC FINDEN



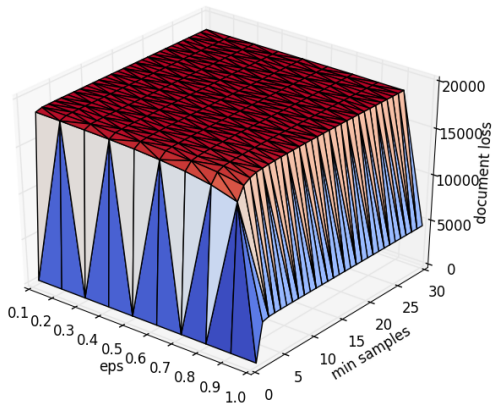
Euclidian



L2

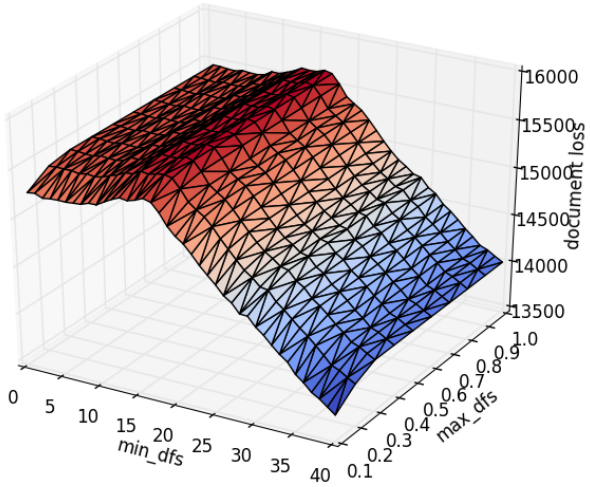


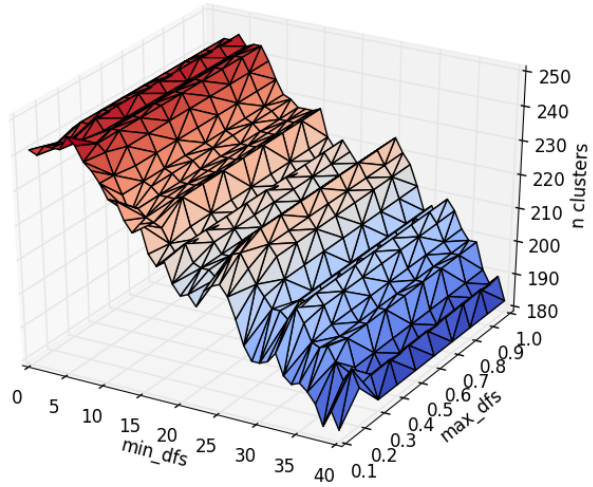
Cosine

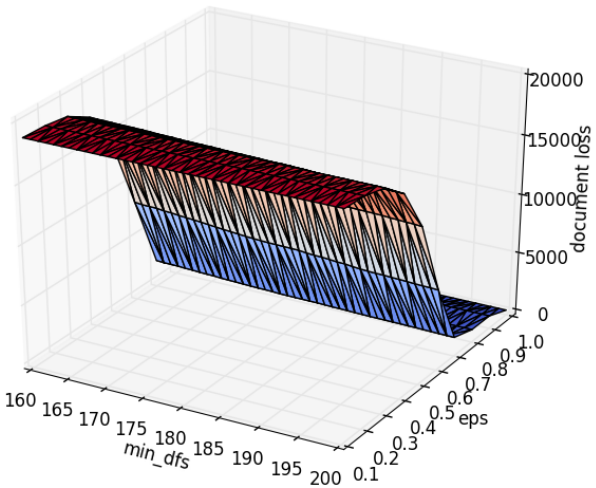


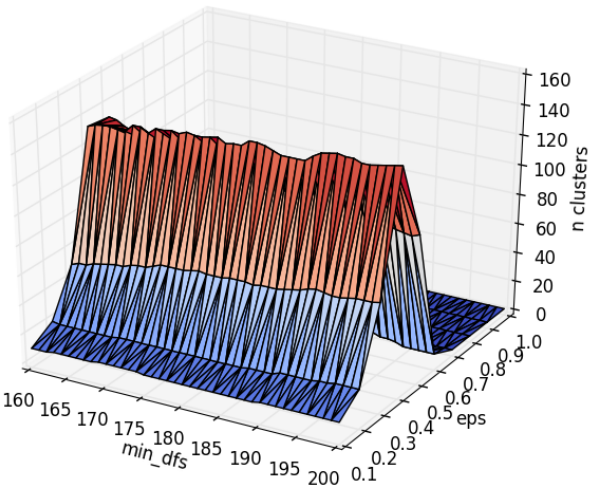
Minkowski

GUTE PARAMETER FINDEN - TESTDATEN

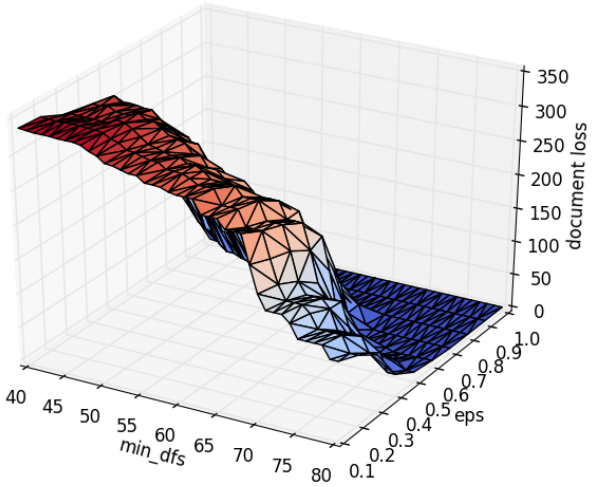


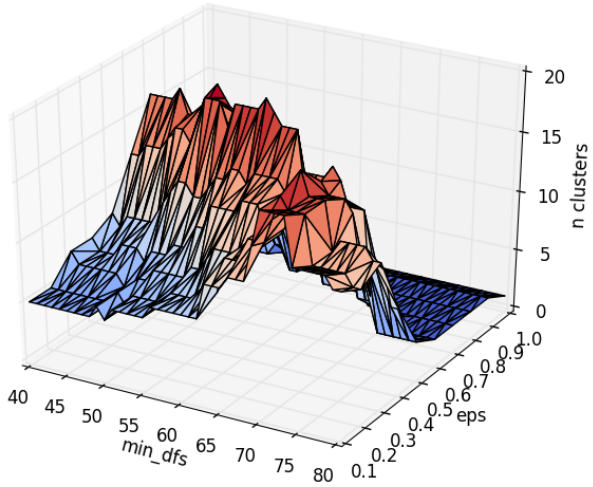


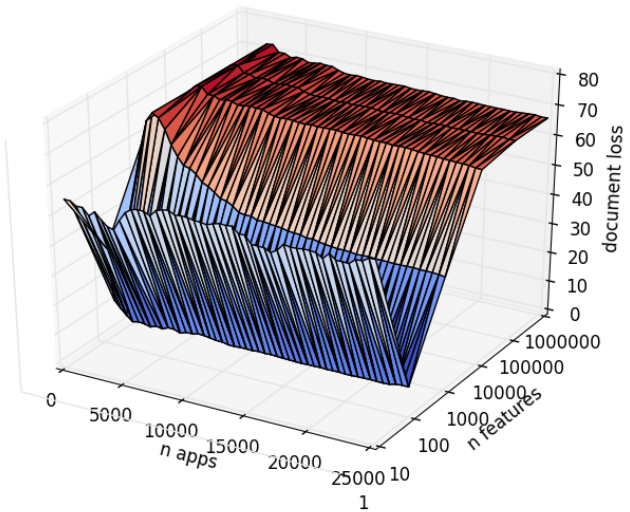


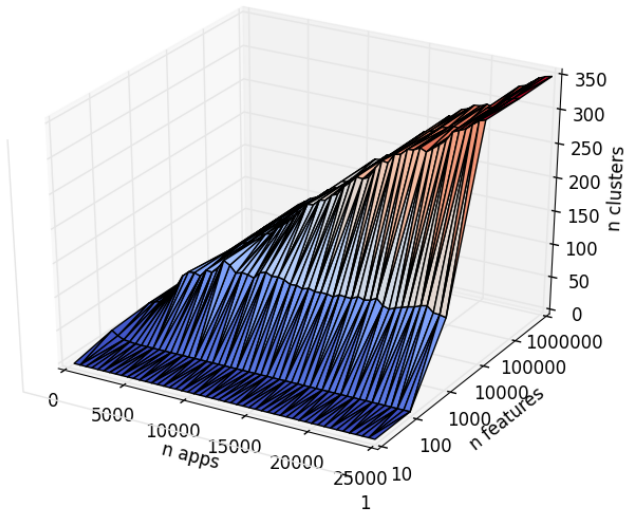


GUTE PARAMETER FINDEN - GOOGLE PLAY DATEN









GUTE PARAMETER FINDEN

- max-df: 0.01
- min-df: 0.005
- eps: 0.45
- min-samples: 30
- features: 1500

$\Rightarrow$ 42 Cluster
$\Rightarrow$ Mehr als 50% Rauschen
$\Rightarrow$ 1 Cluster viel zu groß

KOMBINATION MIT ANDEREN ALGORITHMEN

- K-Means
  - Anzahl Cluster aus DBSCAN → mäßiger Erfolg
  - Anzahl GP Kategorien → mäßiger Erfolg
- Classifier
  - DecisionTree → miserabler Erfolg
  - BernoulliNB → miserabler Erfolg
  - MLP → miserabler Erfolg
  - AdaBoost → miserabler Erfolg
  - KNeighbors → akzeptabler Erfolg

⇒ Kein Verlust mehr durch Rauschen
⇒ Zu großer Cluster wurde noch größer
⇒ Cluster beinhaltet mehr als 50% apps

HIERARCHICAL DBSCAN

Aufteilung von zu großen Cluster in kleinere.

⇒ Sprengt den Arbeitsspeicher.

> *Dies liegt an der mieserablen Implementierung in SKLearn. Es ist besser, wenn du's selbst implementierst.*
> *- Viele Leute bei Stackoverflow*

Eigene Variante in Kombination mit KNeighbors:

- Zu große Cluster werden erneut mit DBSCAN geclustert (kleineres Epsilon)
- Dabei entstandendes Rauschen wird mithilfe KNeighbors neu verteilt

⇒ Clusterqualität wurde schlechter, kein guter Erfolg

BEWERTUNG DER APPS

Die Apps werden nach dem Einfluss auf die Privatsphäre bewertet.

1. Sammeln der Berechtigungen innerhalb eines Clusters

Permissions

0	**4**	**9**	**10**	**11**	**12**

Mit den Berechtigungen:

ID	**Name**
0	In-App-Purchases
4	Calender
9	Pictures/Media/Files
10	Storage
11	Camera
12	Microphone

BEWERTUNG DER APPS

2. Berechnung der Gewichtung

Besteht aus zwei Teilen:

- Relative häufigkeit von App die diese Berechtigung **nicht** haben

0.4	0.8	0.6	0.2	0.0	0.6

- Bösheit der Berechtigung

0.1	0.6	0.1	0.1	0.9	0.9

BEWERTUNG DER APPS

Diese werden miteinander multipliziert.

Permissions

0	**4**	**9**	**10**	**11**	**12**
0.04	0.48	0.06	0.02	0.0	0.54

3. Füllen der Matrix

Permissions

Apps ID	**0**	**4**	**9**	**10**	**11**	**12**
14	0.04	0.0	0.0	0.02	0.0	0.54
42	0.0	0.48	0.06	0.0	0.0	0.0
145	0.04	0.0	0.0	0.02	0.0	0.0
465	0.04	0.0	0.06	0.02	0.0	0.54
1010	0.0	0.0	0.0	0.02	0.0	0.0

BEWERTUNG DER APPS

4. Aufsummieren der Werte

Permissions

Apps ID	**0**	**4**	**9**	**10**	**11**	**12**	$\sum$
14	0.04	0.0	0.0	0.02	0.0	0.54	**0.6**
42	0.0	0.48	0.06	0.0	0.0	0.0	**0.54**
145	0.04	0.0	0.0	0.02	0.0	0.0	**0.06**
465	0.04	0.0	0.06	0.02	0.0	0.54	**0.66**
1010	0.0	0.0	0.0	0.02	0.0	0.0	**0.02**

BEWERTUNG DER APPS

5. Aufteilen in 3 Gruppen mithilfe K-Means

Apps

ID	$\sum$
14	0.6
42	0.54
145	0.06
465	0.66
1010	0.02

- Gut - Grün
  - 80 - 120 degree
- Mittel - Gelb
  - 30 - 79 degree
- Schlecht - Rot
  - 0 - 29 degree

```
# 0 - 100
value = 100 - ((app_values[i] - min_value) *
    100.0) / (max_value - min_value)
# min_range - max_range
value = (value * (color_range[1] -
    color_range[0]) / 100) + color_range[0]
```