

Prof. Dr. Agnès Voisard, Nicolas Lehmann

# Datenbank Systeme, SoeSe 2017

## Project 3. Iteration

TutorIn: Hoffman Christian

Tutorium 3, Gruppe 22

Ingrid Tchilibou, Emil Milanov, Boyan Hristov

9. Juli 2017

---

## Allgemein

Link zum Projekt: [https://github.com/gancia-kiss/dbs\\_projekt](https://github.com/gancia-kiss/dbs_projekt)

## 1. Aufgabe: Data Mining: Clustering

### 1. Ähnlichkeitsmetrik

Um die Gemeinsamkeit je 2 Hashtags zu berechnen, haben wir zwei Metriken genommen. Erstens ist es für uns wichtig, wie oft zwei Hashtags gemeinsam zusammen vorkommen und zweitens, wie oft diese in der selben Woche benutzt werden. So können wir beide Hashtags mit äquivalente Bedeutungen als auch zu einander offensive Hashtags zusammenfassen. Wenn es ein Trend in einer Woche gibt, dann werden alle zu diesem Trend relevante Hashtags in der selben Woche benutzt, aber nicht unbedingt zusammen, da oft nur ein Paar Hashtags per Tweet benutzt werden, und nicht alle, die zu dem Thema relevant sind. Wenn zwei Hashtags aber doch nur zusammen vorkommen, muss das als eine stärkere Bindung markiert werden, deswegen reicht das Datum alleine nicht.

### 2. Clustering

Wir haben jede Woche und jeden Hashtag aus dem Datenbank eine eigene Dimension zugewiesen, damit wir diese in einem Vektorraum representieren können. Falls zum Beispiel ein Hashtag nur in der ersten und zweiten Woche verwendet wurde und zwar 2 mal in der ersten und 4 mal in der zweiten, dann sieht sein Vektor folgendermaßen aus

$$[2, 4, 0, 0, 0, \dots]$$

Wenn man dann die Dimensionen für die Zusammenverwendung von Hashtags hinzufügt, und zum Beispiel unser Beispielshashtag der lexikographisch erste ist und nur zusammen mit dem 4. Hashtag 3 mal vorgekommen ist, dann sieht sein Vektor folgendermaßen aus

$$[2, 4, 0, 0, \dots, 3, 0, 0, 3, 0, 0, \dots, 0]$$

So werden Hashtags, die in der selben Wochen benutzt werden, in der selben Dimensionen zusammen wachsen und folglich ein kürzeres Euklidischen Abstand haben. Das selbe gilt auch für die Zusammenverwendung im selben Tweet.

### 3. Curse of Dimensionality

Wir könnten aber leider ohne weiteres K-Means nicht erfolgreich auf der so konstruierten Vektorraum verwenden. Das liegt daran, dass unsere Daten in einer so großen Anzahl von Dimensionen sehr Sparse werden und dabei den euklidischen Abstand zwischen den einzigen Elemente stark abnimmt, so dass wir am Ende ein riesengroßen Cluster bekommen, wo fast alle Hashtags gesammelt werden, da diese nicht mehr so deutlich von einander unterscheidbar sind.

Als in der Bücher erklärt, wächst zusammen mit der Anzahl von Dimensionen auch der Anteil von Kantenlängen eines Hyperwürfels, um bestimmten Anteil des Hyperwürfels abzudecken. Damit nimmt die Ähnlichkeit im Sinne von euklidischen Abstand zwischen der einzelnen Punkte mit der Anzahl von Dimensionen drastisch ab. Deswegen werden im unserem Fall mehrere Punkte im selben Cluster gesammelt, obwohl die eigentlich nicht so nah sein dürfen.

Quellen:

Machine Learning: A Probabilistic Perspective, K.P. Murphy, 2012, S.18

Deep Learning, I. Goodfellow, Y. Bengio, A. Courville, 2016, S. 151

[https://en.wikipedia.org/wiki/Curse\\_of\\_dimensionality](https://en.wikipedia.org/wiki/Curse_of_dimensionality)

4. Lösung Wir haben Dimensionality Reduction verwandt, um die Punkte wieder näher zu bringen, damit wir erfolgreich K-Means anwenden können. Wir haben uns für den t-SNE ('t-distributed stochastic neighbor embedding') Algorithmus entschieden, der ein nicht-lineares Machine-Learning Algorithmus ist und die Ähnlichkeit der einzelnen Punkte bei der Reduktion versucht zu behalten.

Da aber zwei Dimensionen nicht ausreichend sind, um unseren Daten richtig und verlustlos zu representieren, haben wir uns entschieden, erstmal nur auf 10 Dimensionen den Vektorraum zu reduzieren, K-Means anzuwenden und erst danach weiter auf 2 Dimensionen zu reduzieren für die Representierung. Dabei ist nicht mehr wichtig, dass alle Abstände richtig sind, da wie die Kantenbreite benutzen, um Ähnlichkeit zwischen je zwei Hashtags zu äußern.

Weiter haben wir für die Implementierung die Bibliothek Numpy benutzt, um leichter und deutlich effizienter Euklidische Abstände berechnen zu können, was wichtig für den K-Means Algorithmus ist. Wir haben dann alle Clusters, Kanten und Koordinaten von den Hashtags in beiden 10-Dimensionalen und 2-Dimensionalen Vektorraum in dem Datenbank gespeichert, damit wir leichter das Ergebniss analysieren können, bevor wir die Front-end Seite fertig gemacht haben.

## 2. Aufgabe: Datenvisualisierung

### 1. Visualisierung von dem Clustering

Für den 'Maximum bang for the buck' haben wir für die Front-End Seite AngularJS zusammen mit Bootstrap genommen, da beide minimale Konfiguration erfordern und gute Leistung anbieten. AngularJS hat uns geholfen eine modernere Single-Page-Application entwickeln zu können und leichter Daten aus unserer API auf der HTML Seiten visualisieren zu können, mit Hilfe von Templates und 2-way-data-binding. Die Architektur ist natürlich für größere Projekten nicht so effizient, für unsere kleine Anwendung finden wir aber die Entscheidung ganz gut. Bootstrap ist ein CSS Framework, dass uns erlaubt hat, fast kein CSS zu schreiben und trotzdem schönes Endergebniss zu bekommen, damit sehen auch alle unsere UI Elemente einheitlich aus.

Wir haben gesehen, dass die Bibliothek 'SigmaJS' leicht mit JSON Dateien arbeitet, deswegen hatten wir keine Schwierigkeiten die Daten, die wir schon in dem Datenbank gespeichert haben, über Ajax Anfragen zu den Klient zu transportieren.

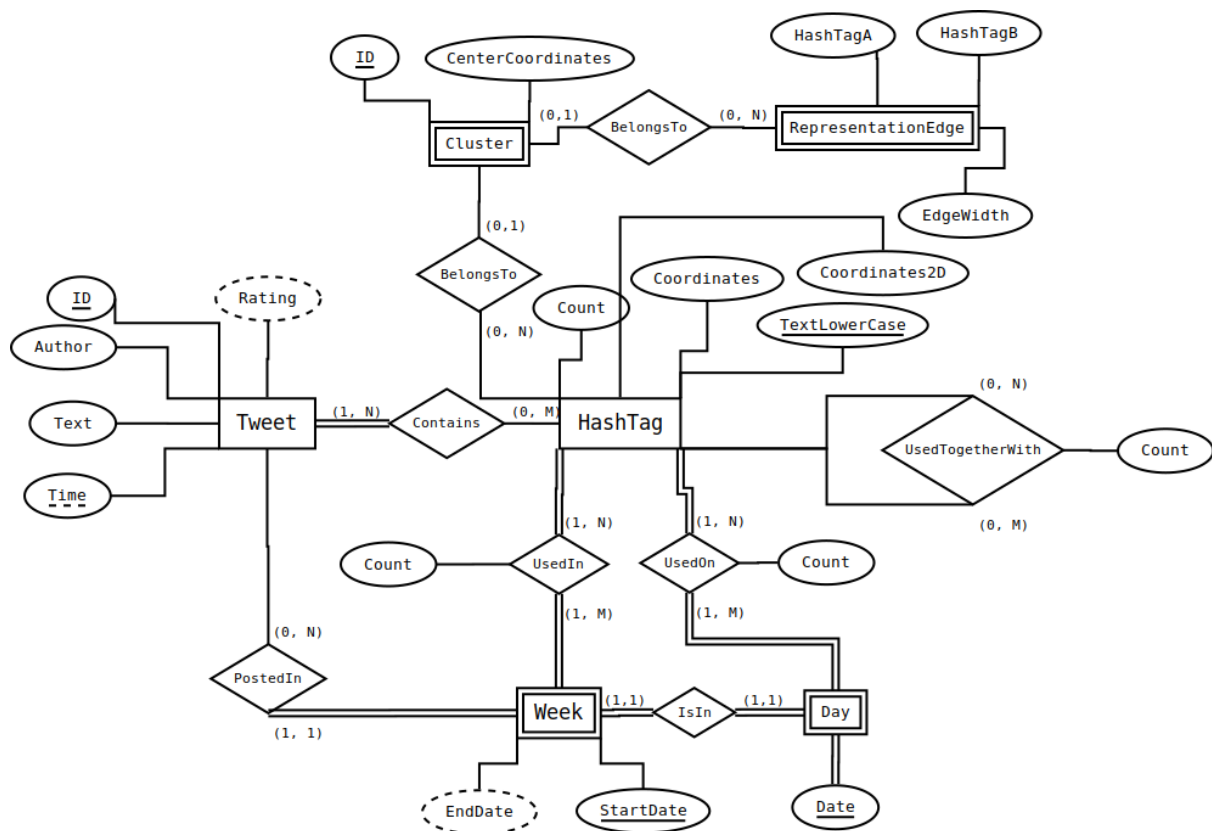
Für die Visualisierung haben wir uns entschieden, stärkere Beziehungen zwischen Hashtags mit dickeren Kanten darzustellen. Dabei wurde schon ein Constraint in der Aufgabenstellung gegeben, und nämlich das zwei Hashtags, die nie mit einander zusammen benutzt wurden, nicht mit einer Kante verbunden sein dürfen. Wir haben ein weiteres Constraint eingefügt, dass das auch für zwei Hashtags gilt, die sich nicht in dem selben Cluster befinden. Die verschiedene Farben bezeichnen bei uns auch die verschiedene 7 Cluster.

## 2. Visualisierung von der Häufigkeit

Für die Visualisierung der Häufigkeit haben wir die Bibliothek Matplotlib benutzt. Wir haben schon in der letzten Projektiteration eine Tabelle eingeführt, wo gespeichert wird, in welchen Wochen ein Hashtag benutzt wurde und wie oft. Anhand von der existierenden Implementierung haben wir das selbe auch für die einzelne Tage gemacht und dann anhand davon die Visualisierung gemacht. Da wir auch eine Tabelle für die Wochen haben, haben wir den Nutzer die Möglichkeit gegeben, die gewünschte Einheit zu wählen.

## Anpassung des ER-Diagramms und damit auch des Datenbanks

Wir haben noch alle für die Clusters und ihre Visualisierung benötigte Daten in dem Datenbank gespeichert, damit wir die Antwortzeit (Response-Time) des Servers minimieren. Das ER-Diagramm sieht jetzt folgendermaßen aus



## Schwierigkeiten

Die größte Schwierigkeiten, die wir in dem Projekt hatten, waren bei der Clustering

### 1. Ersten Ansatz, Idee nicht realisiert

Wir haben am Anfang gedacht, dass wir das absolut minimalste Abstand zwischen alle Daten, in den zwei Hashtags benutzt wurden, nehmen können und dazu gegen proportional die Anzahl der Zusammenverwendungen nehmen können und dann die zwei Werte einfach skalieren und aufsummieren können. Die Idee ist für Abstandsberechnung ganz gut, damit kann man aber die Hashtags nicht in einem Vektorraum ordnen und dafür kann man auch K-Means in der Situation nicht anwenden.

### 2. Zweiten Ansatz

Wir haben dann die Hashtags richtig in einem n-dimensionalen Vektorraum geordnet, dabei aber keine Dimensionsreduktion gemacht. Dabei haben wir über 'Curse of dimensionality' gelernt. Die Clusters, die dabei entstanden sind, waren nur für die Hashtags gut, die ganz oft verwendet wurden, die nicht so berühmte aber wurden alle zu dem selben Cluster zugeordnet.

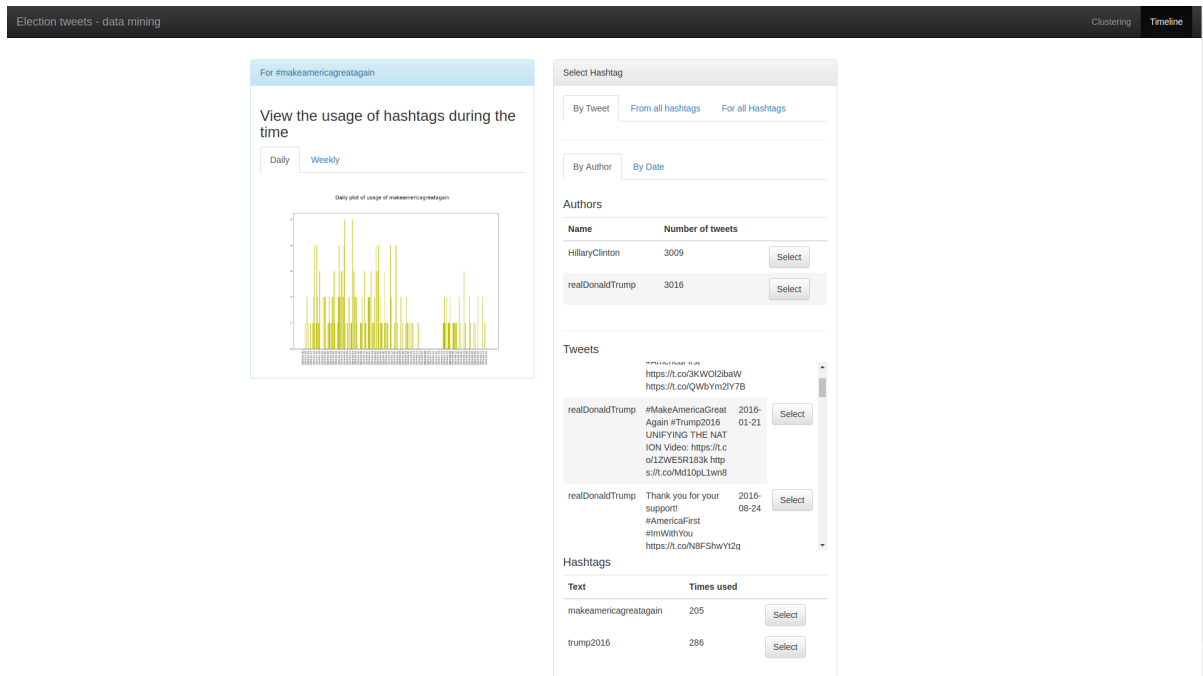
### 3. Dritten Ansatz

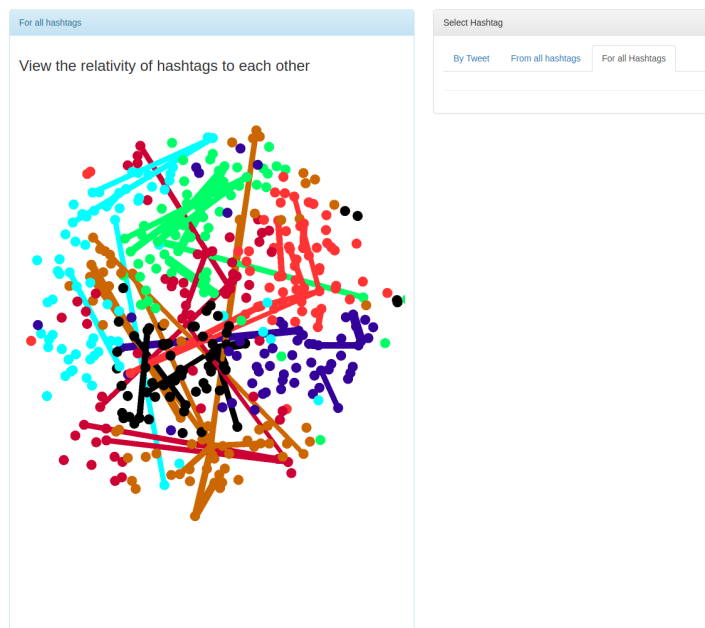
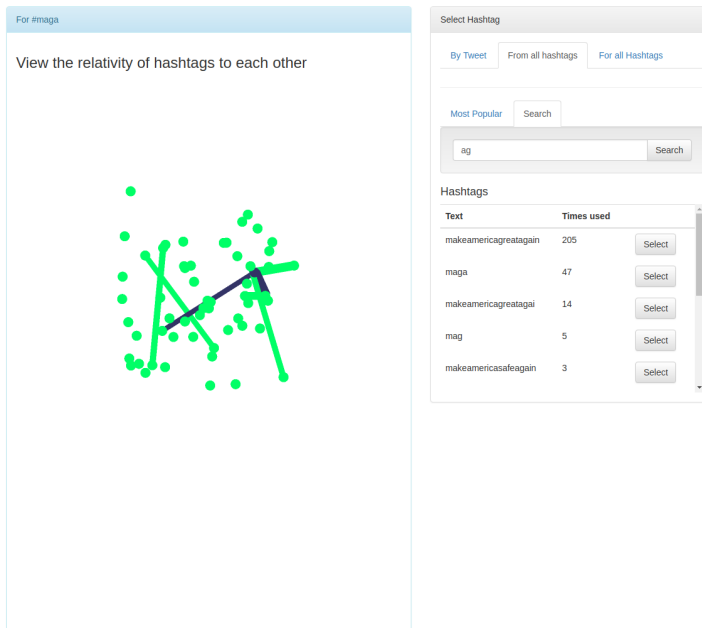
Wir haben Dimensionsreduktion angewandt und einigermaßen gute Clusters bekommen

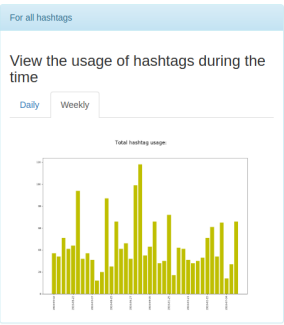
### 4. Vierten Ansatz

Wir haben weiter gelesen, wie wir den t-SNE Algorithmus besser konfigurieren können, wie wir die zwei Features gut skalieren können und wie viele Clusters wir erstellen sollen. Die richtige Werte für die meisten dieser 'Tweaks' haben wir experimentell gefunden. Für die Skalierung des Gewichts der einzige Features haben wir überlegt, dass eine Kombination von 2 Hashtags in einem Tweet 2 mal in usedTogetherWith markiert wird und nur einmal in dem Datum, weiter gibt es auch deutlich mehrere Hashtags (damit auch Dimensionen dafür) als Wochen. Deswegen haben wir die Dimensionen für die Wochen mit 3 skaliert.

## Screenshots







Select Hashtag

[By Tweet](#) [From all hashtags](#) [For all Hashtags](#)

[By Author](#) [By Date](#)

Authors

Name	Number of tweets	
HillaryClinton	3009	<a href="#">Select</a>
realDonaldTrump	3016	<a href="#">Select</a>