

# Lab 01 Unsupervised Learning ‘Song Clustering’

Bogumila Dubel | Johantha Hänni | Yannic Wetzel

## TF IDF:

### Einführung:

Das Vektorisieren mit TF IDF ist ein wichtiger Schritt im Prozess des Text - Clusterings. Der TF IDF - Vektors gilt nachher als Grundlage für die Durchführung der Clustering-Prozesse. In diesem Lab wird man sich mit den folgenden Verfahren: ‘PCA’, ‘k-Means’ und ‘Hierarchical Clustering’ auseinandersetzen.

### Tool:

- TfidfVectorizer von scikit-learn

### Beobachtungen:

- Vielzahl an Konstruktor-Parametern, mit denen der TF IDF Vektor beeinflusst werden kann
- Die folgenden Konstruktor-Parameter haben einen deutlichen Einfluss auch bei gut vorbereiteten Daten, die vom Aussreisse bereinigt sind:
  - stop\_words = ‘english’ => es wurden nur die englischen stopwords berücksichtigt, die von der NLTK zur Verfügung gestellt wurden. Teilweise gibt es in den Daten spanische oder französische Texte. Die stopwords für diese Sprachen wurden nicht berücksichtigt.
  - sublinear\_tf = True => beim Logarithmieren bekommt man deutlich bessere Resultate im Dendrogramm, s. Annang ‘ilogarithmisch’ vs ‘linear’,
  - norm => Default is L2. Wenn man jedoch anstatt default die L1 Norm verwendet hat dies einen deutlichen Einfluss auf das Dendrogramm s. Anhang ‘norm\_L1’
- Die folgenden Konstruktor-Parameter haben einen grossen Einfluss, vor allem wenn die Daten einige Ausseisser beinhalten:
  - max\_df = 0.1132 => es hat das beste Modell in dendrogram geliefert, nachdem die erste Aussreisser-Gruppe entfernt wurde (siehe Codezeile: 128-134). Bei jeder kleinen Value-Anpassung z.b von 0.1132 auf 0.15 oder 0.1 kann man einen grossen Einfluss auf das Daten Clustering beobachten (s. ‘max\_df\_01\_without\_1\_outlier\_group’, ‘max\_df\_01132\_without\_1\_outlier\_group’, ‘max\_df\_015\_without\_1\_outlier\_group’).
  - min\_def => dito max\_df

### Resultate:

Text-Document (s. github.zhaw): ‘words\_per\_artist.txt’

## PCA:

### Einführung:

Das Plotten von Principal Component Analysis gibt vor allem einen guten Überblick darüber: Texte von welchen Künstlern Ähnlichkeiten aufweisen. Diese befinden sich nebeneinander in dem gleichen Axen-Viertel. Die Künstler, die sich nah am Achsenursprung befinden, liegen im Mittelfeld und heben sich nicht stark von den anderen Künstlern ab. Hingegen liegen die Künstler weit vom Achsenursprung entfernt, deutet es auf Sondereigenschaften

und dementsprechend auf Ausreisser hin.

#### Tools:

- PCA und StandardScaler von scikit-learn

#### Beobachtungen:

- Das Plotten von PCA ohne vorherige Standardisierung verleiht schon den ersten Eindruck, welche Künstler sich ähneln. Das Erkennen der Aussreisser ist jedoch schwierig, fast unmöglich, s. 'pca\_without\_standard\_scaler'
- Das verwenden des StandardScalers als Preprocessing für den TF IDF Vector hilft dabei die Aussreisser zu erkennen. Es hat aber keinen Einfluss auf die Clustering-Resultate, deshalb wegen besserer Leserlichkeit hat man auf die StandardScaler beim Generieren von pca.png verzichtet. Der StandardScaler wurde nur dazu verwendet die zu entfernenden Aussreisser zu identifizieren.

#### Resultate:

PNG (s. github.zhaw): 'pca.png'

## k-Means:

#### Einführung:

K-Means Clustering wurde durchgeführt um die Text nach den Genres zu clustern. Da in den Daten 10 verschiedene Cluster vorkommen, wurde der Konstruktor - Parameter n\_clusters = 10 gesetzt.

#### Tools:

- KMeans von scikit-learn

#### Beobachtungen:

- init - Parameter hat einen grossen Einfluss darauf, wie die Daten geclustert werden. Es wurde der Value = 'k-means++' gegenüber 'random' vorgezogen
- Bei jedem Durchlauf werden andere Resultate in der Textdatei 'words\_genres.txt' generiert. Das Setzen des Parameters max\_iteration = 5000 hat dabei geholfen, dass die Resultate von den unterschiedlichen Durchläufen sich einander mehr ähneln.
- Es treten noch sehr viele stopwords auf, welche in NLTK nicht berücksichtigt wurden.
- Die nicht englische Texte werden teilweise zu einem Cluster gebildet, wie z.B Französisch. Dies zeigt, dass es auf unterschiedliche Wörter korrekt geclustert werden. Es verfälscht jedoch das Genre - Clustering.

#### Resultate:

Text-Document (s. github.zhaw): 'words\_genres.txt'

## Hierarchical clustering:

#### Einführung:

Das hierarchische Clustering generiert mit dendrogram eine Cluster - Repräsentation, die vom Menschen am besten zu lesen und intuitiv zu verstehen ist.

#### Tools:

- cluster.hierarchy von SciPy

#### Beobachtungen:

- SciPy bietet eine Vielfalt an Optionen an, anhand von welchen geclustert werden kann wie: linkage(y), single(y), complete(y), average(y) => durchs Ausprobieren und Plotten von verschiedenen Modellen konnte man feststellen, dass für diese Aufgabenstellung linkage die am besten geeignete Methode ist.
- Wie schon am Anfang erwähnt TF IDF Vector hatte einen grossen Einfluss auf das

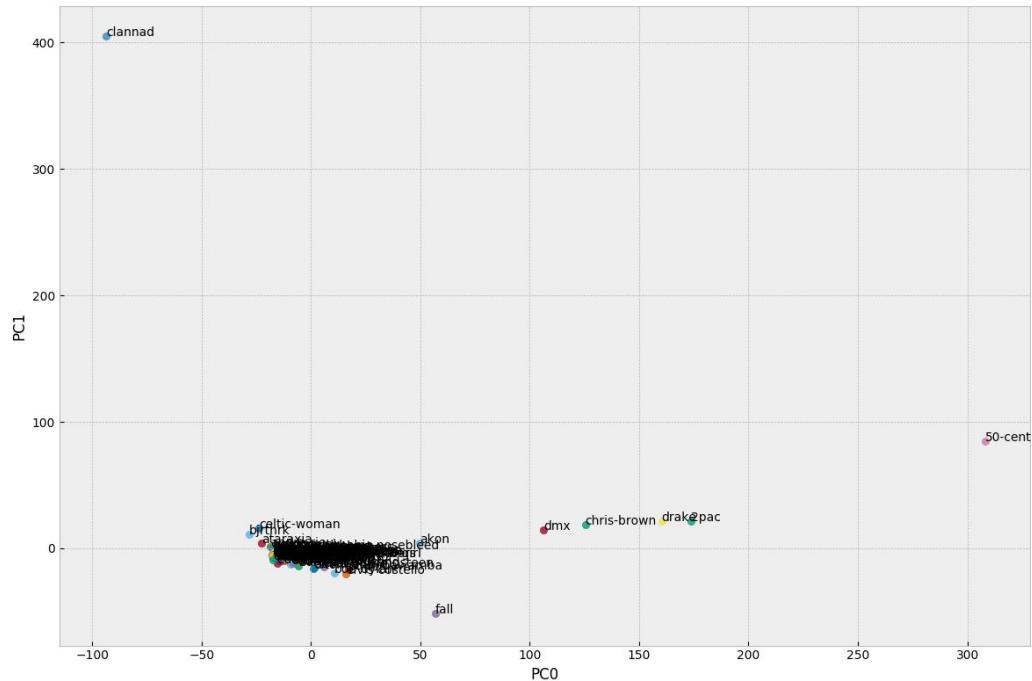
Dendrogram, s. Resultate im Anhang und Kommentare zu TF IDF

**Resultate:**

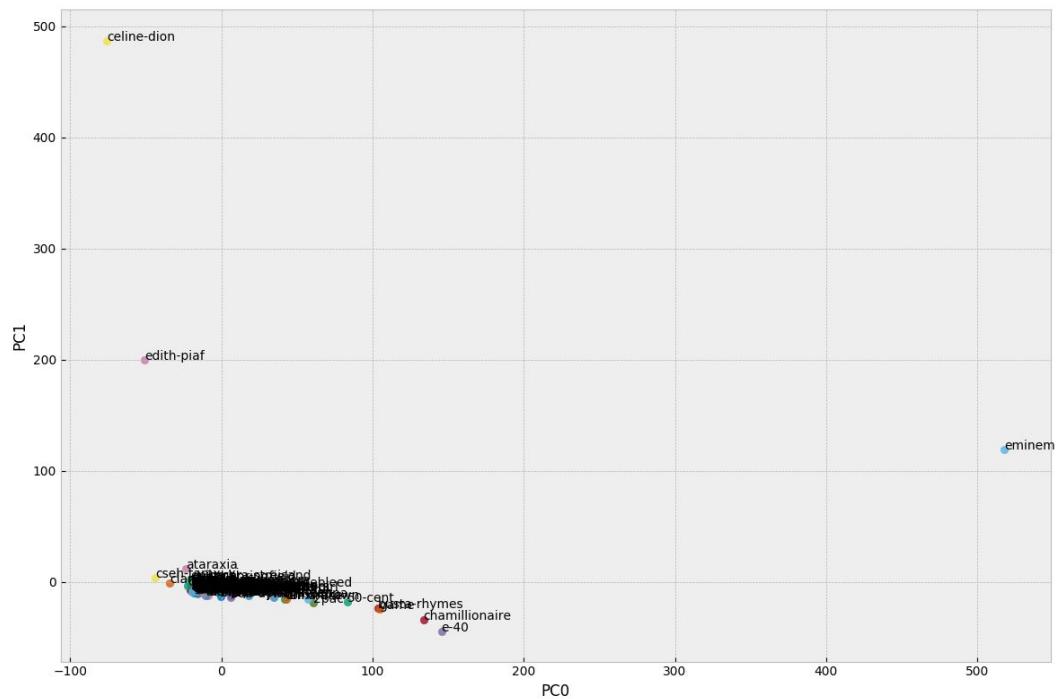
SVG (s. github.zhaw): dendrogram.svg'

## Anhang:

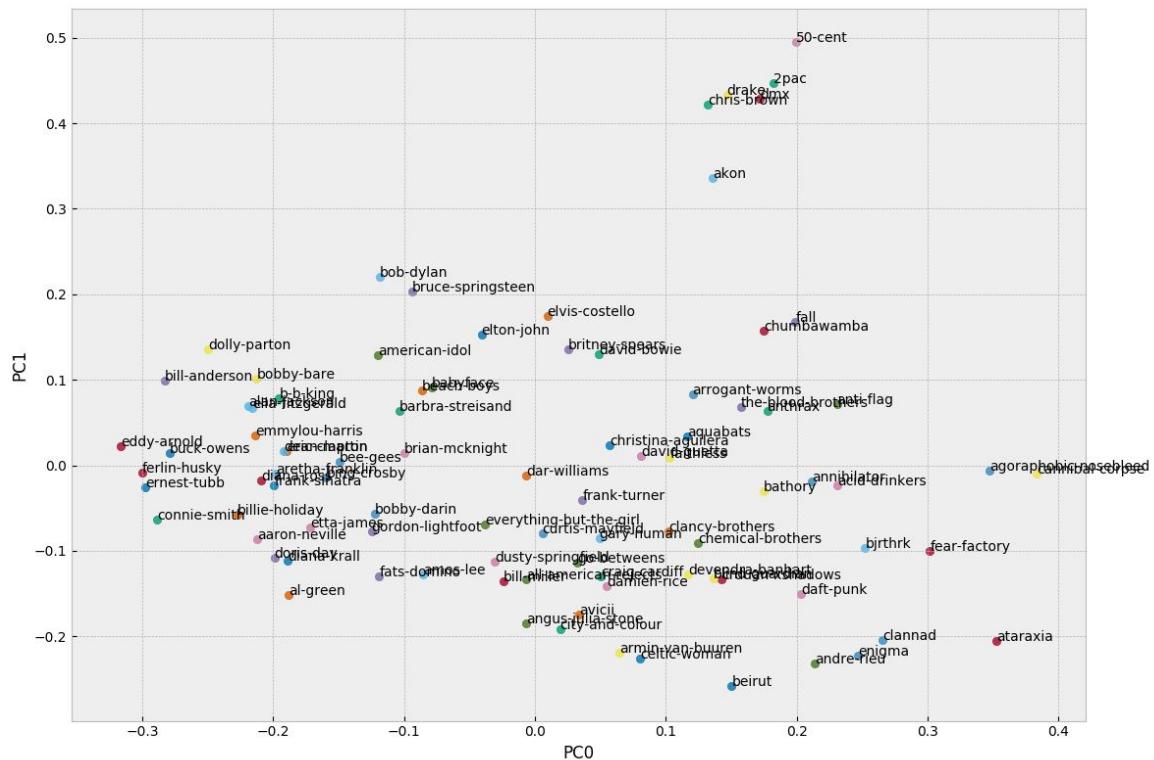
### PCA with StandardScaler without outliers



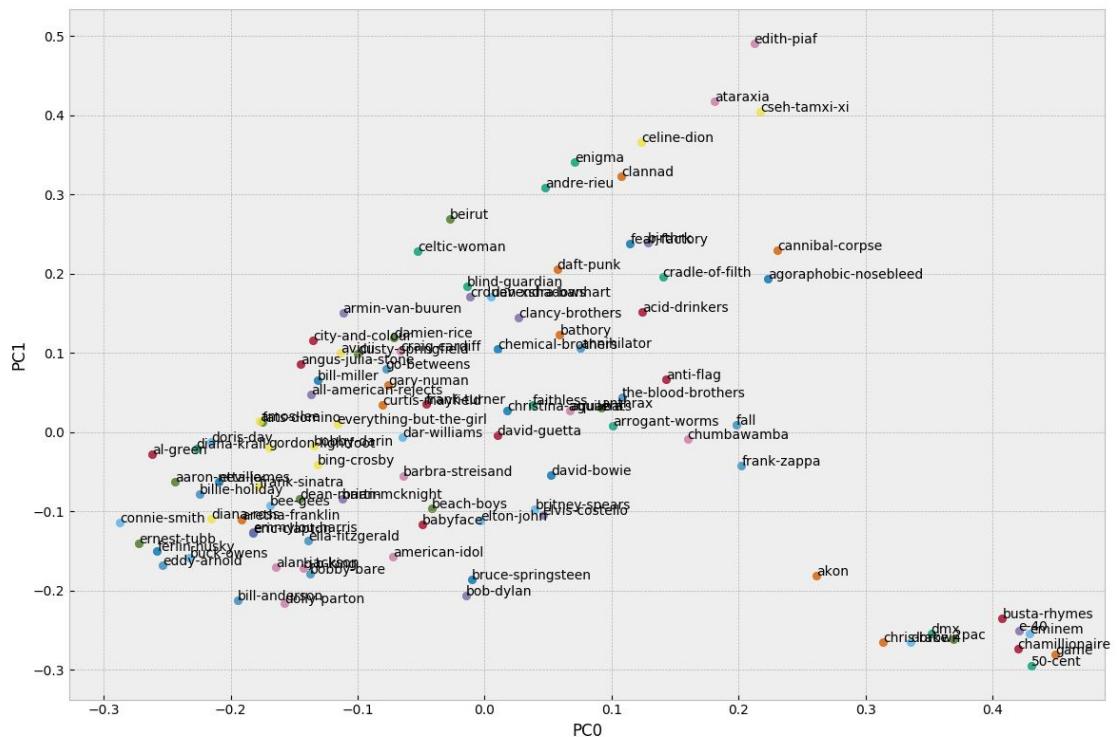
### PCA with StandardScaler with outliers



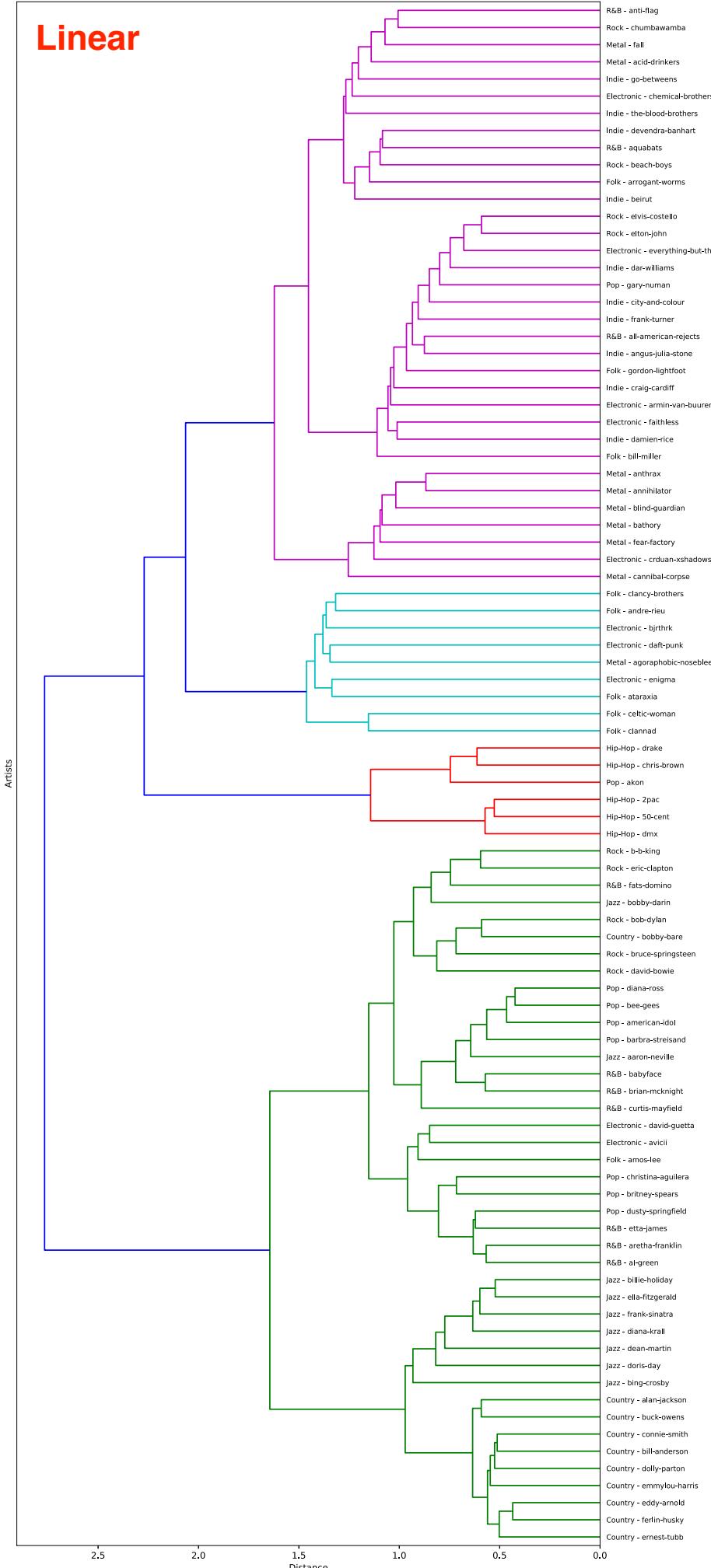
## PCA without StandardScaler without outliers



## PCA without StandardScaler with outliers



Hierarchical Clustering Dendrogram



Hierarchical Clustering Dendrogram

