IE1 - SKB Bericht - monteric, kabasbet

# Motivation

Da wir zusammen KI1 und KI2 besucht haben, war es für uns eine optimale gelegenheit das gelernte in einem neuen, unbekannten Umfeld anzuwenden. Mittels den damals gezeigten Verfahren des Unsupervised Learning können wir die Dokumente und Queries automatisch in «Topics» kategorisieren. Diese werden von den Algorithmen selbst bestimmt und haben so direkt keine Verbindung zu den eigentlichen Themebereichen. Ein Vorteil davon ist das durch diese Kategorisierung ein grossteil der Daten abgetrennt wird (prunnen), wodurch sich das eigentliche Ranking beschleunigt. Ein weiterer Vorteil wäre das solche Modelle kontinuierlich anhand des Nutzerverhaltens angepasst werden könnten, wodurch sich genauere Resultate erzielen lassen können.

# Verwendete Systeme

## Python mit Jupyter Notebook

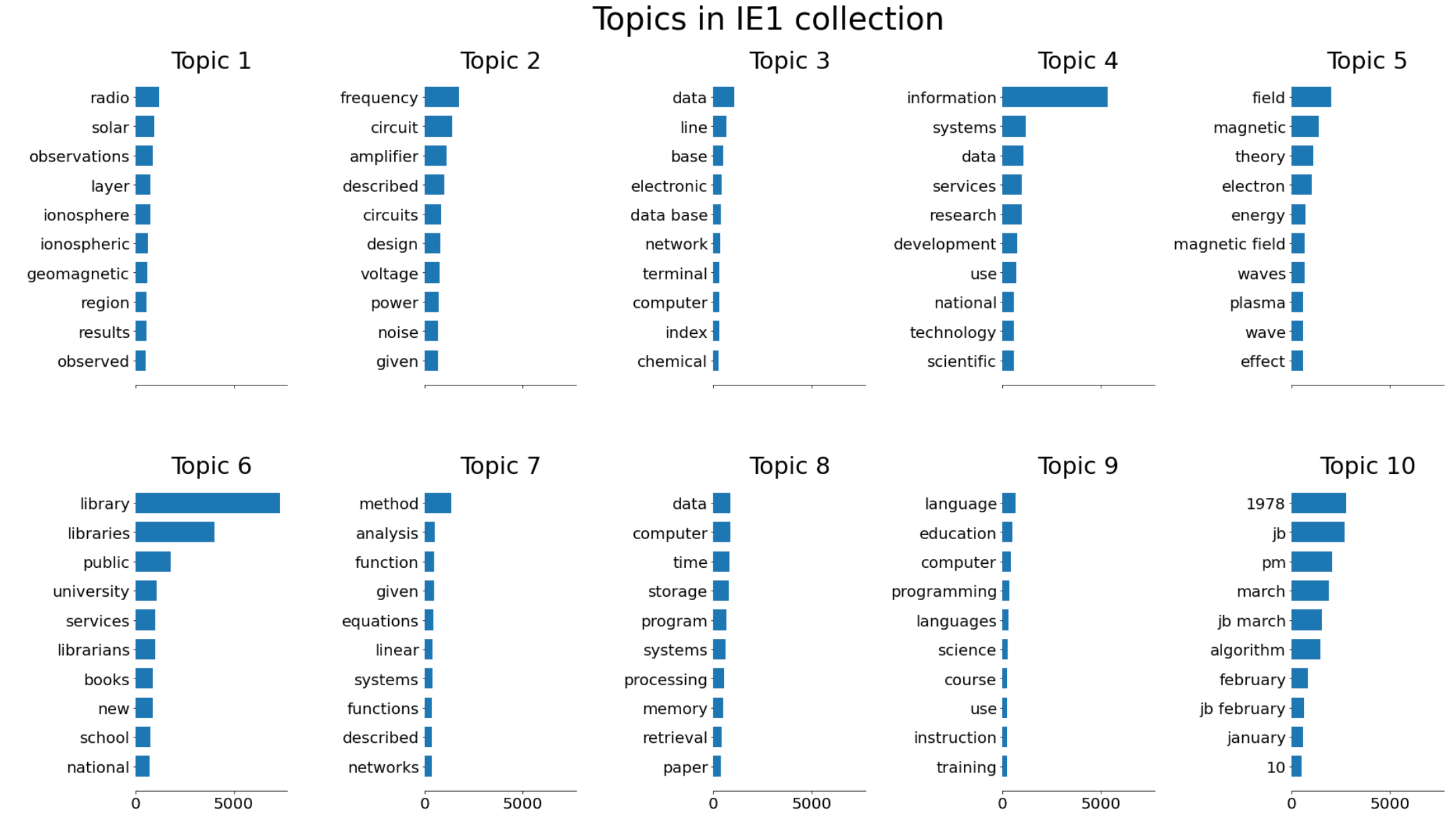
Als Programmiersprache wurde Python ausgewählt, da im Bereich ML hier am meisten verfügbar ist. Mit Jupyter konnten wir Zeitsparend das System aufbauen, dokumentieren und testen. Das anschliessend funktionierende System wurde als separates Skript aufgebaut, das mit diversen Parametern ausgeführt werden kann.

## Sklearn

Eine ML-Bibliothek, die verschiedene Klassifikations-/Regressions- und Clusteringalgorithmen beinhaltet. Wir haben folgende verwendet:

* CountVectorizer: Der CountVectorizer wird genutzt um die initialen Texte (Dokumente und Queries) in Matrizen zu wandeln. Hier werden automatisch Features aus den Texten gewählt und gezählt. Anhand bestimmer parameter wie Stop-Words, Anzahl features usw. lassen sich die resultierenden Matrizen optimieren.
* TfIdfVectorizer: Der TfIdfVectorizer macht das gleiche wie der CountVectorizer, nutzt dafür aber einen Algorithmus der die Häufigkeit der Wörter pro Dokument und Dokumentsammlung zählt, und daraus nur die wichtigsten nimmt (d.h. Wörter wie “und”, die oft vorkommen, werden nicht berücksichtigt).
* LatentDirichletAllocation: Die LDA kann mit diversen parametern optimiert werden. Die ganze Dokumentensammlung sowie die einzelnen Dokumente werden untersucht. Die entstehende Dirichlet-Verteilung sortiert die Dokumente nach Wahrscheinlichkeiten den Themen, woraus dann die mit den grössten Wahrscheinlichkeiten für unsere Suche genommen werden.
* pairwise\_distance: Diese funktion berechnet eine Distanzmatrix von zwei Vektoren. Auch hier gibt es mehrere parameter die genutzt werden können um die Distanz zu berechnen. Es wurde initial versucht mit der Funktion cosine\_similarity die Ähnlichkeit zu bestimmen, es wurde aber keine korrekte Funktionalität mit unserer Datenstruktur zustande gebracht.

# Verfahren

1. Lesen und Parsen der Inputfiles in ein Dictionary
2. Das System trainieren: Die Dokumente werden mit Hilfe eines Vectorizers (Count-oder TfIdf-Vecorizer) in eine Matrix transformiert. Dabei werden zuerst Stoppwörter eliminiert und die restlichen Wörter, je nach Verfahren, anders gewichtet bzw. ein Score gegeben. Ausserdem können wir steuern, wie viele Wörter/Features wir nutzen wollen.
3. Aus diesen Scores kann das System mittels LDA die Dokumente in n-Topics gruppieren. Das sieht dann so aus:  
     
   Die Wörter auf der Y-Achse sind die Features, sortiert nach dem Score, der im Schritt 2 vergeben wurde.
4. Die Dokumente werden dann den Topics zugewiesen und erhalten einen Score. Je höher der Score, desto eher passt das Dokument zum Topic
5. Das gleiche Verfahren in Schritt 2 & 3 wird dann auch für die Queries verwendet. Das heisst jede Query wird auch einem Topic zugewiesen.
6. Im letzten Schritt sucht das System für jede Query, mit Hilfe des Topics, nach möglichen dazugehörigen Dokumenten und berechnet die Distanz zwischen Query und Dokument. Je kleiner die Distanz desto eher passt die Query zum Dokument.

# Verwendeter Index

Der Index im Ranking wird anhand der Vektoren der einzelnen Dokumente/Queries erstellt. So wird von kleinster zur grössten Distanz, maximal aber 50 Elemente, sortiert.

# Resultate

## Mit Count Vectorizer

Das resultierende Ranking konnte durch Stichproben evaluiert werden und von uns als korrekt funktionierend betrachtet. Das eingeteilte Topic wird durchforstet und mittels Distanzberechnung das Ranking erstellt, wodurch zuoberst das beste erscheint.

## Mit Tf-Idf Vectorizer

Die Resultate mit TfIdf sind schlecht ausgefallen. Die berechnete Distanz ist praktisch identisch für alle Dokumente. Dies liegt daran, das TfIdf nach den relevantesten (in den meisten Fällen seltensten!) Wörter sucht. Dies führt dazu das die Queries und Dokumente alle auf den gleichen Inhalt beschränkt werden, wodurch diese sehr nahe beieinander liegen.

# Bemerkung

Das System ignoriert im jetzigen Zustand Themenübergreifende Dokumente sowie Suchanfragen. Die Unterstützung dessen ist aber nicht unmöglich – man könnte das System in überschaubarer Zeit anpassen, um auch dies zu ermöglichen. Auch ist das Durchforsten der Dokumente nicht bemerkenswert schnell.