The Automatic Creation of Literature Abstracts

Hans Peter Luhn, 1958

BJÖRN WILMSMANN

Inhalt

Inhalt

- Überblick
- Grundidee: Auswahl der repräsentativen Sätze eines Textes
- Relevante Terme und Relevanz von Sätzen
- Der Algorithmus
- Implementierung

Überblick

Überblick

- Hans Peter Luhn (1896-1964)
  - in den 1940ern bis 1960ern Informatiker bei IBM
  - über 80 Patente
  - Luhn Algorithmus (Prüfsummenverfahren)
  - KWIC (Keywords in Context, Concordancing)

Überblick

Überblick

- Abstracts: Ergänzen Titel, ermöglichen schnelle Auswahl relevanter Texte
- Erstes, wegweisendes Verfahren zur Textzusammenfassung
- Trotz des Titels: Keine Abstractgenerierung, sondern Auswahl der ‚besten‘ Sätze
- Vision von 1958: Abstracterstellung in Zukunft nur noch durch Maschinen

Überblick

Luhn über die Transkription gedruckter Texte in maschinenlesbare Fom:
‚For material not yet printed tape-punching devices attached to typewriters [...] could readily produce machine-readable records as by-products.‘...

Grundidee

Grundidee

- Auswahl der Sätze, die den Text am besten repräsentieren
- Ansatz: Beste Repräsentanten sind die Sätze, welche die meisten für den Text relevanten Terme enthalten; also: Auswahl der relevanten Terme
- Dann: Berechnen eines Maßes, das die Relevanz eines Satzes in Abhängigkeit von den darin enthaltenen Termen ausdrückt

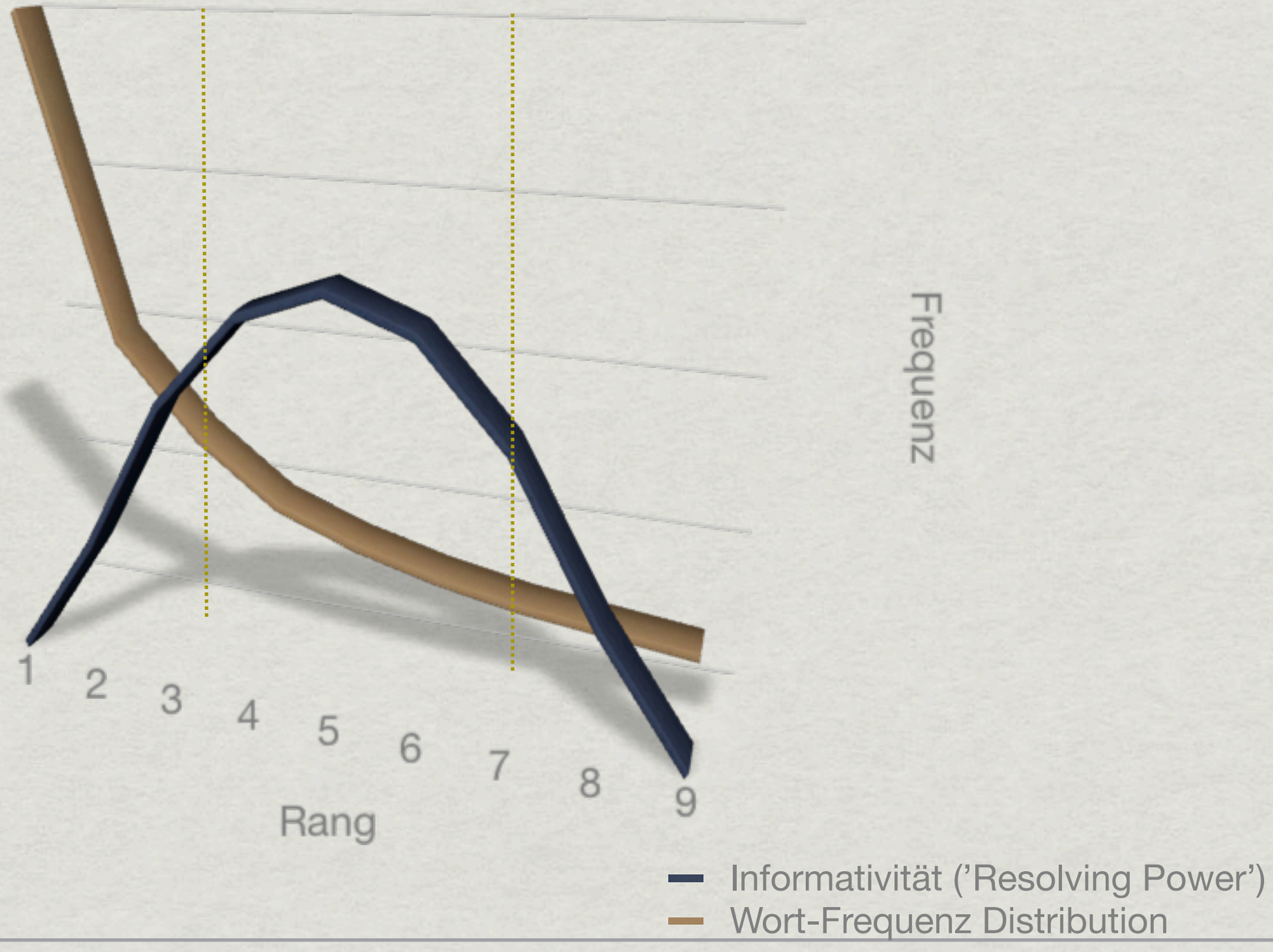
Relevante Terme

Relevante Terme

- Ansätze zur Filterung:
  - Stopwords: Sprachabhängig, müssen weitestgehend manuell erstellt und gepflegt werden
  - Anhand statistischer Eigenschaften: Rang der Termfrequenz, häufigste (Stopwords) und seltenste Worte werden entfernt, diese Variante nutzt Zipf-Verteilung

Relevante Terme

Relevante Terme



Relevanz von Sätzen

Relevanz von Sätzen

- Idee: Segmente, die zwischen zwei relevanten Termen liegen; relevante Terme dürfen höchstens durch 4 bis 5 nicht-relevante Terme getrennt sein

- Relevanzberechnung: $R_s = |T_r|^2 / |T|$ mit R_s = Relevanz eines Satzes, T_r, T Mengen der relevanten bzw. aller Terme in einem Segment, bei mehreren Segmenten zählt der höchste Wert (Anmerkung: Die Anzahl der Segmente bietet sich eigentlich auch als Parameter an.)

Der Algorithmus

Der Algorithmus

- Zerlege Text in Sätze
- Zerlege Sätze in Token
- Finde und filtere Types

Der Algorithmus

Der Algorithmus

- Für jeden Satz: Starte das aktuelle Segment bei Index 0
- Für jedes Token in einem Satz: Falls das Token zu den relevanten Termen gehört, füge es zur Liste der relevanten Terme für diesen Satz hinzu

Der Algorithmus

Der Algorithmus

- Berechne Relevanz des aktuellen Segments anhand der genannten Formel
- Falls die Differenz zwischen Laufindex und dem Index des letzten relevanten Terms > 4: Beginne neues Segment
- Bei mehreren möglichen Segmenten: Wähle das mit der höchsten Relevanz
- Abstract: Alle Sätze über Schwellenwert

Der Algorithmus

Der Algorithmus

- Abstraktes Beispiel:
  - (_ _ [* _ _ _ * _ _ * *] _ _ _ _ _ [* _ *] _ _ _ _ _)
    - *: Relevant
    - _: Nicht relevant

Der Algorithmus

Der Algorithmus

- Konkretes Beispiel aus dem Wikipedia Text über Hans Peter Luhn:
  - He joined IBM as a senior research engineer in 1941, and soon became manager of the information retrieval research division.

Der Algorithmus

Der Algorithmus

- Relevante Worte (intuitiv gewählt):
  - senior, engineer, manager, information, retrieval
- Segmentierung
  - He joined IBM as a [senior* research engineer*] in 1941, and soon became [manager* of the information* retrieval*] research division.

Der Algorithmus

Der Algorithmus

- Berechnung der Scores für die Segmente
  - [senior* research engineer*]:
    - $2^2 / 3 = 1,\overline{3}$
  - [manager* of the information* retrieval*]:
    - $3^2 / 5 = 1,8$
- Satz hat eine Relevanz von 1,8

Der Algorithmus: Laufzeitverhalten

Der Algorithmus: Laufzeitverhalten

- Eigene Analyse:
- Lineares Laufzeitverhalten: O(n)
- n = |T|
- T = Menge der Token im Text

Der Algorithmus: Fine Tuning

Der Algorithmus: Fine Tuning

- Filterung durch Stopword Lexikon
- Anschließender Filterschritt anhand von statistischen Eigenschaften
- Verfahren zur Abbildung orthographisch ähnlicher Terme auf gemeinsamen abstrakten Term (in der Zielsetzung ähnlich wie Stemming)
- Methoden verbessern Recall, Precision, sparen Speicher

Implementierung

Implementierung

- Framework: Grails
- Algorithmus selbst wurde in ca. 20 Zeilen implementiert
- Implementation mit reiner Filterung nach Termfrequenz und ohne Abbildung auf abstrakte Terme

Zusammenfassung

Zusammenfassung

- Einfaches, intuitives Verfahren
- Abstracts geben für viele Texte einen guten Überblick über den Inhalt
- Eigene Anmerkungen:
  - Texte sind nicht standardisiert, weswegen die Qualität stark variiert
  - Abstracts wirken unnatürlich, da eher inkohärent

FRAGEN?