

Patent - Triz40 Mappingansätze

WS19 - Widersprüche und Management-Methodiken (WUMM)



Gliederung

1. Outline
2. Vorverarbeitung
3. Ähnlichkeit
4. TF-IDF
5. Wordnet
6. W2V
7. Evaluierungsmöglichkeiten
8. Weitere Ansätze



Outline

- **Daten:** Patenttexte und Triz40 Methoden (englisch, deutsch), Originaltexte (russisch)
- **Problem:** Klassifizierung von Patent Texten in 40 Klassen (Triz40)
- **Ansatz:** NLP, TextMining und Information Retrieval Algorithmen zur Klassifizierung
 1. TF-IDF
 2. Wordnet
 3. Word2vec



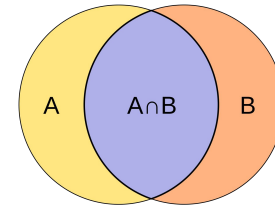
Vorverarbeitung

- Stopwörter entfernen (und, oder, der, die, das)
 - via listen
- Wörter auf den Wortstamm reduzieren (Häuser -> Haus, gesprungen -> springen ...)
 - z.B. via Porterstemmer (englisch)
- POS Tagging
 - z.B. via OpenNLP (deutsch) , NLTK (englisch)

Ähnlichkeit

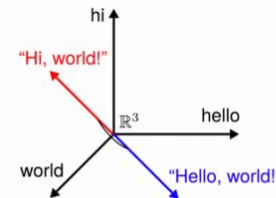
- Jaccardähnlichkeit

$$J_{\delta}(A, B) = 1 - J(A, B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}$$

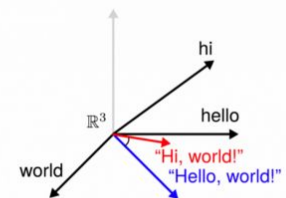


- Kosinusähnlichkeit

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$



Cosine Similarity



Soft Cosine Measure

Source: https://github.com/Karim-Teknologies/presentation/blob/master/slides/soft_cosine_similarity.ipynb



Termfrequenz - inverse Document Matrix

- Termfrequenz matrix

$$\text{tf}(t, D) = \frac{\#(t, D)}{\max_{t' \in D} \#(t', D)}$$

- inverse Dokumenthäufigkeit

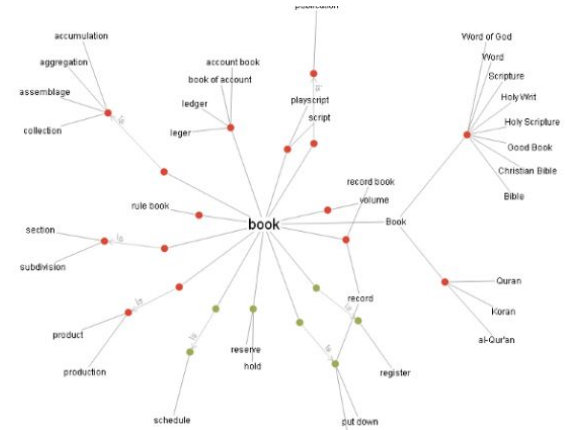
$$\text{idf}(t) = \log \frac{N}{\sum_{D: t \in D} 1}$$


- tfidf-Maß

$$\text{tf. idf}(t, D) = \text{tf}(t, D) \cdot \text{idf}(t)$$

Wordnet

- wissensbasiert
- von der Princeton Universität seit 1985 entwickelt
- auflösung von Synonymen nach Wortsemantik
- Graph aus Wörtern (kanten entsprechen ähnlichkeit)



- 
- Wortähnlichkeit != Textähnlichkeit
 - *Corpus-based and Knowledge-based Measures of Text Semantic Similarity*
(Mihalcea et al.)

$$\text{Sim}(T_1, T_2) = \frac{1}{2} \left(\frac{\sum_{w \in T_1} \max\text{Sim}(w, T_2) \cdot \text{idf}(w)}{\sum_{w \in T_1} \text{idf}(w)} + \frac{\sum_{w \in T_2} \max\text{Sim}(w, T_1) \cdot \text{idf}(w)}{\sum_{w \in T_2} \text{idf}(w)} \right)$$



Word2Vec

- Verfahren des maschinellen Lernens
- entwickelt von Google
- sogenanntes Wordembedding
- löst semantische Zusammenhänge basierend auf einem Trainingsmodell auf
- spannt Vektorraum auf und vergleicht über Kosinusähnlichkeit



Evaluierungsmöglichkeiten

- UPSTO / EPO Datensätze klassifizieren und händisch evaluieren
- Beispiele aus dem Seminar?
- ggf. Datenaufbereitung / Modelanpassungen / Datensatzerweiterung ...

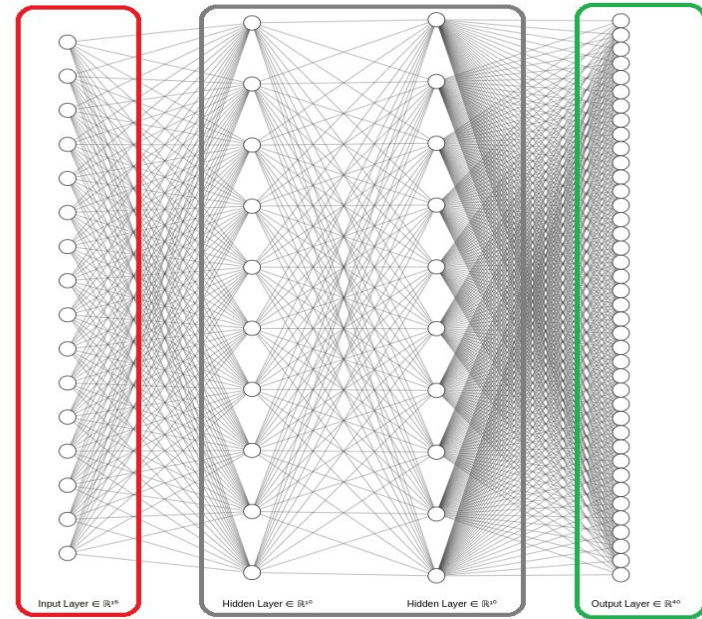


Weitere Ansätze

- Glove (stanford)
- Latent Semantic Indexing
- Embeddings + Tensorflow?

Neuronales Netzwerk

- Wordembeddingschicht
- n - Scramble + Dropoutschichten
- weitere?
- Ausgabeschicht





Quellen

- <https://de.wikipedia.org/wiki/WordNet>
- <https://de.wikipedia.org/wiki/Tf-idf-Ma%C3%9F>
- <https://de.wikipedia.org/wiki/Jaccard-Koeffizienttutorials>:
- <https://medium.com/@adriensieg/text-similarities-da019229c894>
- <http://www.lumenai.fr/blog/quick-review-on-text-clustering-and-text-similarity-approaches>
- <https://rare-technologies.com/word2vec-tutorial/>



Quellen

- <https://nlpforhackers.io/wordnet-sentence-similarity/>
- <https://radimrehurek.com/gensim/models/word2vec.html>
- <https://radimrehurek.com/gensim/models/doc2vec.html>
- <https://arxiv.org/ftp/arxiv/papers/1510/1510.02755.pdf>
- <https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf>
- <https://www.sciencedirect.com/science/article/pii/S0957417414006472>