24.02.2017

Machine Learning WS 16/17 Abschlusspräsentation Projekt: <wikification>

Alexander Haas & Sven Bodemer

Hochschule RheinMain
Dozent: Prof. Ulges

Gliederung

- Datengrundlage
- Wikification – Ansatz
  - Technologien
  - Architektur
- Evaluation
- Fazit
- Ausblick

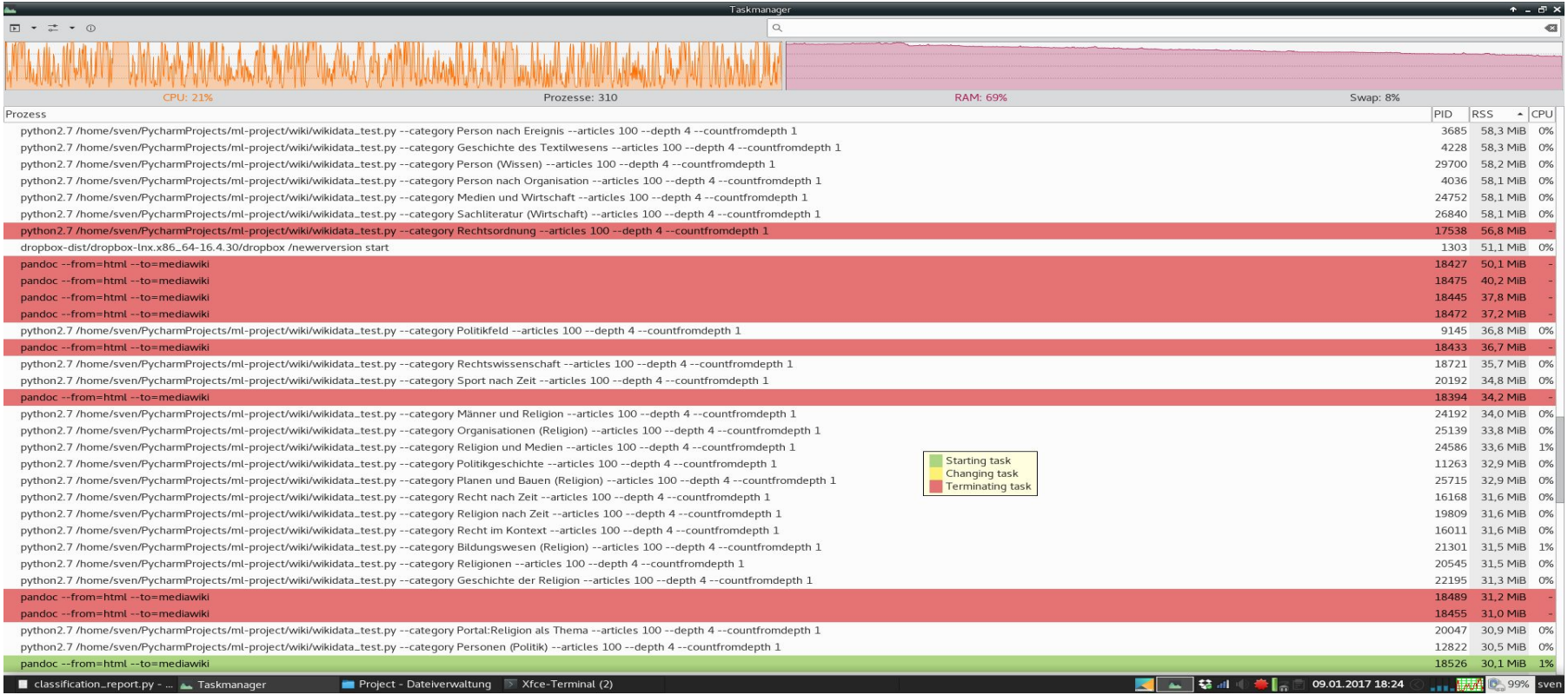
Datengrundlage

- Eingabe: unbekannter Text
- Ausgabe: Kategorie
- Crawling der 8 Hauptkategorien + Subkategorien (1 Mio. Artikel!)
- Normalisierung der Daten
  - NLTK-Stopwords, Snowball-Stemmer, Tokenizer, PyPandoc, RegExp
- Crawling + Normalisierung in einem Schritt (~15 h !)

Datengrundlage

- Artikel wurden von HTML → Wiki-Markup umgewandelt und im Dateisystem gespeichert
  - Keine Datenbank, kein Server!
  - Kategorien Ordner
  - Artikel Dateien (z.B. algorithmus.wiki)
- Warum? → “rapid prototyping”
- Vorteil: erleichterte Fehlersuche, übersichtlicher

Systemauslastung während des Crawlings



Wikification-Ansatz

Die Daten sind gesammelt, aber wie geht es jetzt weiter ?

Quelle: https://upload.wikimedia.org/wikipedia/commons/thumb/e/e7/Wikify.svg/1280px-Wikify.svg.png

Unser Wikification-Ansatz

01 **Klassifikation nach Kategorie**
Welche Kategorie hat ein Text ?

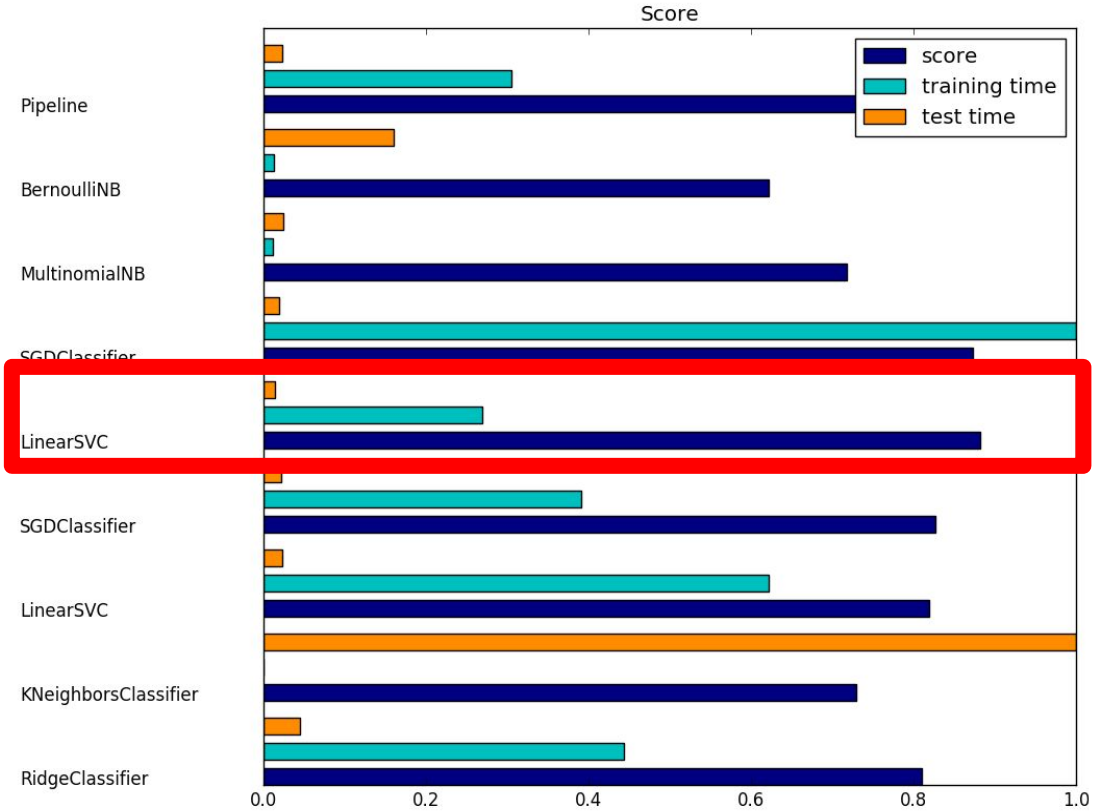
02 **Link-Detektor**
Was wird verlinkt, und was nicht?

03 **Link-Unterscheidung**
Eindeutige/Mehrdeutige Links

04 **Disambiguation**
Commonness & Relatedness

05 **Dashboard**
Eingabe von neuen Texten,
Visualisierung der Verlinkungen

01 - Klassifikation n. Kategorie



- Evaluation, welcher Klassifikator für unsere Problemstellung am Besten geeignet ist
- SVM (LinearSVC) bestes “scoring” im Verhältnis zur Trainingszeit

UI - Klassifikation n. Kategorie

- Eingabe: neuer, unbekannter Text
- Ausgabe: Kategorie

- Text-Klassifikator für Kategorien
  - TfidfVectorizer + LinearSVC (sklearn)
  - 60% Training, 40% Test
  - One-Vs-Rest

02 - Link-Detektor

$$LP = \frac{\#\text{Links von t in Wiki}}{\#\text{Vorkommen von t in Wiki}} \approx \frac{API\text{-}Call(\#\text{Vorkommen von t})}{C* \#\text{Vorkommen in Freq.-Liste}} \geq T$$

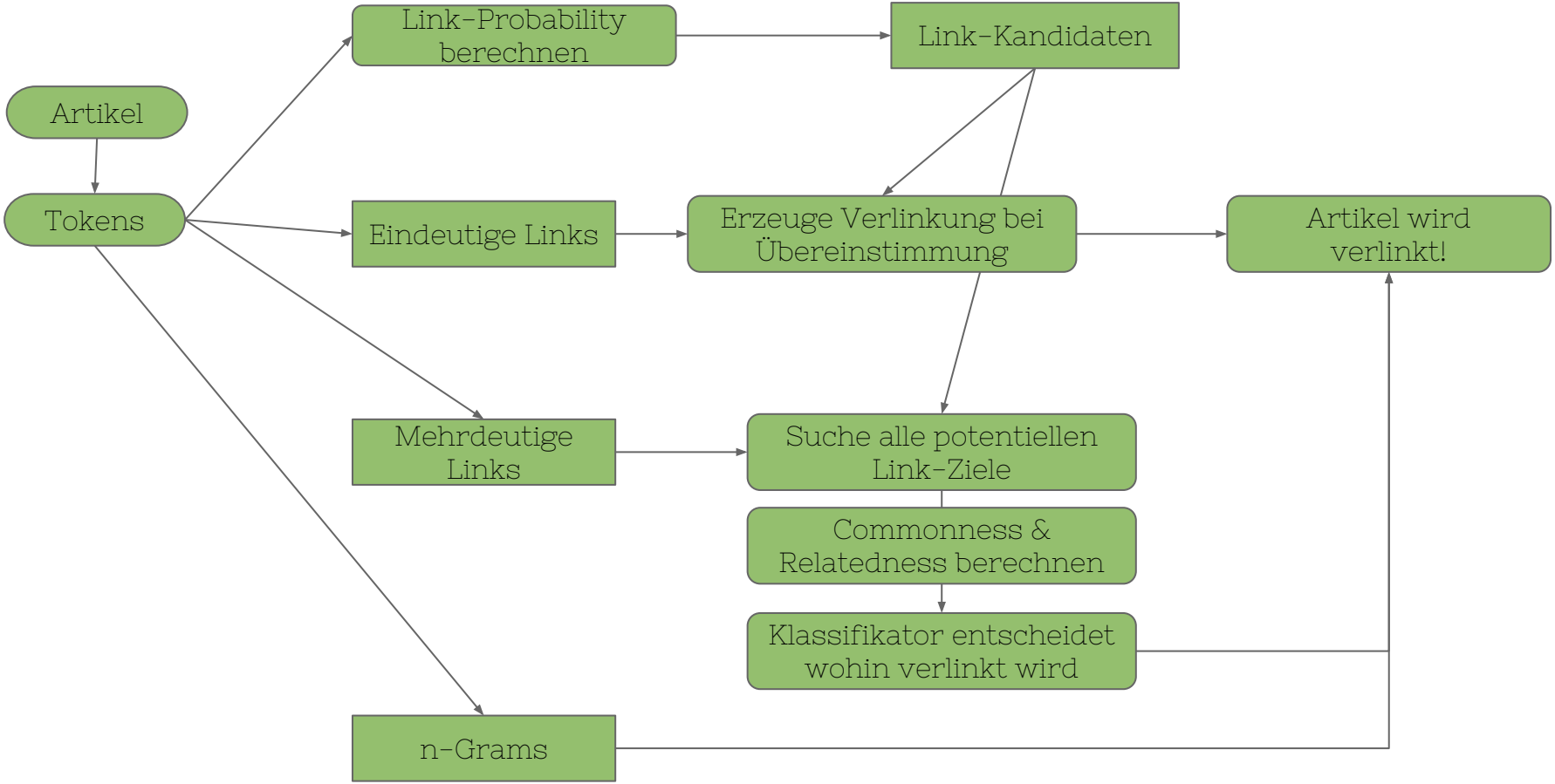
- LP □ Link Probability
- Basis: “Word-Frequency-List” (700k Wörter) & API
  - Freie Parameter: [C = 2.25, T = 0.5]
  - Bsp. t = 'tolkien'
  - API-Call(#t) = 34
  - #t in Freq.-List = 4
  - 34 / (2.25 * 4) □ 3.77 > T

👍 ‘tolkien’ ist Link-Kandidat!

03 - Link-Unterscheidung

- Je nach Link-Typ wird unterschiedlich erkannt:
  - Eindeutige Links (z.B. Informatik)
  - Mehrdeutige Links (z.B. Baum)
  - N-grams: Link aus mehreren Worten
- N-grams: idR. eindeutig und wichtig, sodass immer verlinkt wird
- Eindeutige Links: Fast jedes Wort ist ein möglicher Kandidat. Je nach Link-Probability wird über Verlinkung entschieden.

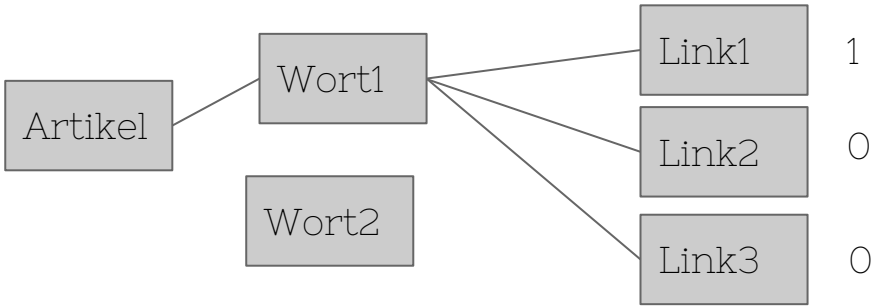
Link-Unterscheidung, aber wie!?



04 - Disambiguation

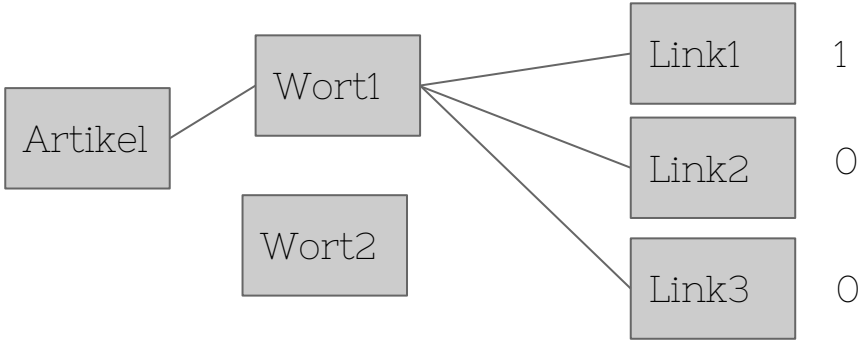
1. Mehrdeutige Links: Ein Wort kann auf mehrere unterschiedliche Artikel verwiesen werden
2. Commonness: zählen der Backlinks aller Link-Kandidaten
3. Relatedness:
   a. Links: Alle eindeutigen Links des Eingabe-Textes und Ziel-Artikel werden verglichen.
   b. Text: Alle Worte beider Texte werden verglichen.
   c. Klassifikation: Eingangstext und Ziel-Artikel werden von 8 Klassifikatoren bewertet.

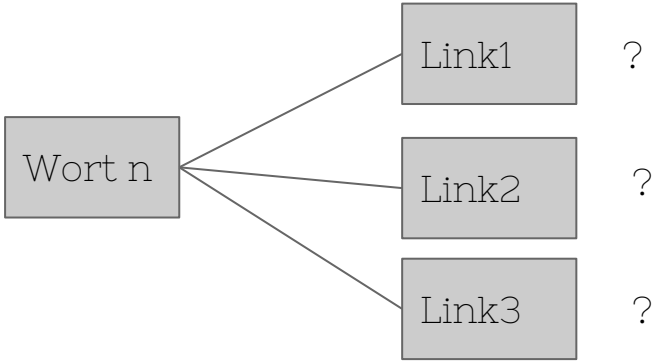
04 - Disambiguation



	Commonness	Relatedness Links	Relatedness Text	Classifier (1) Wissen	...	Classifier (8) Sport	Label
Wort 1 (train)	backlinks(Link 1) / backlinks(all links)	Artikel ∩ Link1	Artikel ∩ Link1	Artikel – Link1	...	Artikel – Link1	1 or 0

04 - Disambiguation





	Commonness	Relatedness Links	Relatedness Text	Classifier (1) Wissen	...	Classifier (8) Sport	Label
Wort 1 (train)	backlinks(Link 1) / backlinks(all links)	Artikel ∩ Link1	Artikel ∩ Link1	Artikel – Link1	...	Artikel – Link1	1 or 0
Wort n (detect)	Backlinks (link1) / (link 1+2+3)	Text ∩ Link1	Text ∩ Link1	Text – Link1	...	Text – Link1	Decision by Link classifier

Evaluation

Wie gut konnte unser System eine Verlinkung erzeugen?

Quelle: https://upload.wikimedia.org/wikipedia/commons/thumb/e/e7/Wikify.svg/1280px-Wikify.svg.png

Evaluation

- 77% der Artikel werden richtig klassifiziert
- 54% der Links ‘matchen’
- 78% der Links wurden korrekt Disambiguiert
  (kl. Datensatz 10 Artikel!)
  - Wikipedia-API hat disambiguation erleichtert/eingeschränkt

Fazit & Ausblick

Wie verlief die Entwicklung dieses Systems? Was sind die Stärken und Schwächen? Welche offenen Punkte gibt es noch ?

Quelle: https://upload.wikimedia.org/wikipedia/commons/thumb/e/e7/Wikify.svg/1280px-Wikify.svg.png

Lessons learned

- Unerwartet viele Unicode-Probleme
- Bei großen Datenmengen von Anfang an Laufzeit betrachten
- Von Anfang an klare Strukturen: Config mit allen relevanten Pfaden
- Training von anfänglich 580 Klassifikatoren für die Kategorisierung :(
- Parallelisierung des “Crawling” (500k in ~15h)
- API-Calls immer noch sehr teuer

Ausblick

- API-Calls kosten zu viel Zeit
  - Auf Web-Requests sollte komplett verzichtet werden
  - Datenbank wäre die bessere Design-Entscheidung gewesen
- Tests waren meist auf kleinen Artikeln, da große zu viel Zeit gekostet hätten

Live-Demo

Quelle: https://commons.wikimedia.org/wiki/File:10_sharing_book_cover_background.jpg

Quelle: https://beckysblogs.files.wordpress.com/2012/10/frodo.jpg