Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

Wikiplag

01.02.2019

Erkennung von Plagiaten aus der deutschen Wikipedia mittels eines neuronalen Netzes

Andrej Loparev, Bach Do, Claudio Vindimian, Max Williams, René Strietzel, Samuel Erb, Steven Mi, David Ketels

Betreuung: Prof. Dr.-Ing Hendrik Gärtner

Fragestellungen

- Wie sind die Wikipedia-Artikel gespeichert?
- Was ist die Benutzerschnittstelle?
- Wie ermitteln wir Plagiate?

Gliederung

- Frontend
- Systemaufbau, verwendete Technologien
- Datenbank
- Vorauswahl
- Plagiatserkennung
  - Inspirationsquelle → Wie kamen wir auf den Lösungsansatz
  - Input für das neuronale Netz
  - Neuronales Netz
  - Evaluation
- Die nächsten Schritte

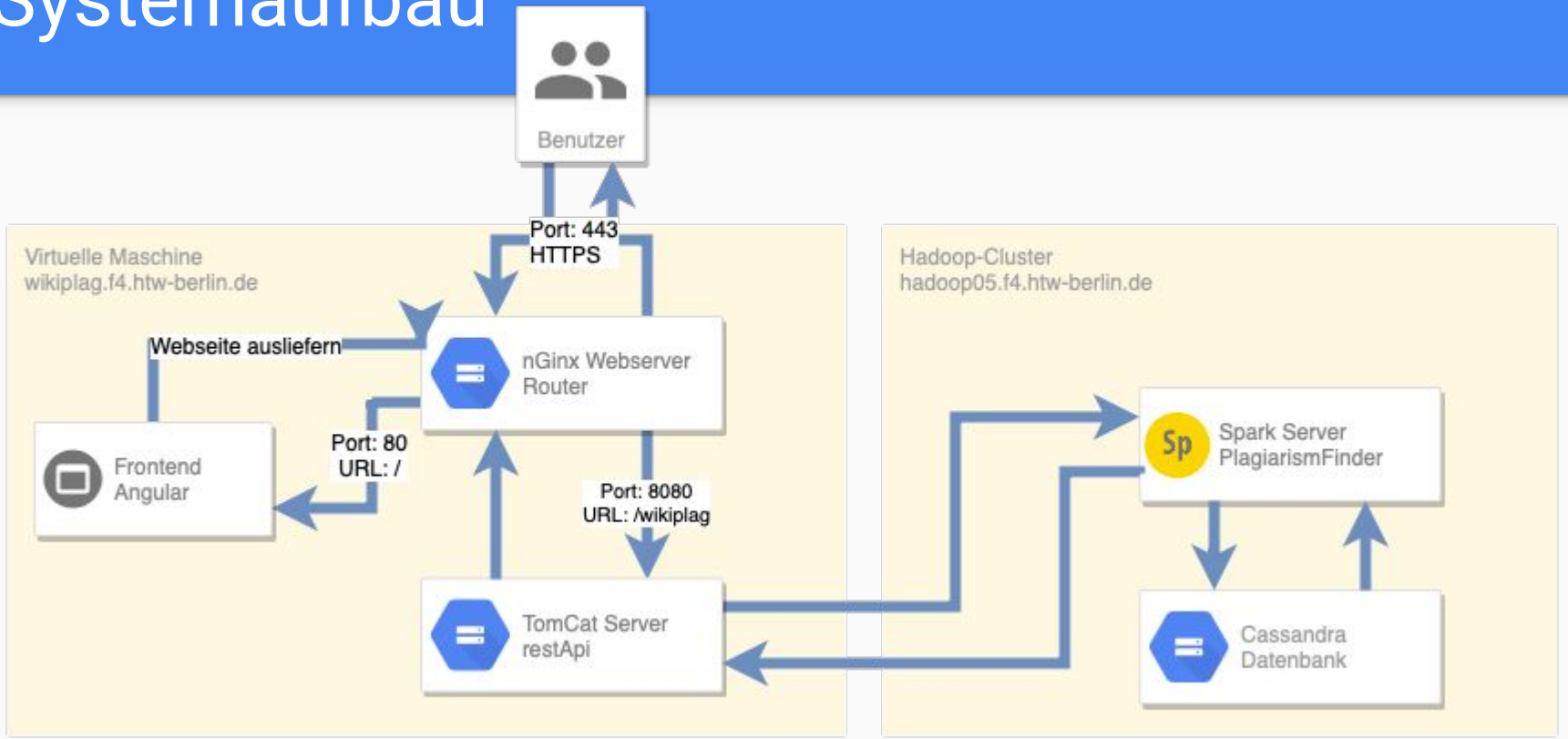
Frontend

Frontend

- https://wikiplag.f4.htw-berlin.de/
- Erstellung mit Angular 6
- 3 Hauptkomponente:
  - Input : Texteingabe, Wort-counter, Search history
  - Output: Eingabe Text, Ergebnis, pdf-Generator
  - Navigationbar

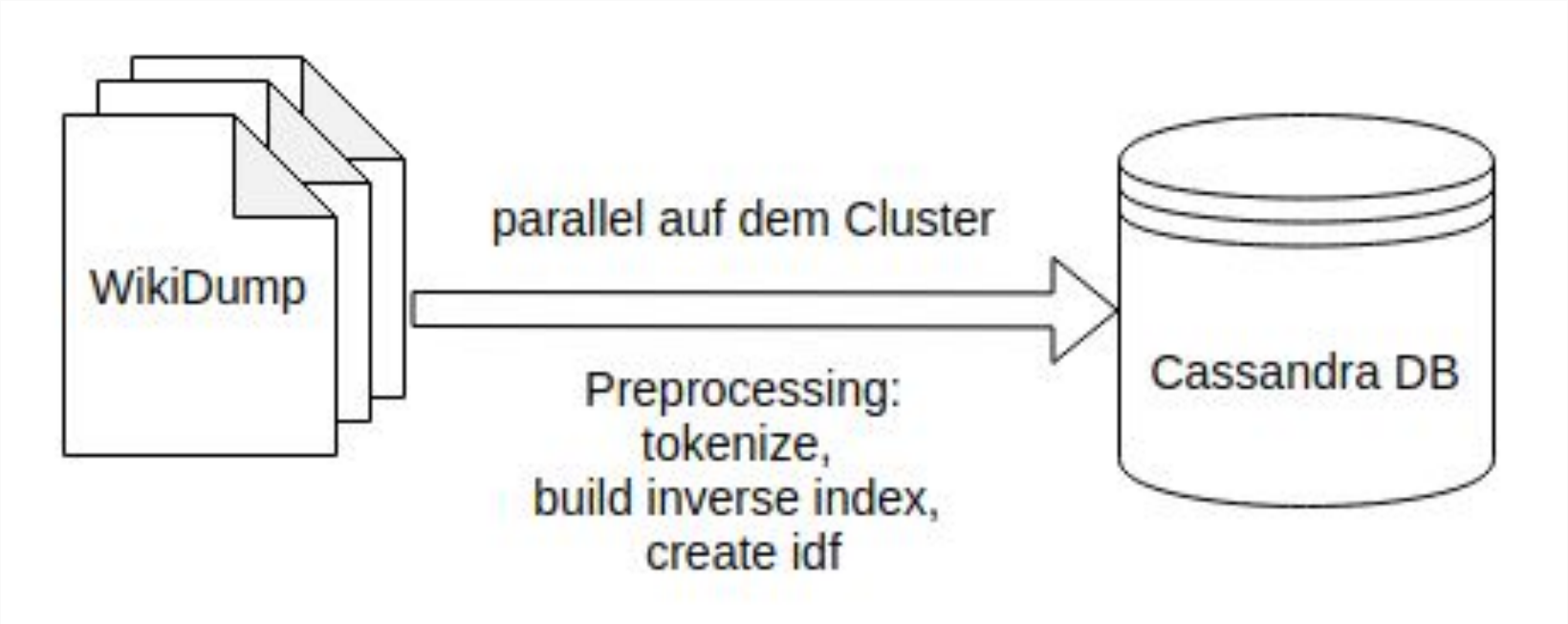
Systemaufbau

Systemaufbau



Datenbank

Speicherung der Wiki-Inhalte



Tokenisierung: Beispiel

Satz:

Archaeopteryx gilt als Übergangsform, die zwischen theropoden Dinosauriern und den Vögeln vermittelt.

Tokens:

'archaeopteryx', 'gilt', 'als', 'übergangsform', 'die', 'zwischen', 'theropoden', 'dinosauriern', 'und', 'den', 'vögeln', 'vermittelt'

Vorauswahl

Idee

Die Anzahl der zu vergleichenden Dokumente zu verringern

Idee

Die Anzahl der zu vergleichenden Dokumente zu verringern

Lösungsmöglichkeiten:

- nur die Dokumente weitergeben in denen die Wörter vom Plagiat/Input auftauchen
- Problem: nicht alle Wörter im Plagiat/Input sind relevant

Problem: Relevanz der Wörter

nicht alle Wörter im Plagiat/Input sind relevant

Problem: Relevanz der Wörter

nicht alle Wörter im Plagiat/Input sind relevant

Lösungsmöglichkeiten

- Gewichtungen in Wörter einfügen
- Basierend auf die Gewichtungen die Wichtigkeit der Wörter feststellen

Bewertungskriterium - IDF

- Inverse Document Frequency/Inverse Dokumenthäufigkeit

$$IDF_t = \frac{N}{f_t} \qquad IDF_t = log\left(\frac{N}{f_t}\right)$$

N = Anzahl der Dokumente

f_t = Anzahl der Dokumente in dem Term t auftaucht

Bewertungskriterium - IDF

- Inverse Document Frequency/Inverse Dokumenthäufigkeit

$$IDF_t = \frac{N}{f_t} \qquad IDF_t = log\left(\frac{N}{f_t}\right)$$

N = Anzahl der Dokumente

f_t = Anzahl der Dokumente in dem Term t auftaucht

- Je höher der Wert desto weniger taucht ein Wort in allen Dokumenten auf
- nur die Terme mit einem höchsten IDF Wert sind relevant
  - Terme mit einem genringen IDF Wert sind meistens Stopwörter
  - Stopwörter: Wörter die häufig auftauchen aber keine Bedeutung haben z.B. der, die, das,

Bewertungskriterium - IDF

Dokumente:

- Dokument 1: Ich studiere Informatik
- Dokument 2: Ich studiere BWL
- Dokument 3: Informatik ist Informatik

IDF - Werte

- Ich = 3/2, studiere = 3/2, Informatik = 3/2, BWL = 3/1, ist = 3/1

Umsetzung

- Vorbereitung: für alle Wörter die in Wikipedia Artikel auftauchen wird der IDF Wert berechnet und in der Datenbank gespeichert
  - Inverser Index: Wort -> IDF Term

Umsetzung

- Vorbereitung: für alle Wörter die in Wikipedia Artikel auftauchen wird der IDF Wert berechnet und in der Datenbank gespeichert
  - Inverser Index: Wort -> IDF Term
- Vorselektion: nur die Dokumente weiterleiten welche die n-Wörter enthalten die den höchsten IDF Wert haben
  - n: wie viele “wichtige” Wörter wir beachten wollen

Umsetzung

List(**kimas**, **nachgeahmte**, **denkprozess**, tamagotchi, **entscheidungsstrukturen**, solver, **menschenähnliche**, ungewissen, gegebenes, projektarbeit, skripte, teilgebieten, **menschlichem**, empires, **projektgruppe**, newell, output, aufsetzen, erschaffung, computerspielen, wahrzunehmen, **intelligent**, herauszufinden, **algorithmen**, **implementiert**, definitionen, modifikationen, fazit, ausblick, zeige, denkens, vorstellen, denkt, physiologie, wohingegen, psychologische, kurzes, intelligenz, befassen, **naturwissenschaftlichen**, **ideal**, **realisierung**, breites, gebe, künstliche, richtige, konzepte, unbedingt, selbständig, wichtiges)

- **50 wichtigsten Wörter bei einem Plagiat über künstliche Intelligenz**

Plagiatserkennung: Inspiration

Plagiatserkennung: Inspiration

- Quelle
  - Name: Plagiarism Detection Framework using Monte – Carlo Based Artificial Neural Network for Nepali Language, Januar 2018
  - Forscher: Rakesh Kumar Bachchan, Arun Kumar Timalsina
  - Link: www.rroij.com/open-access/plagiarism-detection-framework-using-monte-carlo-based-artificial-neural-network-for-nepali-language.pdf
- Konzept
  - Vergleichspaare: Satz / Absatz (Bei uns 13-Grame)
  - Vergleichsmaße: Kosinus-, Jaccard-Ähnlichkeit
  - Input für NN: Kosinus, Jaccard, zu vergleichende Textabschnitte
  - Output des NN: Plagiat/Kein Plagiat

Input des neuronalen Netzes

Input des neuronalen Netzes

Was haben wir?

- den zu überprüfenden Text
- Liste von Artikeln, die als potenzielle Plagiate infrage kommen

Was wollen wir?

- Berechnung des **Jaccard-Koeffizienten** und **Kosinus-Ähnlichkeitswert** für alle möglichen **N-Gramm**-Paare

N-Gramm

- Text wird in Fragmente zerlegt
- N aufeinanderfolgende Fragmente werden als N-Gramm zusammengefasst
- Fragmente können sein: Buchstaben, Wörter, Laute
- in unserem Fall -> Wörter

N-Gramm: Beispiel

Satz: *Das Auto, das auf der anderen Straßenseite steht, ist rot.*

3-Gramme:
{Das Auto das; Auto das auf; das auf der; auf der anderen; der anderen Straßenseite; anderen Straßenseite steht; Straßenseite steht ist; steht ist rot}

Jaccard-Koeffizient

- Kennzahl für die Ähnlichkeit von Mengen
- Formel:

$$\frac{\text{Anzahl der gemeinsamen Elemente (Schnittmenge)}}{\text{Größe der Vereinigungsmenge}}$$

Jaccard-Koeffizient: Beispiel

Menge A = { das, ist, ein, ziemlich, schönes, Haus }

Menge B = { das, ist, ein, sehr, altes, Haus }

Anzahl Schnittmenge = | {das, ist, ein, Haus} | = 4

Größe Vereinigungsmenge = | {das, ist, ein, ziemlich, sehr, schönes, altes, Haus} | = 8

=> Jaccard = 4/8 = **0.5**

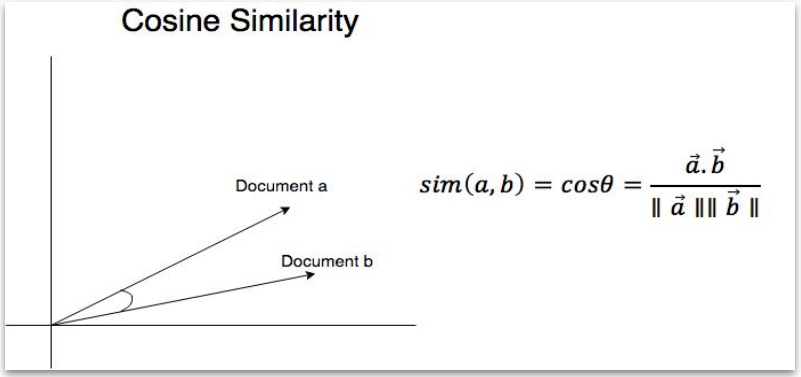
Der Jaccard-Koeffizient reicht von 0 bis 1. Je näher der Jaccard-Koeffizient an der 1 liegt, desto ähnlicher sind sich die beiden Mengen.

Kosinus-Ähnlichkeit

- N-Gramme werden zu Vektoren umgewandelt
- Kosinus des Winkels zwischen zwei Vektoren wird bestimmt

Implementierung

- Term Frequency - Wie oft kommt ein Wort vor?
- IDF - Wie wichtig ist ein Term?
- Vektoren gebaut
- Cosinus Formel anwenden



Kosinus-Ähnlichkeit: Beispiel

A = { 2, 3, 4, 5 }

B = { 0, 3, 2, 1 }

A * B = 22

||A|| * ||B|| = 7.34 * 3.74 = 27.49

= Cos => 0.80

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

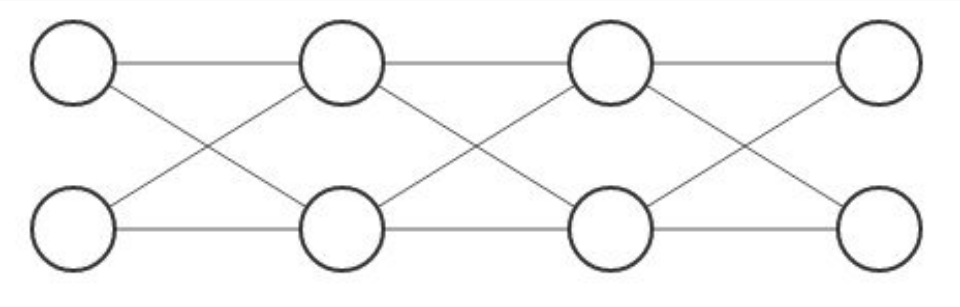
Output

N-Gramm 1	N-Gramm 2	Jaccard	Kosinus
String (Userinput)	String (Wiki)	Double	Double

Neuronales Netz

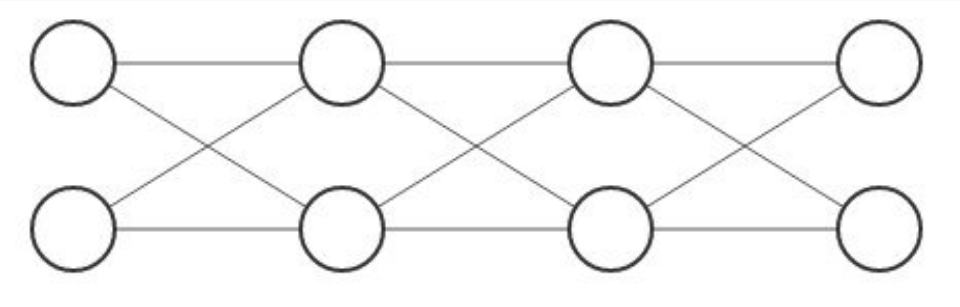
Neuronales Netz

- Input des Netzes:
  - Kosinus- und Jaccard-Ähnlichkeit
- Output des Netzes
  - Plagiat / kein Plagiat



Neuronales Netz

- Data
  - Trainieren: 1/3
  - Evaluieren: 1/3
  - Testen: 1/3
- Datenaufbau
  - Eigenes gelabeltes Beispiel
  - Plagiat: 120 Paare
  - Keine Plagiat: 308 Paare



Evaluation

Evaluation

- Accuracy: **0.9097**
  - (tp + tn) / (tp + fp + tn +fn)
  - 0.97 bei Bachchan und Timalsina
- Precision: **0.9128**
  - tp / (tp + fp)
- Recall: **0.9097**
  - tp / (tp +fn)

tp = True Positive

tn = True Negative

fp = False Positive

fn = False Negative

Fazit: Das Konzept Funktioniert

Die nächsten Schritte

Die nächsten Schritte

TODO

- Preprocessing optimieren
- Vorauswahl optimieren
- Mehr Trainingsdaten
  - Höhere Modellkomplexität ermöglichen
- Inbetriebnahme

- Dokumentation: https://github.com/WikiPlagWS2018

Selbstreflexion

Vielen Dank!

Quelle: http://www.hawlina.de/?attachment_id=386

Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences