Eruierung von Methoden zur Exploration von Textwiederverwendung in großen Datenmengen am Beispiel der Wikipedia

Verteidigung Bachelorarbeit

Tristan Licht

Gutachter : Junior-Prof. Dr. Matthias Hagen, Prof. Dr.-Ing. Norbert Siegmund
Betreuer : Dr. Martin Potthast, Michael Völske

Bauhaus-Universität Weimar

Ablauf

1. Einleitung
2. Textabbildungen und Ähnlichkeiten
3. Implementierung
4. Ergebnisse

Einleitung

Ziele:

- Quantitive Aussage über den Anteil von einzigartigen Textinhalten in Textmengen treffen.
- Einen Ausblick über die Arten von Textwiederverwendung liefern.
- Methoden des Information Retrieval in Hinblick auf die Erkennung von Textwiederverwendung zu analysieren.

Datensätze:

- ClueWeb12 - 733 Millionen Seiten
- Wikipedia - 5.14 Millionen Seiten
- Common Crawl - 2 Milliarden Seiten

Textabbildungen

Bedingungen:

- Der Kreuzvergleich der Artikel des Wikipedia-Datensatzes umfasst 13 Billionen Operationen.
- Ein Vergleich aller Artikel auf Wortebene aller Artikel ist nicht schnell genug berechenbar.

Ziel:

- Für jedes Dokument eine Kandidatenmenge ermitteln, die alle anderen Dokumente des Korpus mit gleichen Textabschnitten umfasst.
- Eine Textrepräsentation finden, deren Vergleichsoperation hinreichend schnell berechenbar ist.

Textabbildungen

Methoden zur Textabbildung:

- Vector Space Model
- Tf-Idf gewichtetes Vector Space Model
- Simhash Hashverfahren
- Latent Semantic Indexing
- Paragraph Vectors basierend auf Worteinbettungen

Vector Space Model

Algorithmus:

1. Erstellung eines indexierten Wörterbuches anhand der im Korpus enthaltenen Worte.
2. Konstruktion eines Nullvektors für jedes Dokument, wobei die Dimensionsgröße der Länge des Wörterbuchs entspricht.
3. Für jedes Wort im Dokument wird der Vektor an der Indexposition inkrementiert, welche der Indexposition des Wörterbuches entspricht.

Beispiel:

I. Der Mann steht auf der Leiter.

II. Die Frau sitzt vor der Leiter.

1.

Index	1	2	3	4	5	6	7	8	9
Wort	der	mann	steht	auf	leiter	die	frau	sitzt	vor

3.

I	2	1	1	1	1	0	0	0	0
II	1	0	0	0	1	1	1	1	1

Tf-Idf Gewichtung

Probleme:

- Alle Worte besitzen die gleiche Gewichtung.
- Dokumente mit unterschiedlichen Inhalten können als ähnlich Vektoren abgebildet werden.

Term frequency – Inverse document frequency :

$term\ frequency$:	$tf(t,d) = f_{t,d}$	$t \in d$
$inverse\ document\ frequency$:	$idf(t,D) = log\frac{N}{n_t}$	$N = \lvert D \rvert$
	$tfidf(t,d,D) = f_{t,d} \cdot log\frac{N}{n_t}$	$d \in D, n_t = \lvert \{d \in D : t \in d\} \rvert$

Vorteil:

- Terme, die in vielen Dokumenten zu finden sind, bekommen eine niedrigere Gewichtung.
- Worte, die in wenigen Dokumenten auftauchen, bekommen eine höhere Gewichtung.

Ähnlichkeitsberechnung

Kosinus-Ähnlichkeit

$$\frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Paragraphenähnlichkeit:

- Alternativ zur Ähnlichkeitsberechnung auf Dokumentenebene, können die Dokumente in Absätze geteilt und diese Verglichen werden.
- Als Indiz für Textwiederverwendung zwischen zwei Dokumenten wird dann die höchste Kosinus-Ähnlichkeit zwischen zwei Paragraphen der Dokumente verwendet.

Vergleich der Retrieval-Methoden

Vergleich der Repräsentationsmethoden:

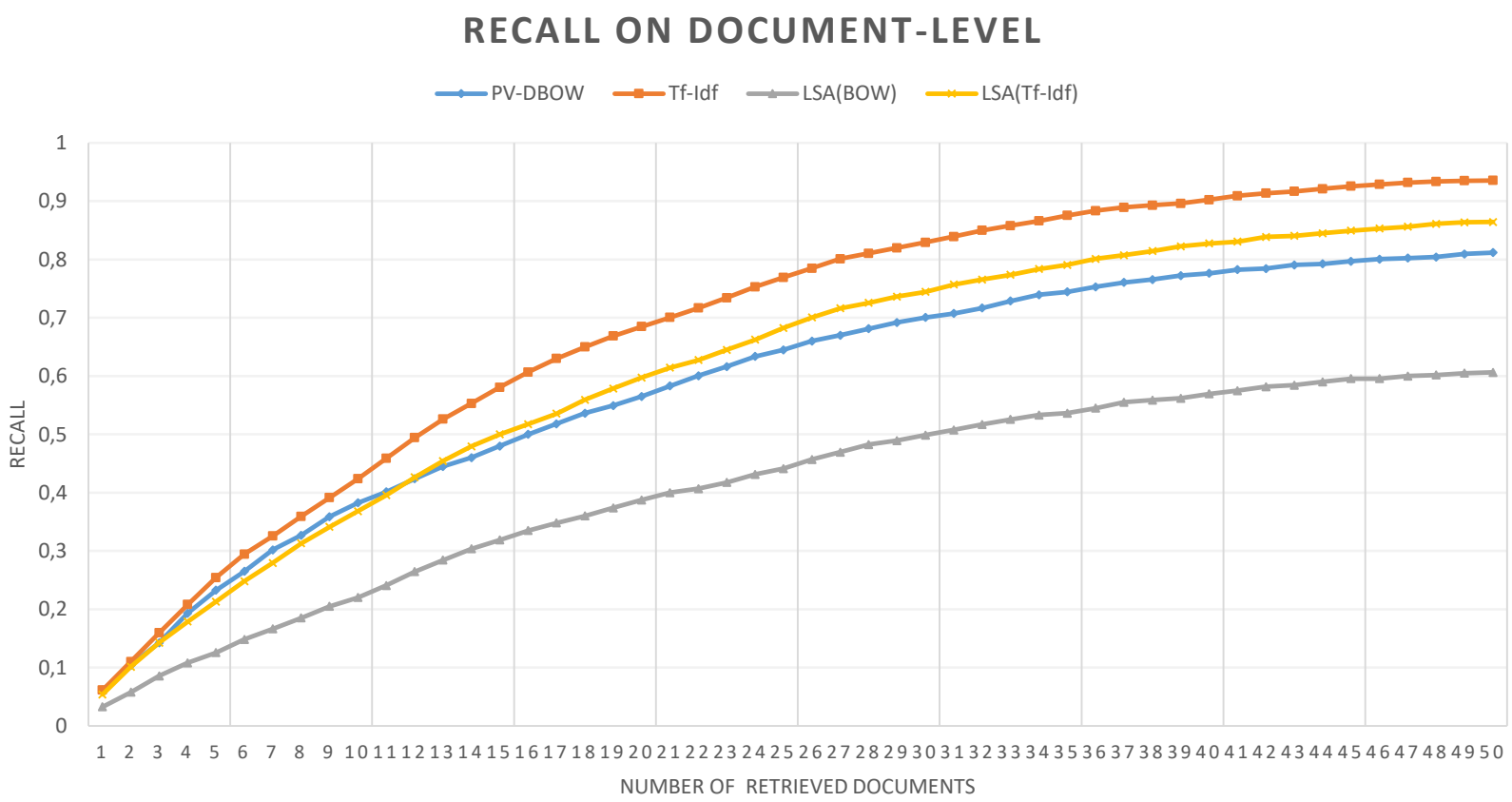
Gesucht wurde nun die Textabbildungs- und Vergleichsmethode, welche sich am besten eignet, zu einem Anfragedokument alle Dokumente eines Korpus mit gemeinsamen Textabsätzen zu liefern.

1. Alle Dokumente des Korpus mit allen Repräsentationsmethoden abbilden.
2. Zu jedem Anfragedokument die Ähnlichkeiten mit jedem anderen Dokument des Korpus berechnen.
3. Berechnung des Recalls aller Repräsentationen für die ersten 50 Ränge.

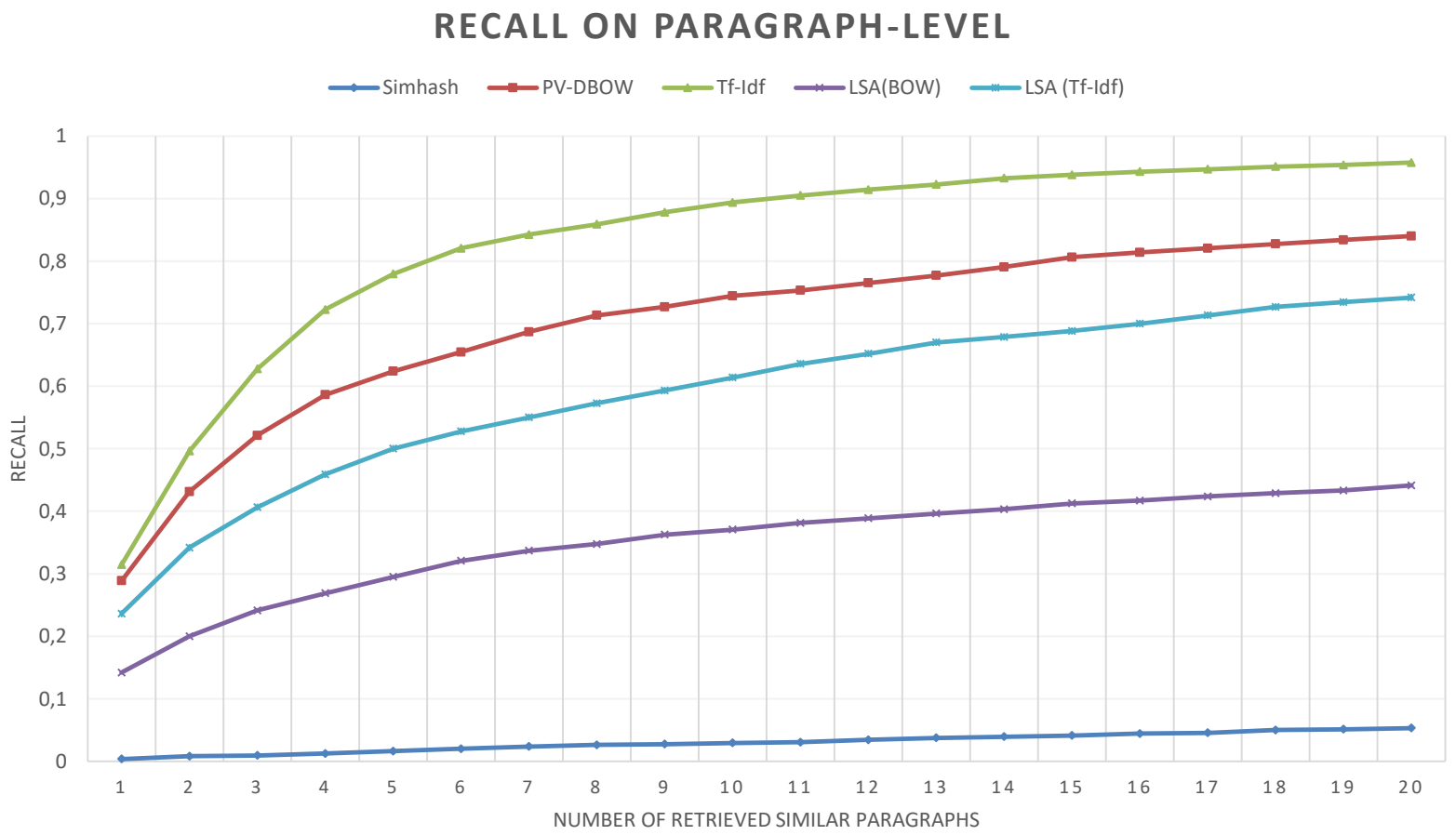
Hierzu wurde der PAN14 Plagiatskorpus genutzt:

- 3.385 Quelldokumente
- 179 Plagiatsdokumente, welche aus bis zu 16 Quelldokumenten plagiiert wurden.
- Kenntnis über alle Plagiatsfälle

Recall von Plagiaten des PAN14-Korpus



Recall von Plagiaten des PAN14-Korpus



Preprocessing

Preprocessing des Korpus

- Wikipedia XML-Dump vom 1.5.2016
- Extraktion aller Artikel des „Main namespace“
- Stoppwort Entfernung
- Filterung von Disambugierungsseiten und Artikeln mit weniger als 100 Worten
- Template-Erweiterung

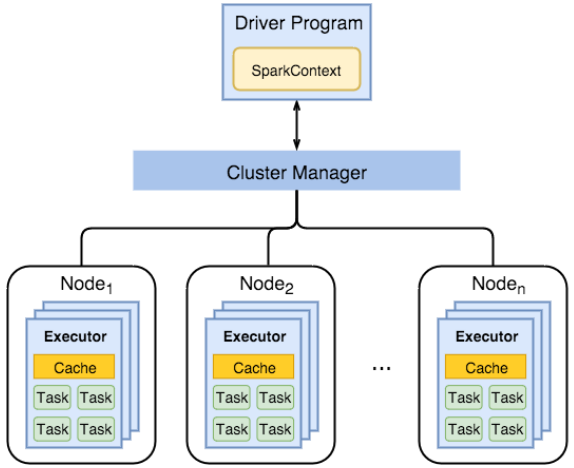
Artikel in Namespace 0	5.139.351
Artikel gefiltert durch Länge	2.400.125
Artikel mit Disambiguierung	118.468
Verbleibende Artikel	2.620.758
Anzahl der Paragraphen	8.507.799

Einteilung der Artikel in Paragraphen

- Unterteilung der Artikel in Paragraphen anhand von Absatzüberschriften
- Zusammenführung von Paragraphen mit weniger als 50 Worten
- 8.507.799 Paragraphen resultieren in 36 Billionen Operationen

Apache Spark

- High Performance Cluster Computing
- In-Memory Data Storage
- API‘s in Java, Scala, R, Python
- Lambda-Funktionen auf den Daten



https://http://datastrophic.io/core-concepts-architecture-and-internals-of-apache-spark/

Workspace

- Betaweb Cluster mit 100 Nodes
- 400 Executor mit 7GB RAM und 5 Executor-Kernen
- Webis17 mit 64 GB RAM als Spark Driver

Implementierung

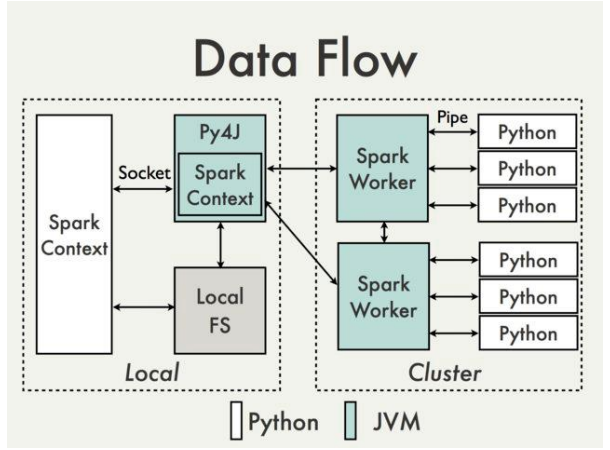
1. Preprocessed XML Dokumente als Objekte laden
2. Berechnung der Tf-Idf Vektoren mit der Spark Mllib
3. Gleichmäßige Verteilung der Vektoren auf allen Worker-Instanzen
4. Zweite Kopie der Vektoren partitionieren und einzeln zur Berechnung auf alle Worker kopieren

Apache Spark

Ergebnis und Optimierung

- 1% der Ähnlichkeiten nach 8 Stunden berechnet, was in 32 Tage für alle Dokumente enden würde
- Ersetzen der Mllib-Funktion durch Cython-Implementierung
- Verbesserung der Datenverteilung durch Modulo-Restklassen

→ Alle Ähnlichkeiten nach vier Tagen und 13 Stunden berechnet



https://cwiki.apache.org/confluence/display/SPARK/PySpark+Internals

Kosinus Ähnlichkeit	Anzahl der Dokumentenpaare
[0.9,1.0]	1.440.388
[0.8,0.9)	9.983.895
[0.7,0.8)	160.716.484
[0.6,0.7)	273.926.278
[0.5,0.6)	287.084.295
[0.4,0.5)	307.513.389
[0.3,0.4)	627.619.558
[0.2,0.3)	3.060.431.052
[0.1,0.2)	31.624.808.958

Ähnlichkeiten aller Wikipedia Artikelpaare anhand ihrer ähnlichsten Paragraphen

Textvergleich

Abgleich auf Wortebene

http://www.picapica.org/poster

- Die $36{,}2 * 10^{12}$ Dokumentenpaaren konnten durch Filterung aller Paare mit Kosinus < 0.5 auf $7{,}33 * 10^{8}$ reduziert werden.
- Für den Textvergleich der Paare wurde das PicaPica-Textalignment verwendet.
  → Aus 2 Texten alle ähnliche Passagen extrahieren
- Als Blackbox betrachtet: Durch die Eingabe der Texte und Parametern für Mindestlänge und Ähnlichkeitsschwellwert alle entsprechenden Textpassagen errechnen.

Parallelisierung

- Nutzung der Spark Java-API
- Verteilung der Kandidatenpaar-Indizes auf alle Worker
- Komprimierte Wikipedia-Artikel als Hashmap im Speicher jedes Nodes
- Rechenzeit: 11 Stunden und 22 Minuten auf 130 Nodes

Ergebnis

Gefundene Textwiederverwendung

- 4,25 Millionen Dokumentenpaare mit gleichen Textpassagen und 70% Übereinstimmung
- 210.479 beteiligte Artikel

Template-Cluster

- Artikel sortiert nach der Anzahl der gefundenen Textpassagen für einen Artikel
- Themenverwandte Seiten, die auf dem selben Template basieren, resultieren in vielen gefundenen Textwiederverwendungen.
- Beispiele: US School Districts, Städte in der Schweiz

Self-Reuse

- Dokumentenpaare sortiert nach den häufigsten Übereinstimmungen
- Exakte Textwiederverwendung in verschiedenen themenverwandten Artikeln durch den selben Autor
- Beispiele: War in Somalia – Operation Indian Ocean

Ergebnis

Text-Reuse

- Stichprobenartige Suche und größte addierte Passagenlänge
- Exakte Kopie ganzer Absätze, Paraphrasierung
- Autor müsste maschinell ermittelt werden
- Beispiele: Rock Music – American Rock, Islamic philosophy – Early Islamic philosophy

Source-Paraphrasing

- Stichprobenartige Suche
- Paraphrasierung einer externen Quelle
- Beispiel: Traditional Chinese medicine - Acupuncture

Zukünftige Arbeit

- Parameterverfeinerung des PicaPica-Textalignments
- Ein Großteil der Ergebnisse basiert auf Templates
  → Wikipedia ohne Template-Erweiterung erneut verarbeiten
- Performance zu gering für größere Datensätze wie ClueWeb
  → Code-Optimierung der Ähnlichkeitsberechnung und des Textabgleichs
  → Nutzung von aktuellen GPU

Fragen?