Eine Architektur zur Feature-Berechnung für die intrinsische Plagiatanalyse

Torsten Rausche

Einleitung und Motivation

- Plagiatproblematik
- Entwurf einer Softwarearchitektur für die Merkmalsberechnung in
- der intrinsischen Plagiatanalyse
- der Genreanalyse von Webseiten

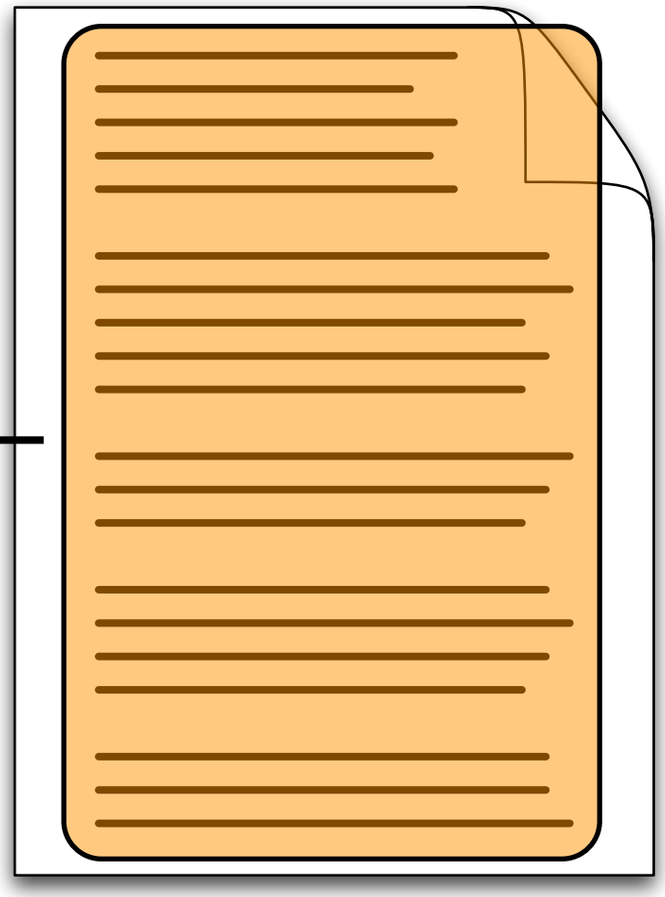
Intrinsische Plagiatanalyse

- keine Referenzdokumente nötig
- Ermittlung von Eigenschaften *(Features)* des Dokuments
- Erkennung von Passagen mit abweichenden Eigenschaften

Intrinsische Plagiatanalyse

n Features:

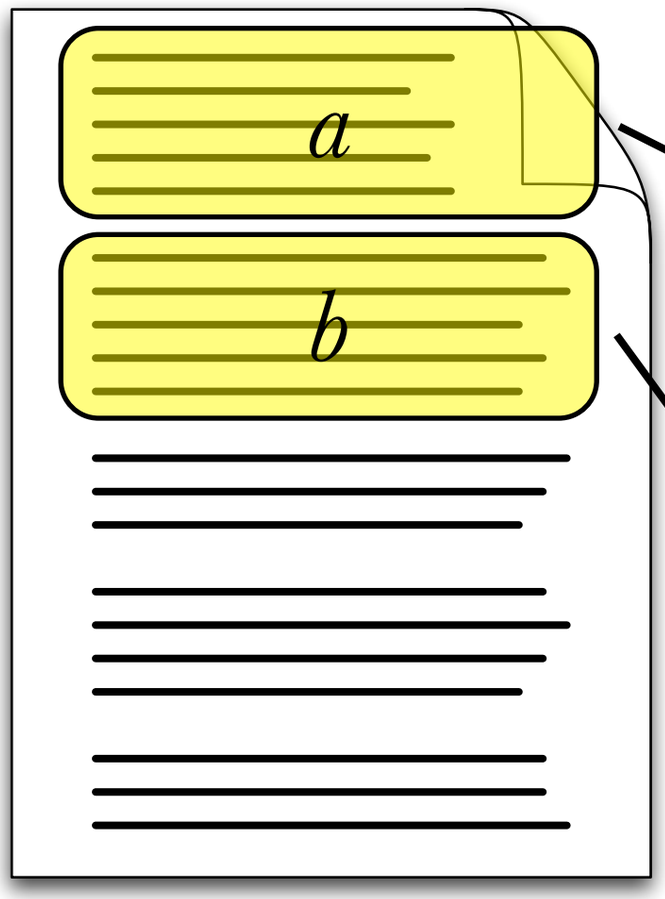
$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$



Intrinsische Plagiatanalyse

n Features:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

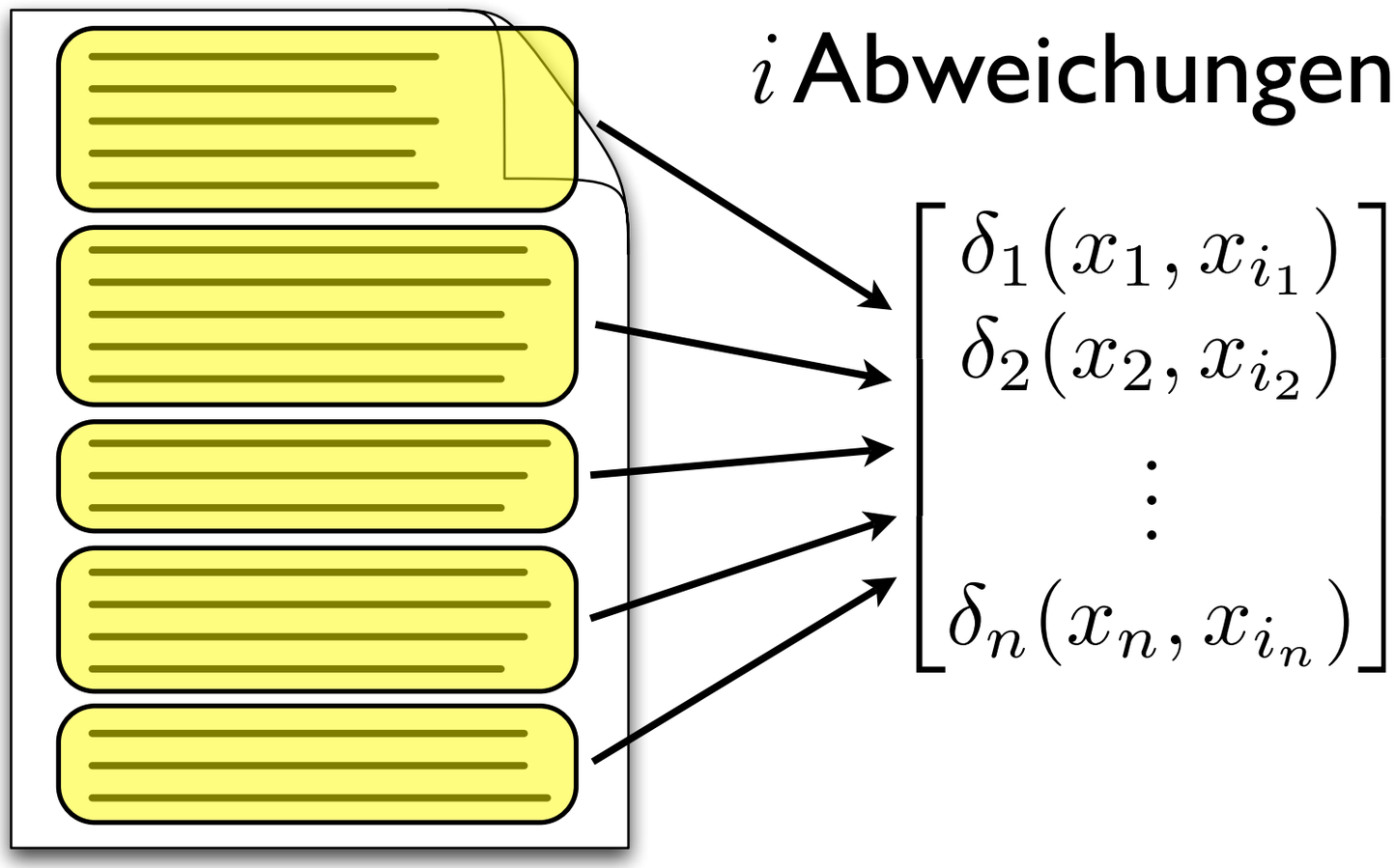


$i \bullet n$ Features:

$$\begin{bmatrix} x_{a_1} \\ x_{a_2} \\ \vdots \\ x_{a_n} \end{bmatrix} \quad \begin{bmatrix} x_{b_1} \\ x_{b_2} \\ \vdots \\ x_{b_n} \end{bmatrix}$$

i Abschnitte ($i \in \{a, b, \ldots\}$)

Intrinsische Plagiatanalyse



i Abweichungen pro Feature:

$$\begin{bmatrix} \delta_1(x_1, x_{i_1}) \\ \delta_2(x_2, x_{i_2}) \\ \vdots \\ \delta_n(x_n, x_{i_n}) \end{bmatrix}$$

Genreanalyse von Webseiten

- Ermittlung von Eigenschaften *(Features)* der Webseite
- Berechnung „on the fly“
- Einordnung in ein Genre

Reduzierung des Zeitaufwands

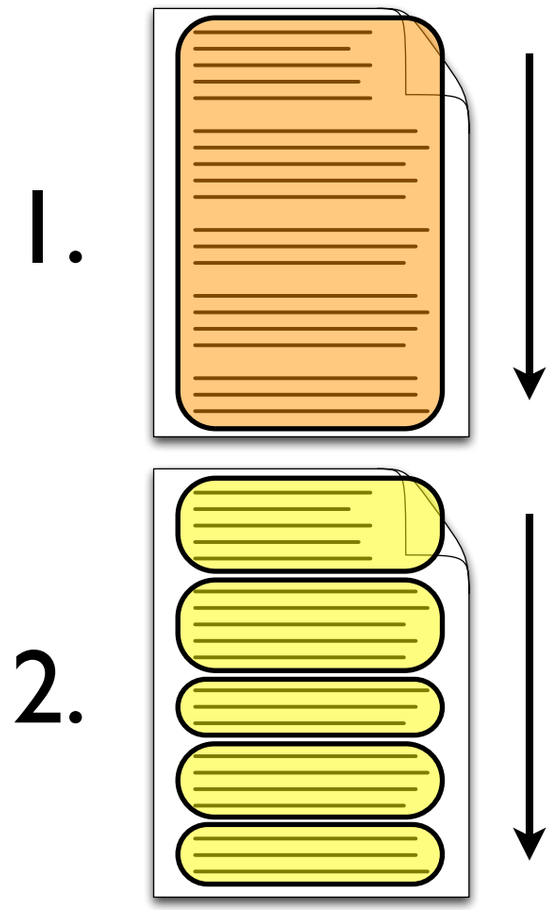
- intrinsische Plagiatanalyse erfordert Berechnung der Features für
  - das gesamte Dokument **und**
  - jeden einzelnen Absatz

Reduzierung des Zeitaufwands

- Feature-Berechnung in 2 Durchläufen:

1. Berechnungen für das gesamte Dokument

2. Berechnungen für die einzelnen Absätze



Reduzierung des Zeitaufwands

- Idee: Feature-Berechnung in einem Durchlauf
- Ein Parser durchläuft den Text einmal
- Sammeln der nötigen Informationen während des Durchlaufs
- Anstoß der Berechnungen, wenn genügend Informationen vorhanden sind

Analyse von Features

- Text-Features basieren auf:
  - einzelnen Zeichen
  - Wörtern
  - Sätzen
  - satzübergreifenden Textstücken (z. B. Absätze)

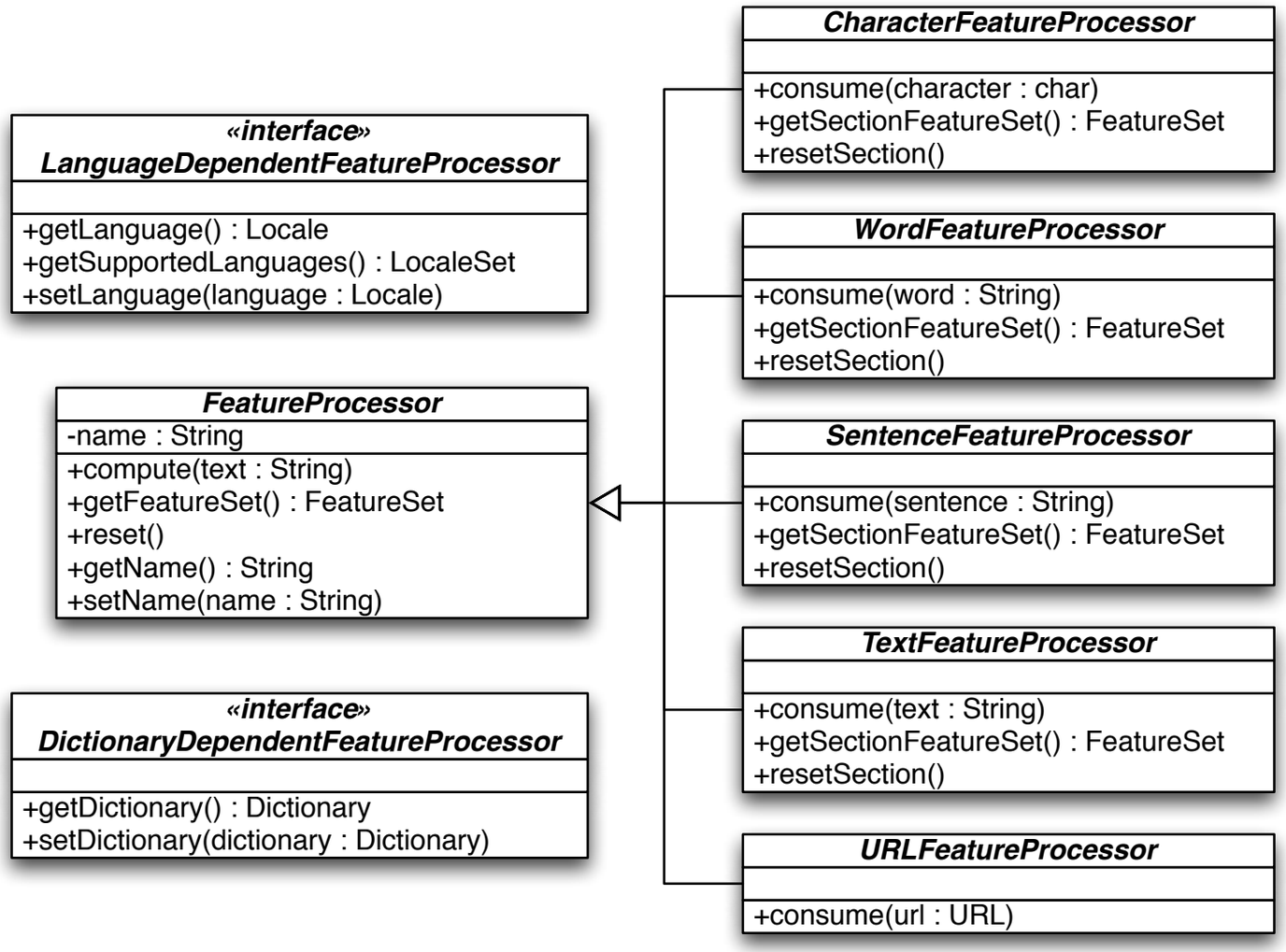
Analyse von Features

- Webseiten-Features basieren zusätzlich auf HTML-Tags:
  - Hyperlinks
  - Meta-Tags
  - eingebettete Objekte (Bilder, Flash, …)
  - Formulare, …

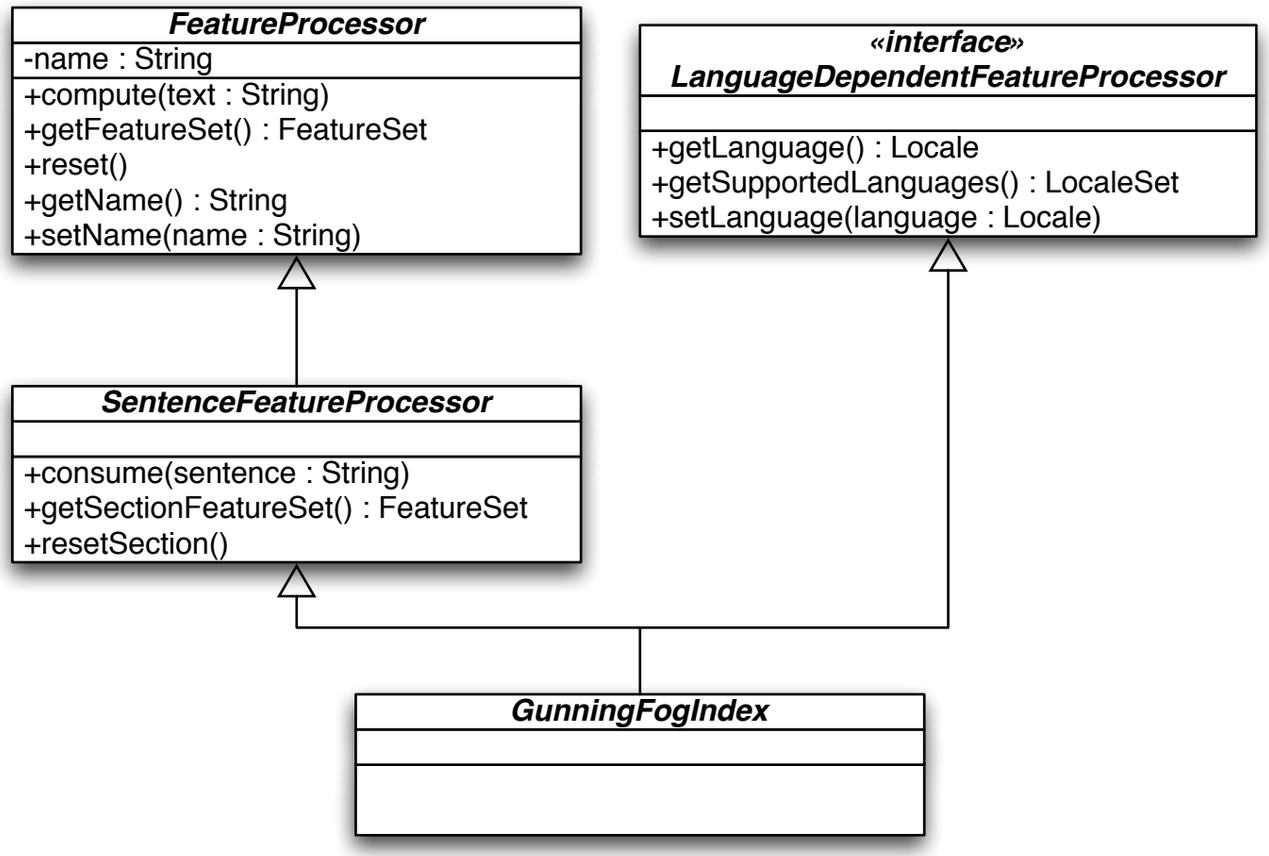
Spezielle Anforderungen

- Schnittstellen für
  - Sprachabhängigkeiten
  - Abhängigkeiten von Wörterbüchern
- Datentypen für Ergebnismengen

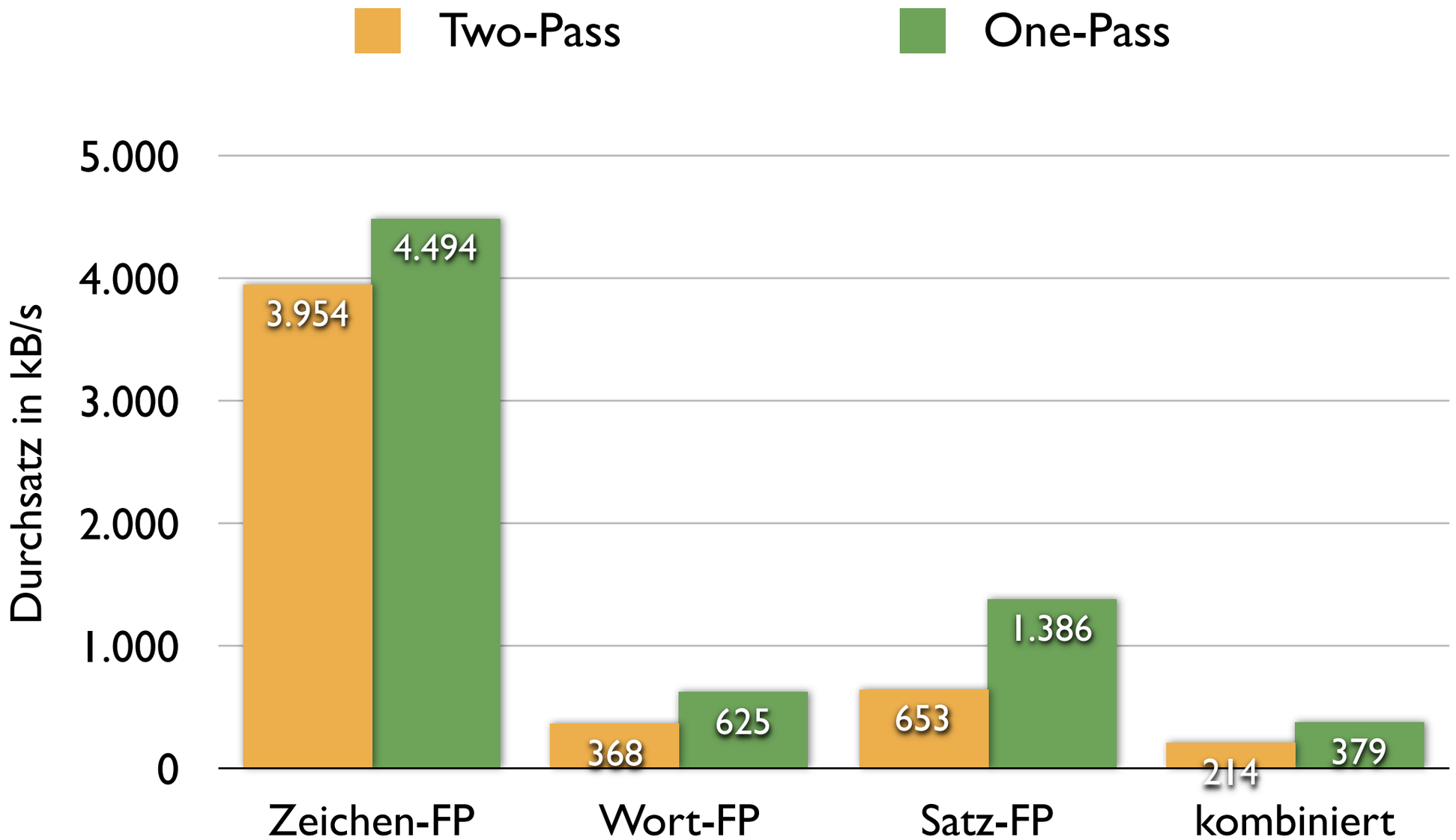
Umsetzung in Klassen



Umsetzung in Klassen



Benchmark-Ergebnisse



Benchmark-Ergebnisse

Feature-Prozessor-Menge	Verbesserung in %
Zeichen-Feature-Prozessoren	14
Wort-Feature-Prozessoren	70
Satz-Feature-Prozessoren	112
kombiniert	77

- One-Pass-Verfahren steigert die Geschwindigkeit

Zusammenfassung

- Entwurf:
  - Berücksichtigung unterschiedlicher Feature-Arten
  - Berücksichtigung spezieller Anforderungen
  - einheitlicher Datenaustausch
  - gute Erweiterbarkeit

Zusammenfassung

- Dokumentation in Javadoc
- Evaluierung:
  - Implementierung in Java
  - Benchmark
- Ergebnis: ermöglicht schnellere Berechnung

Vielen Dank für Ihre Aufmerksamkeit.

Fragen?