
HUMBOLDT-UNIVERSITÄT ZU BERLIN
HUMBOLDT-UNIVERSITÄT ZU BERLIN
Grammar-Based Fuzzing for Libre Office
Daniel Bucher · 14. August 2019
Photo CC BY Mark Turner

Inhalt

- Thematische Einführung
- Aufbau der Anwendung
- Präsentation der Ergebnisse

Thematische Einführung

Thematische Einführung

Definition: Grammatik

Eine Grammatik wird als 4-Tupel Definiert: $G = (V, \Sigma, P, S)$

- V – Menge der Variablen.
- Σ – Alphabet der Grammatik.
- P – Menge der Ableitungsregeln.
- S – Startsymbol der Grammatik.

Thematische Einführung

Beispiel: Grammatik

Sei G eine Grammatik $(V = \{A, B\}, \Sigma = \{w, x, y, z\}, P, S)$

$$
\begin{aligned}
P = \quad S &\rightarrow A \mid B \mid w \\
A &\rightarrow xB \mid y \\
B &\rightarrow yA \mid z
\end{aligned}
$$

Die Grammatik enthält unter anderem folgende Wörter:
xyxyxyxz, *yxyxyy* und *w*

Thematische Einführung

Chomsky-Hierarchie

1. G heißt Typ 3 oder regulär, wenn für alle Regeln P gilt: $P \subseteq V \times (\Sigma V \cup V \cup \epsilon)$
2. G heißt Typ 2 oder kontextfrei, falls für alle Regeln P gilt: $P \subseteq V \times (\Sigma \cup V)^*$
3. G heißt Typ 1 oder kontextsensitiv, falls für alle Regeln P gilt: $P \subseteq ((\Sigma \cup V)^* - \Sigma^*) \times (\Sigma \cup V)^*$ und die rechte Seite jeder Ableitungsregel in P zu gleich vielen oder weniger Zeichen ableitet.
4. Jede Grammatik G ist immer auch vom Typ 0 oder rekursiv aufzählbar.

Thematische Einführung

Fuzzing

- Teilbereich der Softwaretests
- Geeignet zum Generieren und Testen von vielen Eingaben
- Es kann eine hohe Pfadabdeckung erreicht werden
- Größerer Aufwand bei nicht trivialen Eingaben
- Unterteilbar in zwei Teilbereiche
  - Wir betrachten nur das *Blackbox-Fuzzing*

Thematische Einführung

Blackbox-Fuzzing

- Kein Wissen über den Programmcode
- Wenig Wissen über die Programmstruktur
- Weiter unterteilbar in:
  - *Mutationbased Blackbox-Fuzzing*
  - *Modelbased Blackbox-Fuzzing*

Thematische Einführung

Mutationbased Blackbox-Fuzzing

- Benötigt eine Grundmenge valider Eingaben
  - Diese wird auch als *Seeds* bezeichnet
- Anwendung von Mutationsoperatoren auf kopien der Seeds
- Daraus entstehen neue Eingaben

Thematische Einführung

Modelbased Blackbox-Fuzzing

- Es werden keine *Seeds* benötigt
- Verwendung eines Modells als Grundlage des Fuzzings
- Eingabedaten werden aus dem Modell generiert

Thematische Einführung

Probabilistische Informationen

- Auch als Auftrittswahrscheinlichkeiten bezeichnet
- Benötigt eine große Menge an validen Eingabedaten
- Auftrittswahrscheinlichkeiten werden am Modell anotiert
- Können das Blackbox-Fuzzing unterstützen

Thematische Einführung

OpenDocument Standard

- Entwickelt von Sun Microsystems und OASIS
- Format zur Speicherung von Bürodateien
  - Darunter Texte, Tabellen, Päsentationen etc.
- Seit 2011 in Version 1.2
- Besteht aus einem Zip-Dateiarchiv
- Beschreibende Grammatik liegt als *.pdf und als *.rng vor

Aufbau der Anwendung

Aufbau der Anwendung

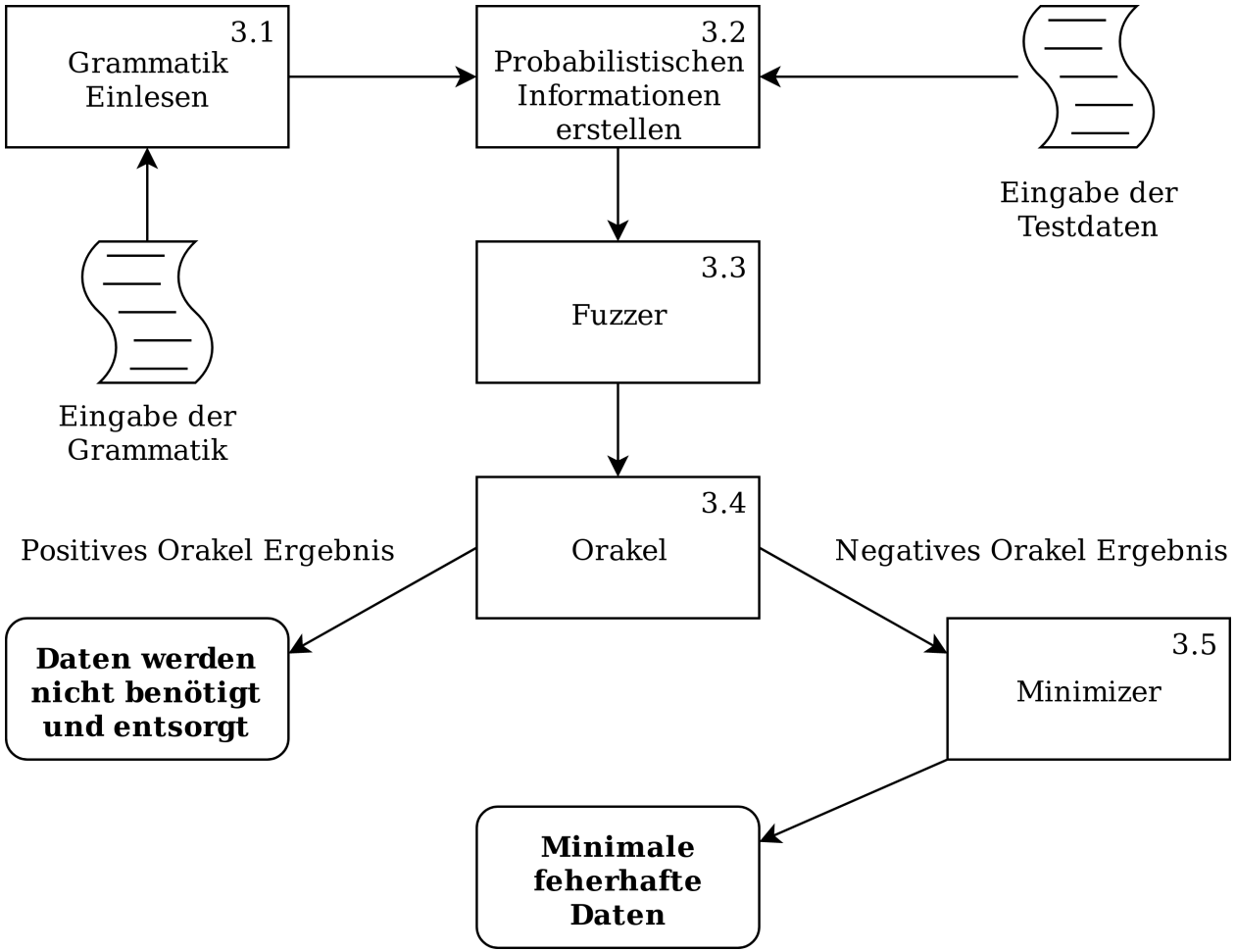
Ziel des Programms

- Grammatik basiertes erstellen von Eingabedaten
- Testen der Eingabedaten gegen *LibreOffice*
- Fehlerverursachende Dateien identifizieren
- Minimieren dieser Dateien zur isolation von Fehlern

- Fehler in *LibreOffice* finden

Aufbau der Anwendung

Schematischer Aufbau



Aufbau der Anwendung

Grammatik

- Vorhanden im Relax NG Format
  - Folgt dem XML-Format
- Frei Verfügbar in Version 1.2
- Grammatik vom Typ2 (*kontextfrei*)

Aufbau der Anwendung

Relax NG Metasymbole

- `<zeroOrMore>` – Sternhüllenoperator (`*` Operator)
- `<oneOrMore>` – Plushüllenoperator (`+` Operator)
- `<choice>` – Auswahloperator (`|` Operator)
- `<optional>` – Optionaloperator (`?` Operator)

Aufbau der Anwendung

Vorgehen: Grammatik einlesen

- Aufbau der Grammatik als XML Baum
- Navigation vom aktuellen Knoten aus
- Dazu wurde die *JDOM2* Bibliothek verwendet

Aufbau der Anwendung

Probabilistische Informationen erstellen

- Idee: Verbesserung der generierten Datein
  - Realitätsnähere Datensätze
- Sammeln von Auftrittwahrscheinlichkeiten
- Annotation dieser in der Grammatik

Aufbau der Anwendung

Vorgehen: Probabilistische Informationen erstellen

1. Analyse der validen Datei
2. Auftrittszahlen der Knoten sammeln
3. Annotation an den Knoten der Metasymbole
4. Weiter mit der nächsten Datei (Schritt 1)

Aufbau der Anwendung

Fuzzer

- Gewähltes Vorgehen: *Modelbased Blackbox-Fuzzing*
- Generierung von Eingabedaten anhand der Grammatik
- Entscheidungen basieren auf:
  - probabilistischen Daten
  - Pseudozufallszahlen

Aufbau der Anwendung

Problemstellung: Evaluation

- Wie werden die generierten Daten evaluiert?

Aufbau der Anwendung

Problemstellung: Evaluation

- Wie werden die generierten Daten evaluiert?

Lösung: Festlegen eines Orakels

- Auswahl einer Entscheidungsstrategie

Aufbau der Anwendung

Ansatz: *Crashtesting*

Vorgehen:

- Starten des Programms mit der zu testenden Datei
- Warten ob das Programm beim öffnen der Datei abstürzt
- Auswertung der Programmgrückgabe

Aufbau der Anwendung

Schwierigkeiten des *Crashtesting*

- Keine Schnittstelle für Programmtests

Aufbau der Anwendung

Schwierigkeiten des *Crashtesting*

- Keine Schnittstelle für Programmtests
  - Lösung: Verwendung der Linuxterminalschnittstelle

Aufbau der Anwendung

Schwierigkeiten des *Crashtesting*

- Keine Schnittstelle für Programmtests
  - Lösung: Verwendung der Linuxterminalschnittstelle
- Kein entscheidbares Ergebnis

Aufbau der Anwendung

Schwierigkeiten des *Crashtesting*

- Keine Schnittstelle für Programmtests
  - Lösung: Verwendung der Linuxterminalschnittstelle
- Kein entscheidbares Ergebnis
  - Lösung: Festlegen eines Timeouts

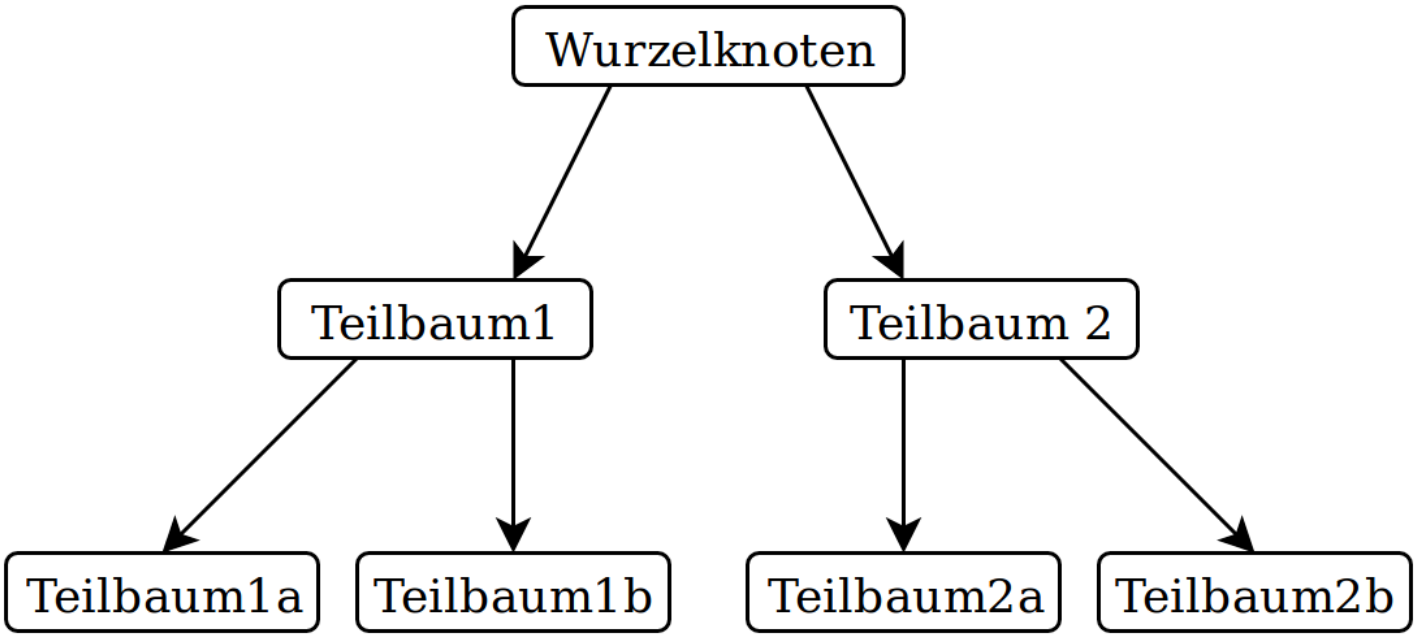
Aufbau der Anwendung

Minimizer

- Idee: Gefundene Fehler isolieren
- Vorgehen: Auffassen der Eingabedatei als Baum
- Ansatz:
  - Teilbäume entfernen
  - Erneut vom Orakel evaluieren lassen
  - Entscheiden, welchen einfluss der Teilbaum hat

Aufbau der Anwendung

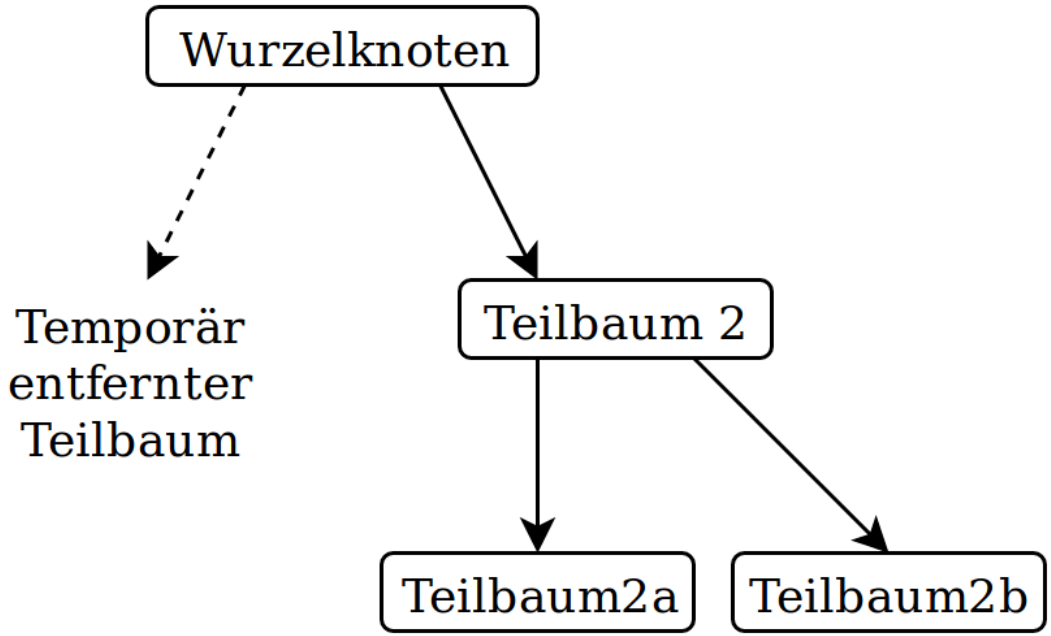
Beispiel: Minimizer



Der Fehlerverursachende Knoten lieg im Teilbaum 1b

Aufbau der Anwendung

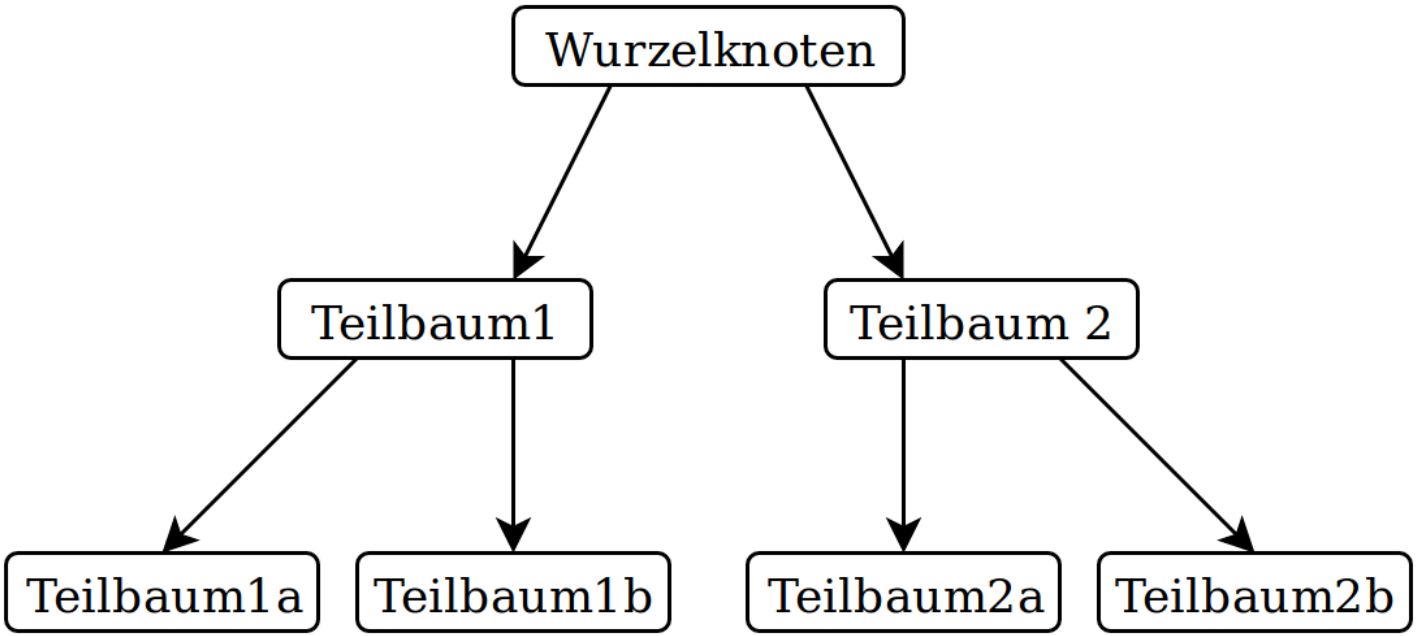
Beispiel: Minimizer



Der Fehlerverursachende Knoten lieg im Teilbaum 1b

Aufbau der Anwendung

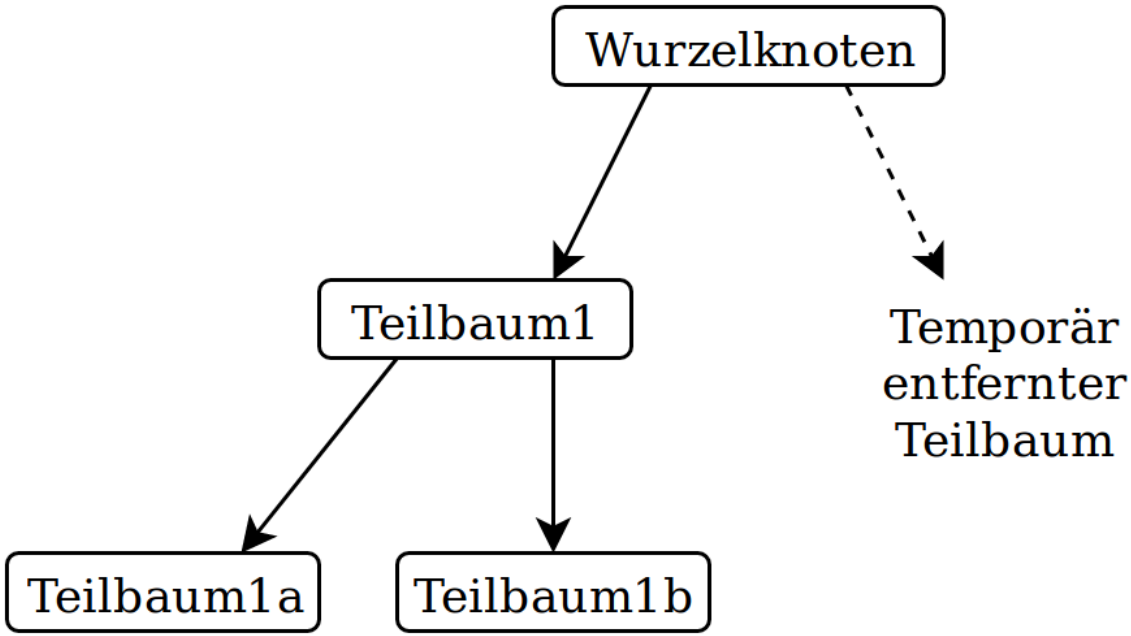
Beispiel: Minimizer



Der Fehlerverursachende Knoten lieg im Teilbaum 1b

Aufbau der Anwendung

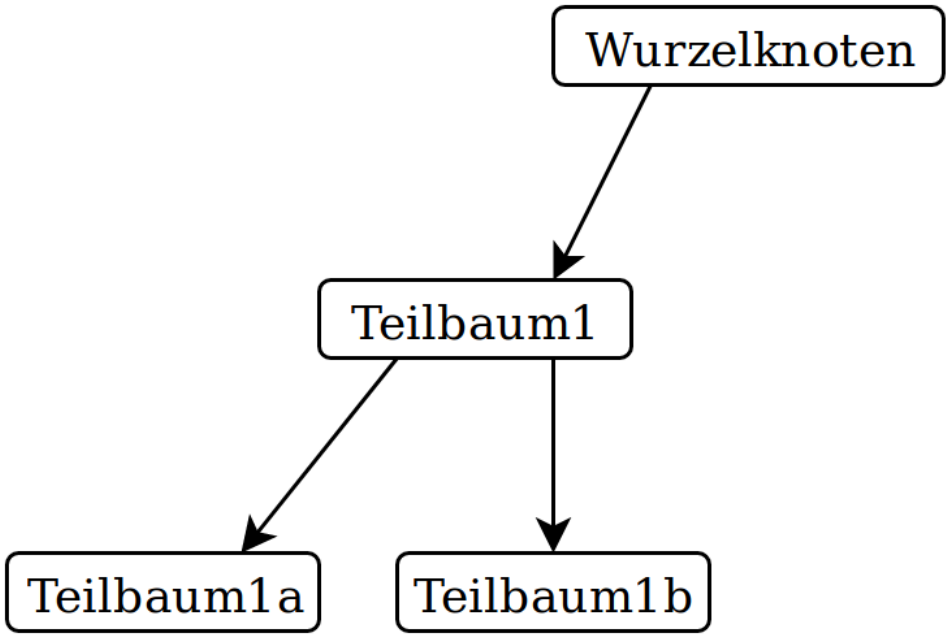
Beispiel: Minimizer



Der Fehlerverursachende Knoten lieg im Teilbaum 1b

Aufbau der Anwendung

Beispiel: Minimizer



Der Fehlerverursachende Knoten lieg im Teilbaum 1b

Präsentation der Ergebnisse

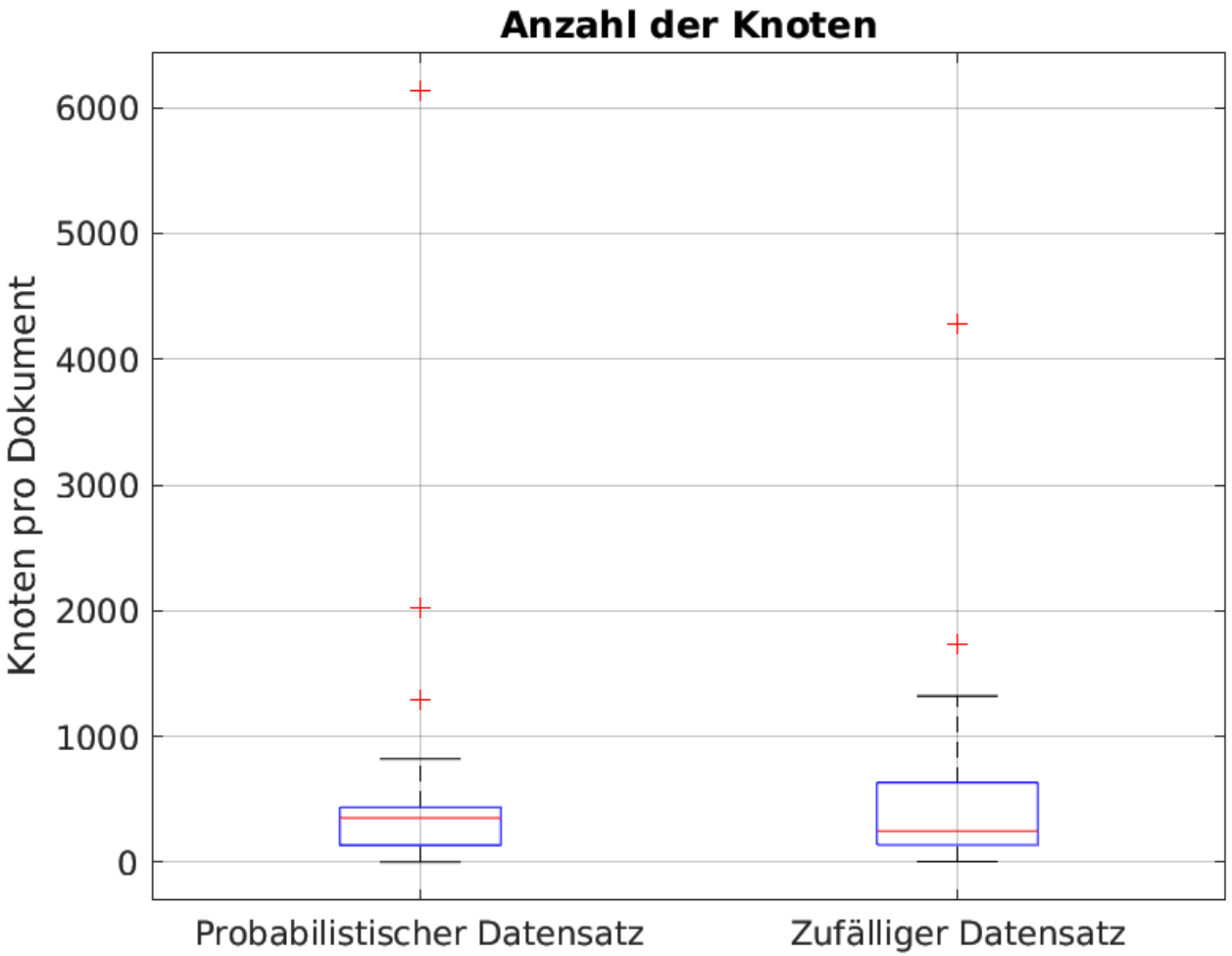
Präsentation der Ergebnisse

Aufbau des Experiments

- Verwendung von 2 Messreihen mit:
  1. Pseudozufallszahlen
  2. Probabilistischen Daten
- Analyse der Ergebnisse anhand von je 30 Sampels

Präsentation der Ergebnisse

Beobachtung: Knotenzahlen



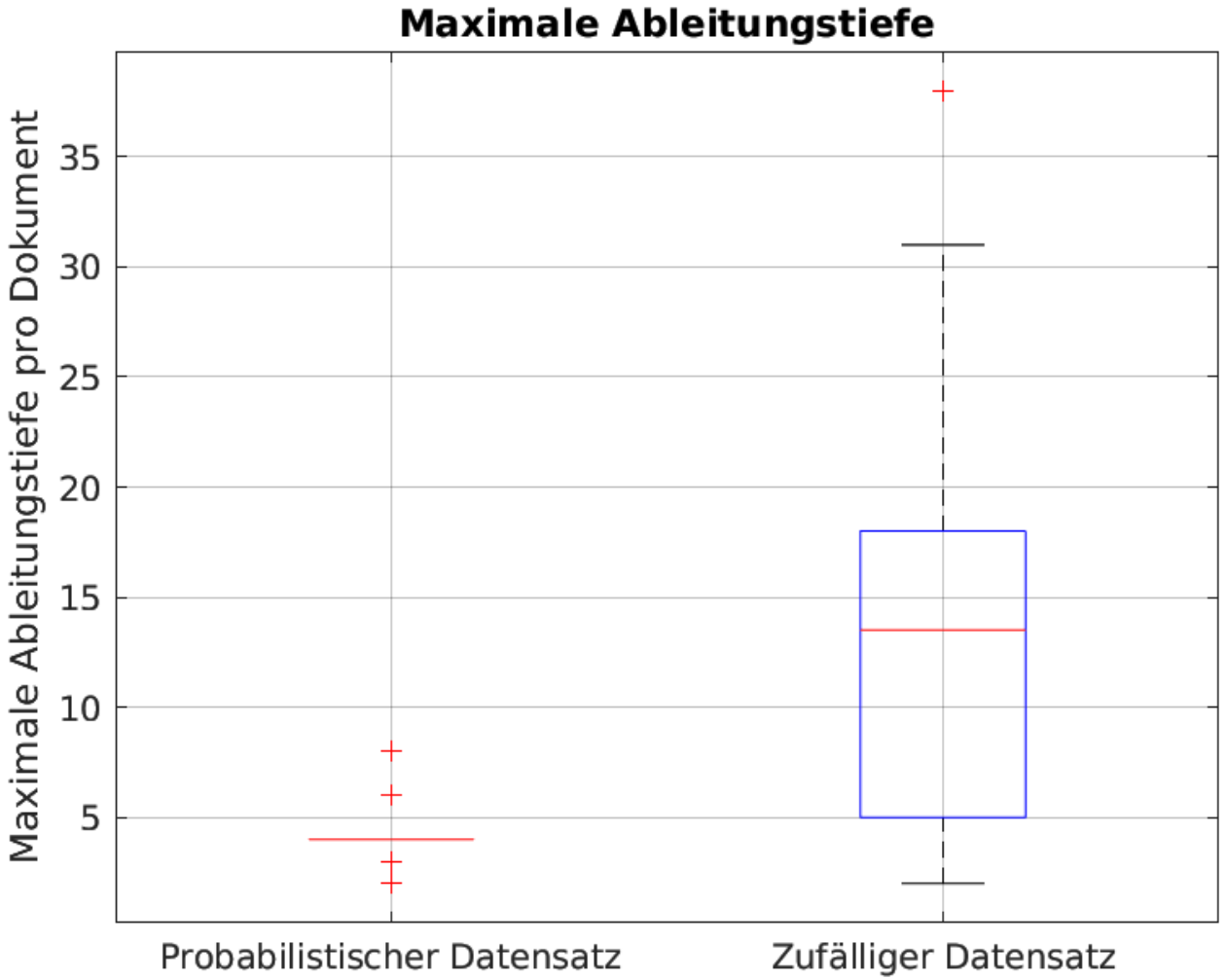
Präsentation der Ergebnisse

Begründung: Knotenzahlen

- Viele Schleifendurchläufe beim zufälligen Datensatz
- Beschränkung durch probabilistische Daten

Präsentation der Ergebnisse

Beobachtung: Ableitungstiefe



Präsentation der Ergebnisse

Begründung: Ableitungstiefe

- Zyklische Ableitungsmöglichkeiten
- Beschränkung durch probabilistische Daten

Quellen

Alle Inhalte dieser Präsentation entstammen meiner Abschlussarbeit:
https://scm.cms.hu-berlin.de/bucherda/ba-thesis

Vielen Dank für die Aufmerksamkeit

Gibt es Fragen?