

# AaS – Annotation als Suchproblem

Rebekka Hubert

Michael Staniek

Simon Will

6. Dezember 2016

## 1 Einführung

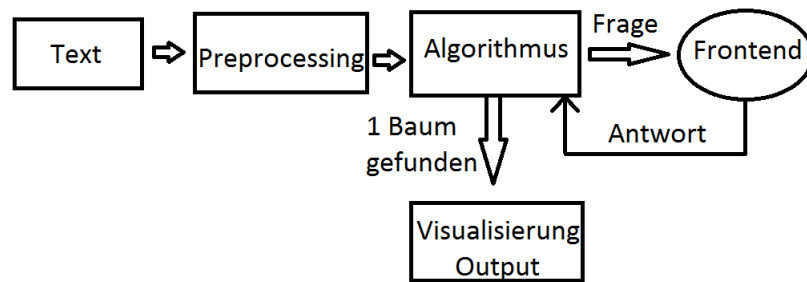
In diesem Dokument werden die Funktion, die Funktionsweise und die Systemvoraussetzungen der Software „Annotation als Suchproblem“ (AaS) beschrieben.

### 1.1 Motivation

Bei AaS handelt es sich um eine Software zur Unterstützung von Annotatoren, die auf Basis eines Parseforest Fragen an den Annotator stellt und anhand dieser den optimalen Parsebaum auswählt. Dafür benötigt der Annotator zwingend Grunderfahrung im Annotieren, aber keinerlei Programmierkenntnisse. Dies beschleunigt den Vorgang der menschlicher Dependenzannotation, die weiterhin die beste Qualität aufweist.

## 2 System

Um die größtmögliche Kompatibilität und Flexibilität zu erreichen, besteht das System aus formal drei Teilbereichen, die durch Schnittstellen kommunizieren. Dies ermöglicht es, unabhängig von der Kernanwendung im Server verschiedene Clients zu entwickeln, die auch als Plugin bestehender Anwendungen realisiert werden können. Konkret handelt es sich um eine Server-Client-Konstruktion, die es dem User ermöglicht, sowohl das gesamte System als auch lediglich Teilbereiche zu installieren: So ist das Preprocessing optional und auch der Client kann für einen bereits installierten Server aufgesetzt werden. Im Folgenden ein Überblick über das gesamte System:



Das System besteht bekanntlich aus drei Teilbereichen, die durch Schnittstellen kommunizieren:

- Preprocessing  
Generieren der möglichen Parsebäume und Übergabe der  $k$  besten an die Fragen-generierung über die Schnittstelle Preprocessing – Fragegenerierung
- Algorithmus zur Fragen-generierung auf einem Server  
erhält die  $k$  besten Parsebäume
- Client  
Übermittlung der Fragen an den Annotator und der Antwort des Annotators an den Algorithmus über die Schnittstelle Client – Server sowie vom Prozess unab-hängige Darstellung des Parsebaums

## 2.1 AaS-Server

Der AaS-Server generiert die Fragen, nach welchen der optimale Parsebaum ausgewählt wird. Näheres im AaSP.

### 2.1.1 Abhängigkeiten

Der Server benötigt folgende Software um korrekt zu laufen:

- Python  $\geq 3.4$
- Python3 packages: asyncio, Flask
- Parser, der die  $k$  besten Parses für einen Satz liefert (z.B. Anders Bjorkelunds übergangsbasierter Parser)
- TCP- oder UNIX-Sockets
- JSON

### 2.1.2 Algorithmus

Der Algorithmus folgt dabei folgendem Schema:

- Generiere aus Parseforest die Frage, welche bei ihrer Beantwortung die meisten Bäume rausfiltert.
  - Suche des Tupels, das den Suchraum am ehesten halbiert:  
 $a \leftarrow \text{len}(\text{pares})$  Anzahl der Parse-Bäume.  
For each tuple  
     $\text{score} \leftarrow \text{abs}(\text{count}(\text{tuple}) - \frac{a}{2})$   
End For
- Nimm das Tupel als Frage

## 2.2 AaS-CLI-Client

Der AaS-CLI-Client ist ein textbasierter Client. Die Nachrichten des Servers werden vollständig aber visuell unaufwendig dargestellt. Nichtsdestoweniger implementiert der CLI-Client das AaS-Protokoll vollständig und unterstützt damit auch die Operationen *undo* und *abort*. Der Benutzer interagiert mit dem System durch einfache Befehle wie „y“ oder „yes“ für die bejahende Benantwortung einer Frage oder „undo 2“, um die letzten beiden Antworten zu widerrufen. Nach Abschluss der Annotation eines Satzes besteht die Möglichkeit den entstandenen Baum noch mithilfe eines externen Texteditors zu bearbeiten.

### 2.2.1 Abhängigkeiten

Der CLI-Client hat folgende Abhängigkeiten:

- Python  $\geq 3.4$
- TCP- oder UNIX-Sockets
- JSON
- Externer textbasierter Texteditor

## 2.3 AaS-Web-Client

*Anmerkung: Der AaS-Web-Client befindet sich zu diesem Zeitpunkt noch im frühen Entwicklungsstadium, folglich besteht keine Garantie, dass alle hier getroffen Aussagen fristgerecht umgesetzt werden können. Dementsprechend ist der folgende Abschnitt unterteilt: Zuerst werden die Features des Client aufgeführt, die in der finalen Version zur Verfügung stehen werden. Danach folgen jene Features, die je nach vorhandenen zeitlichen Rahmen noch folgen werden.*

Der AaS-Web-Client ist eine interaktive GUI und handhabt die Kommunikation zwischen Benutzer und Software. Über den Client werden sowohl die Benutzereingaben

entgegengenommen als auch die Antworten des Servers dem Benutzer zugeführt. Dabei werden dem Benutzer durch den Client mehrere Funktionen zur Verfügung gestellt, die die Annotationsaufgabe erleichtern:

Zuerst einmal werden über den Client die Fragen des Algorithmus in menschenlesbarer Form ausgegeben und können vom Benutzer mit „ja“ oder „nein“ beantwortet werden. Sobald der optimale Parsebaum ausgewählt wurde, wird dieser über den Web-Client ausgegeben. Dieser hat dann die Möglichkeit diesen abzuspeichern oder komplett zu verwerfen.

Alle weiteren Features fallen in den optionalen Bereich. Während der Annotation werden die bereits feststehenden Kanten angezeigt. Nachdem der optimale Baum ausgewählt wurde, erhält der Benutzer die Option diesen abzuändern. Des Weiteren sind noch weitere Benutzeraktionen geplant, *undo* und *abort*. Wird *undo* ausgeführt, wird die letzte Benutzeraktion rückgängig gemacht. Eine bereits beantwortete Frage kann somit ein weiteres Mal beantwortet und ein etwaiger Fehler behoben werden. *abort* beendet den Hilfsvorgang und erfragt vom Server den besten Baum oder den Teilbaum, dessen Kanten feststehen. Bevor der Benutzer den Baum speichert, kann er noch Anpassungen am Baum vornehmen.

### 2.3.1 Abhängigkeiten

Der Web-Client hat bis auf Weiteres folgende Abhängigkeit:

- Python  $\geq 3.4$
- Python-Paket `Flask`
- TCP- oder UNIX-Sockets
- JSON

## 3 Evaluation

Die Evaluation erfolgt automatisch, da dadurch sowohl in der gleichen Zeitspanne mehr evaluiert werden kann als es von Hand möglich wäre als auch menschliche Fehler vermieden werden. Der Goldparse eines Satzes wird hierbei als Ersatz für einen menschlichen Annotator benutzt. Jegliche Fragen des Servers werden an diesen Goldparse gestellt. Sobald der Algorithmus den optimalen Parse gewählt hat, wird dieser mit dem Goldparse verglichen.

### 3.1 Evaluationsmaße

Die Evaluation erfolgt über die Anzahl der korrekt gelabelten Kanten. Zur Bewertung der Ergebnisse werden insgesamt drei Evaluationsmaße benutzt:

- Minimum Edit Distance  
Anzahl der nicht übereinstimmenden, gelabelten Kanten

- Labeled Attachment Score  
Anzahl der gelabelten Kanten, die mit dem Goldstandard übereinstimmen
- Error Counter  
Anzahl der nicht gefundenen Bäume (die Antworten auf die Fragen haben jede möglich Baum ausgeschlossen)