AaS - Annotation als Suche

Rebekka Hubert Michael Staniek Simon Will

February 01, 2017

Übersicht

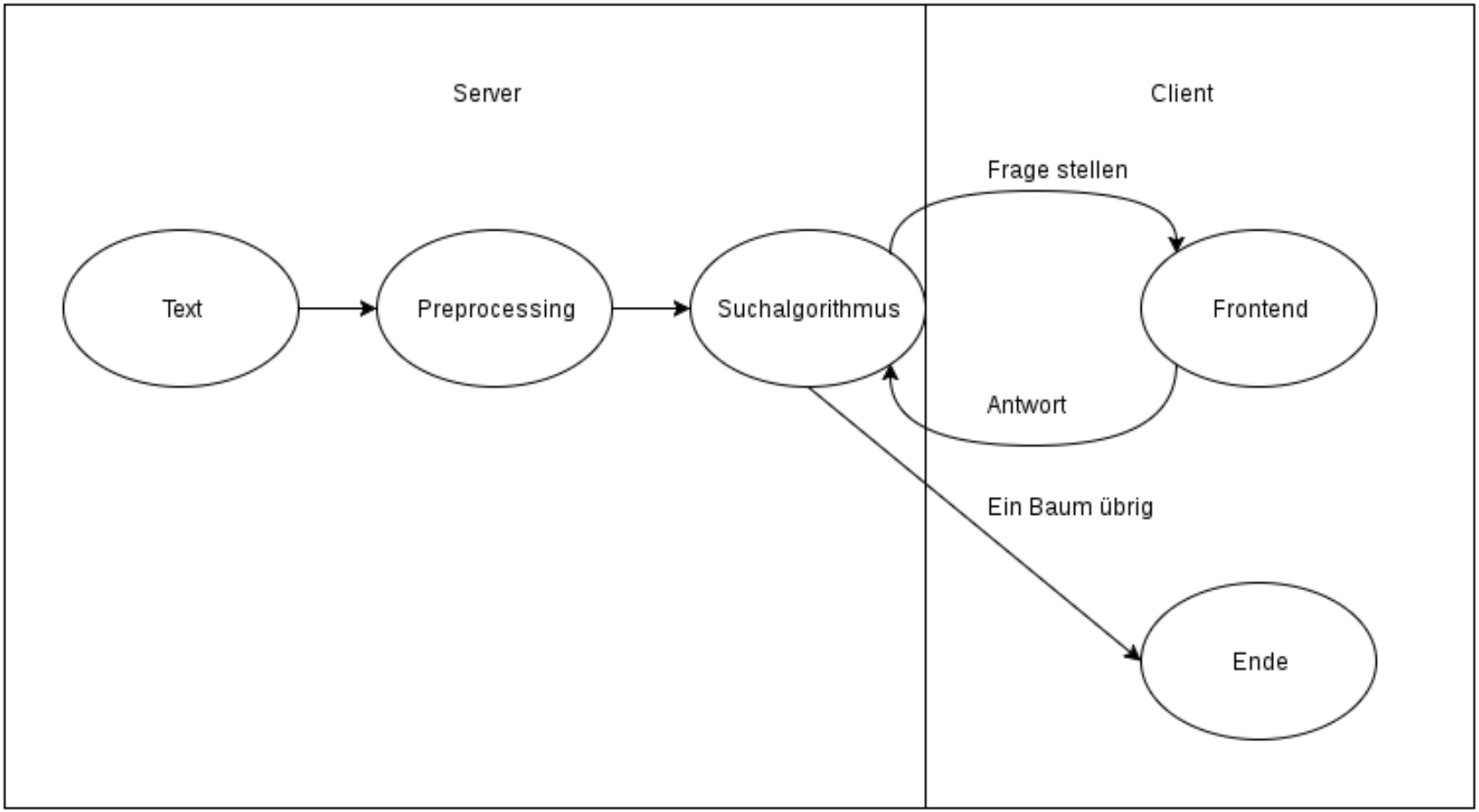
Motivation

- Menschliche Annotation
  - dauert und kostet
  - bei Dependenzparses immer noch ungeschlagen.
- Wir suchen eine Möglichkeit, Annotatoren zu unterstützen
  - Basierend auf vielen möglichen Parses für einen Satz Fragen an den Annotator stellen
  - anhand der Fragen zum optimalen Parsebaum gelangen

Für wen?

- Das Programm soll für Annotatoren gedacht sein.
  - Diese Userklasse hat nicht unbedingt viel Programmiererfahrung.
- Eine gewisse Grunderfahrung im Annotieren wird vorausgesetzt.

Übersicht


Server
Client
Frage stellen
Text
Preprocessing
Suchalgorithmus
Frontend
Antwort
Ein Baum übrig
Ende

Algorithmus

- Generiere aus Parseforest die Frage, welche bei ihrer Beantwortung die meisten Bäume rausfiltert.
  - Suche des Tupels, das den Suchraum am ehesten halbiert:

a $\leftarrow$ len(parses) ▷ Anzahl der Parse-Bäume.
for each tuple **do**
    score $\leftarrow \text{abs}(\text{count}(\text{tuple}) - \frac{a}{2})$
end for

- Nimm den Tupel als Frage

Abhängigkeiten

- Python 3.4 oder neuer (aufgrund des Pakets `asyncio`)
- Parser, der k-best Parses liefert
- TCP- oder UNIX-Sockets
- JSON
- wenn GUI verfügbar sein soll: Flask, graphviz

Requirements

Um die Anpassungsfähigkeit des Systems nicht von Anfang an einzuschränken, bietet es sich an, das System in drei Module aufzuteilen:

- Preprocessing:
  - Generierung der möglichen Parsebäume unter Betrachtung und Abänderung verschiedener Parser
  - Übergabe der k-besten Parsebäume an das System zur Fragegenerierung
- Fragengenerierung
  - System zur Fragegenerierung mithilfe eines Algorithmus (oder mehreren) erhält über Preprocessing-Schnittstelle die Parsebäume

Requirements

- Benutzerschnittstelle:
  - Übermittlung der Fragen an den Annotator
  - Übergabe der Antwort des Annotators an den Algorithmus
  - *Darstellung* des Parsebaums ist unabhängig zu diesem Prozess

Was haben wir geschafft

Zeit für eine Demo

Evaluation

- Automatische Evaluation
  - Methode:
    - der Goldparse wird als Annotator benutzt
    - die Fragen werden an den Goldpars gestellt
    - überprüfen: Entspricht Ergebnis dem Goldpars?
  - Vorteile
    - Überprüfen mehrerer Ergebnisse in derselben Zeit
    - Vermeiden menschlicher Fehler

Resultate der automatischen Evaluation

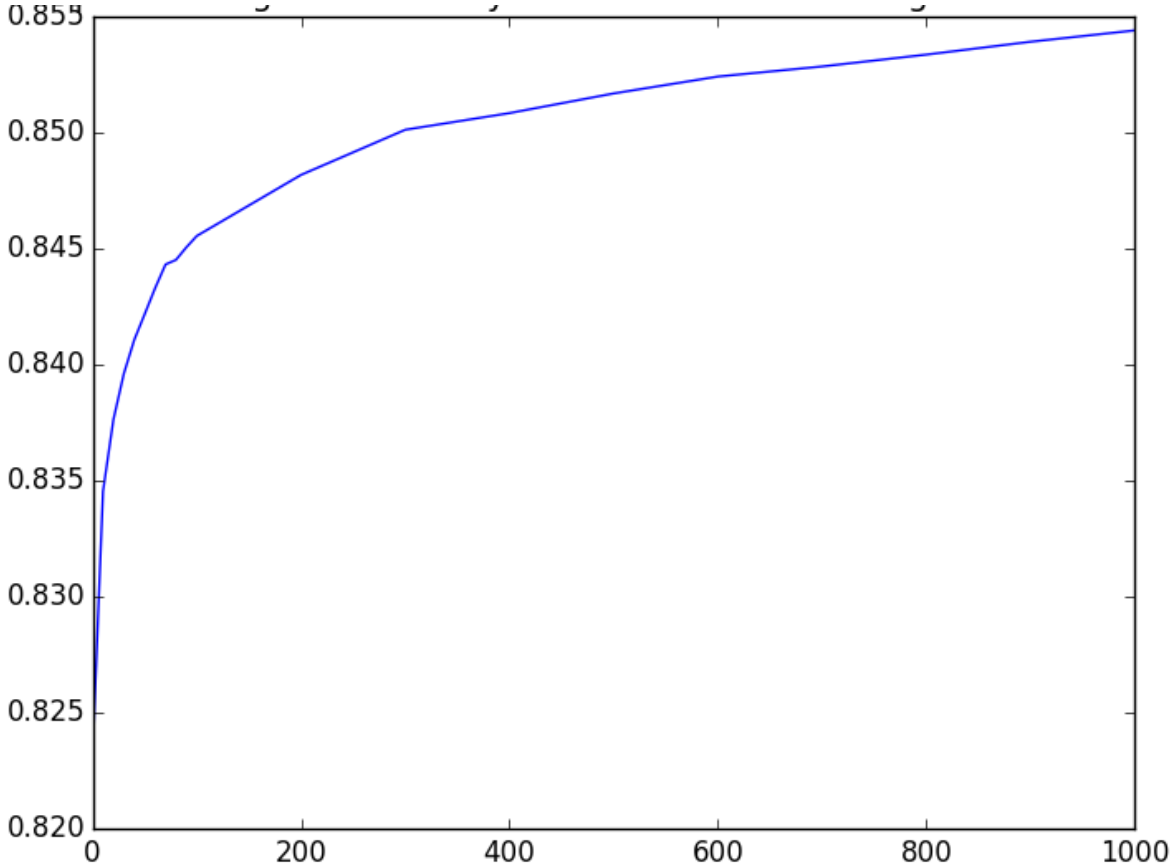


Figure: Labeled Attachment Score gegeben ein Forest der Größe X.

Resultate der automatischen Evaluation

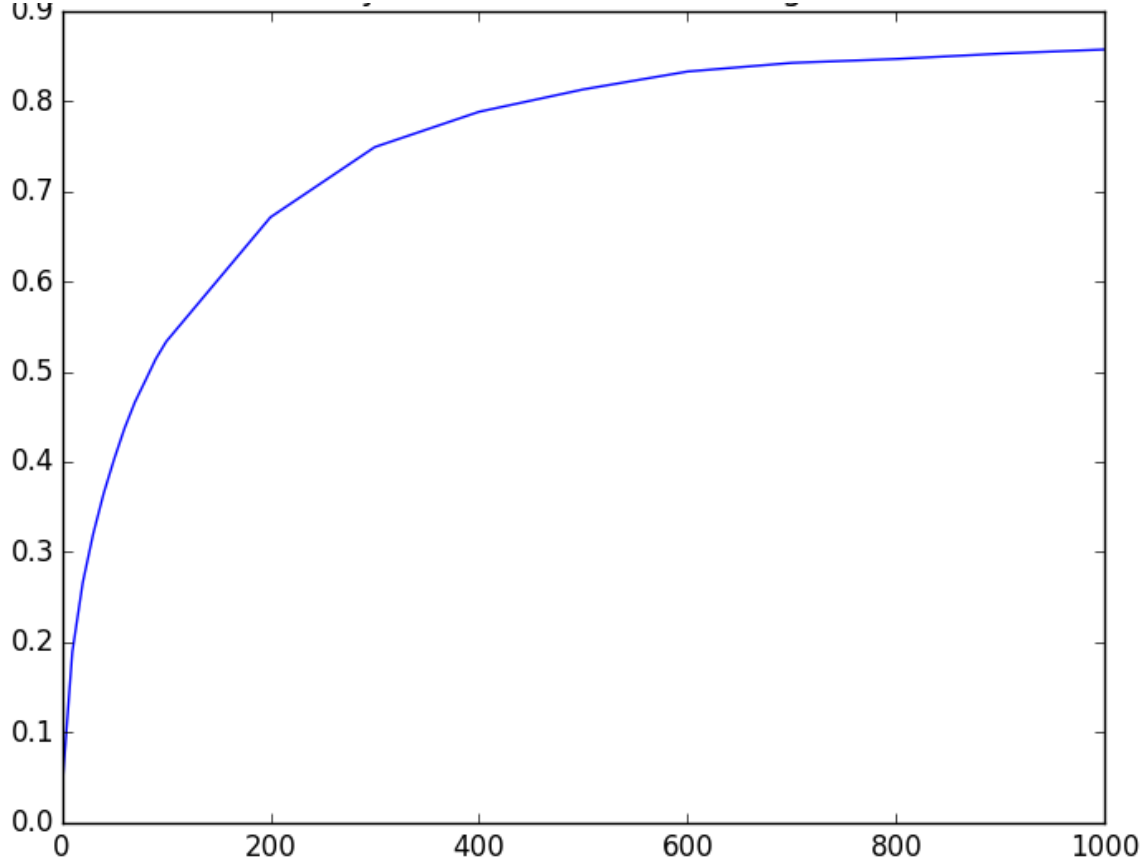


Figure: Durchschnittliche Anzahl an Fragen gegeben ein Forest der Größe X.

Resultate der automatischen Evaluation

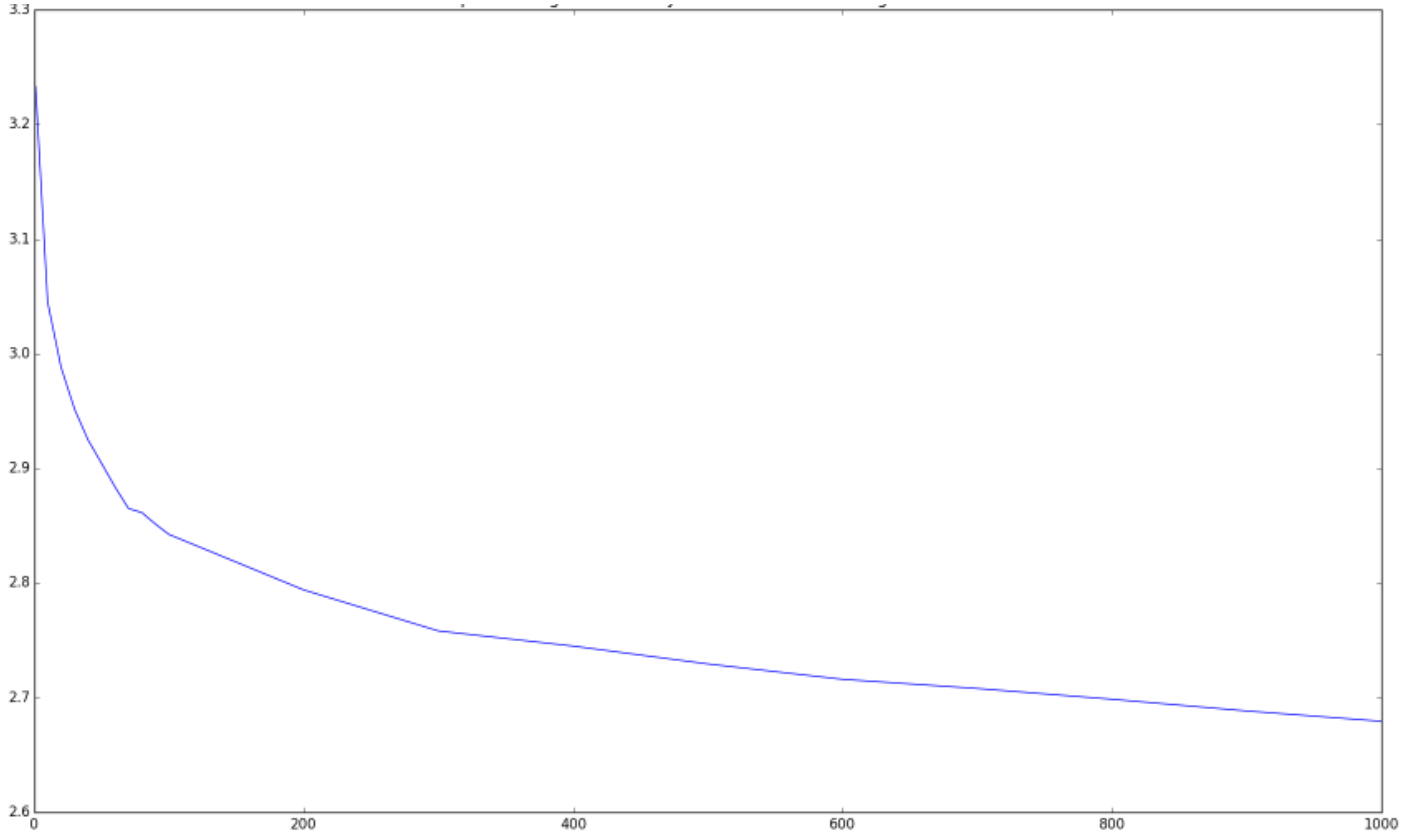


Figure: Durchschnittliche Minimum Edit Distance gegeben ein Forest der Größe X.

Fazit

- Wir haben ein Framework geschrieben, welches hilft Annotation als Suche in einem großen Suchraum zu anzuwenden.
- In unseren Tests haben wir auch bei hohen Datenmengen eine schnelle Antwort des Algorithmus.
- Sehr viel, was ursprünglich eher im Ausblick stand als im Skopus des Projektes (GUI, Undo-Befehl, ...) konnten wir in der Zeit gut implementieren.

Noch bestehende Bugs

- Vom Server beim Parsen erstellte Dateien werden nicht automatisch gelöscht.
- Aliases für Formate sind noch experimentell.

Ausblick

- Weitere Algorithmen zur Fragenauswahl implementieren
- Andere Leute den Annotationsprozess ausprobieren lassen, um ihn noch effizienter zu gestalten.
- Einbauen, dass Fragen nicht nur basierend auf (Dependent, Head, Relation) Tripeln gestellt werden. Wenn unterschiedliche POS-Tags vorhanden sind, kann der Suchraum möglicherweise einfacher eingeschränkt werden, wenn man Fragen bezüglich der POS-Tags stellt.

Was haben wir gelernt?

- Wie man eine Server-Client Struktur in einem Programm aufbaut, und wie man es nicht tut.
- Vorher darauf achten, dass alle Komponenten die wir aussuchen miteinader kompatibel sind, damit es später zu keinen Problemen kommt.
- Kommunikation in einem Projekt ist sehr wichtig; je größer das Projekt ist, desto wichtiger.