Entwurfsbeschreibung

Inhaltsverzeichnis

1 Allgemein

Für das Hauptprojekt wird ein Goldstandard von Sätzen und zugehörigen Annotationen benötigt, mit denen eine Evaluation durchgeführt werden kann. Diese Softwarestudie hat es zum Ziel eine einfache und effiziente Web-Applikation zu entwickeln, mit der der Goldstandard erstellt wird.

2 Produktübersicht

Zum einen kümmert sich die Software um die Verwaltung der annotierten bzw. noch nicht annotierten Sätze und deren In- und Output. Die Sätze werden aus einer Textdatei gelesen und als XML gespeichert. Dieser Teil ist mehrbenutzerfähig, da natürlich mehrere Personen gleichzeitig eine Instanz der Web-Applikation bedienen müssen.

Zum anderen enthält sie eine graphische Oberfläche mit der Annotationen leicht und schnell erstellt werden können:

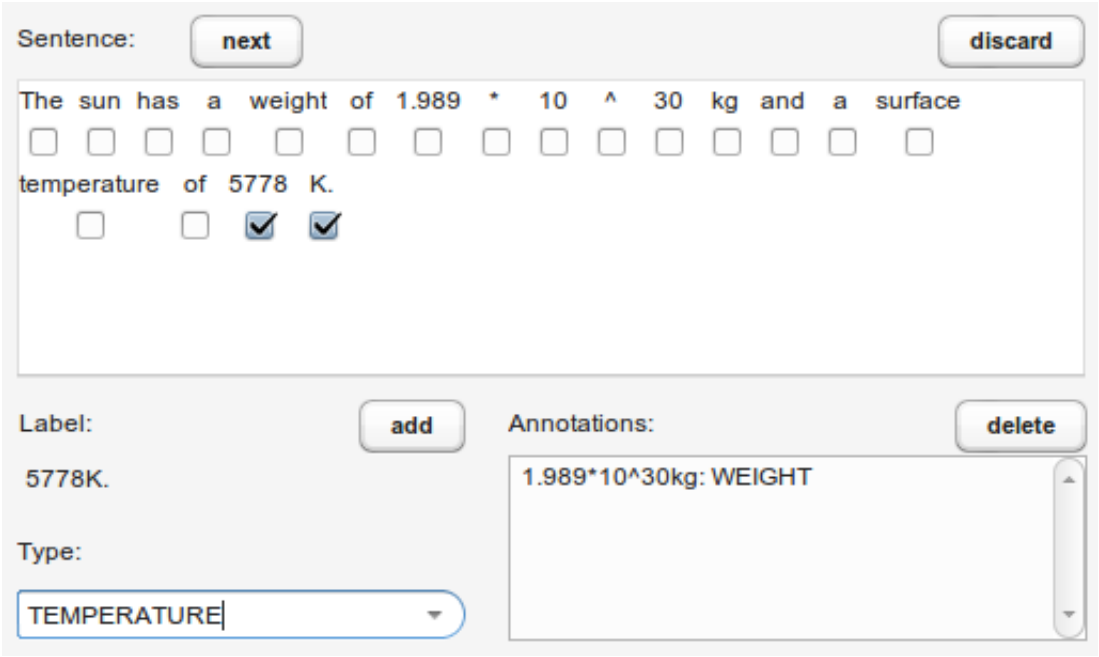

Abbildung 1: Die graphische Oberfläche

Der Ablauf der Benutzung ist dann wie folgt (Bezeichnungen der Knöpfe in Klammern):

1. Ein Satz wird geladen (next).
2. Eine Untermenge der Token des Satzes, aus denen das Label besteht, wird ausgewählt.
3. Der entsprechende Typ wird ausgewählt.
4. Die Annotation wird einer Liste hinzugefügt (add).
5. Nachdem alle Annotationen hinzugefügt worden sind, wird der Satz mit den Annotationen gespeichert und ein neuer geladen (next).

Es können auch einzelne Annotationen wieder gelöscht werden (delete), oder die gesamte Annotierung des Satzes abgebrochen werden (discard).

3 Grundsätzliche Struktur- und Entwurfsprinzipien für das Gesamtsystem

Um die Annotationen zu speichern, wird der jeweilige Satz in dem von der StanfordCoreNLP Pipeline generiertem XML Dokument erweitert. Dazu wird für jede Annotation ein `<annotation>` Element erzeugt, das die einzelnen Token, identifiziert durch ihre Id, und den Typ, enthält.

```
<sentence id="1">
   <tokens>
      <token id="1">
         <word>Tonkinese</word>
         <CharacterOffsetBegin>0</CharacterOffsetBegin>
         <CharacterOffsetEnd>9</CharacterOffsetEnd>
      </token>
      <!-- weitere Token -->
   </tokens>
   <annotation>
      <token>4</token>
      <token>5</token>
      <type>WEIGHT</type>
   </annotation>
   <!-- weitere Annotationen -->
</sentence>
```

Sowohl die Verwaltung der Sätze als auch sämtliche IO-Operationen sind durch die Klasse SentenceServer gekapselt. Dies ermöglicht eine saubere Trennung der GUI vom Restsystem, und macht es überflüssig die GUI thread-sicher zu programmieren. Stattdessen befinden sich alle kritische Abschnitte in den (statischen) Methoden der Klasseninstanz von SentenceServer. Der Server muss einmalig mit dem Pfad zu den Eingabe/Ausgabe Dateien initialisiert werden. Dabei wird geprüft ob bereits eine Ausgabedatei mit vorherigen Annotationen vorliegt.

Der Server stellt drei Methoden für die GUI bereit: getSentence (einen neuen Satz anfordern), returnSentence (einen annotierten Satz zurückgeben), und discardSentence (unannotierten Satz zurückgeben und wieder in der Textdatei speichern).

Für den Datenaustausch sind die beiden Klassen BoaAnnotation und BoaSentence zuständig. Um das System einfach zu halten, ist eine Instanz von BoaAnnotation nicht veränderbar, die Klasse enthält nur den Konstruktor und Getter-Methoden. Eine BoaSentence Instanz wird immer mit dem String des Satzes konstruiert, wobei automatisch mit Hilfe einer StanfordCoreNLP Pipeline die Tokenisierung vorgenommen wird.

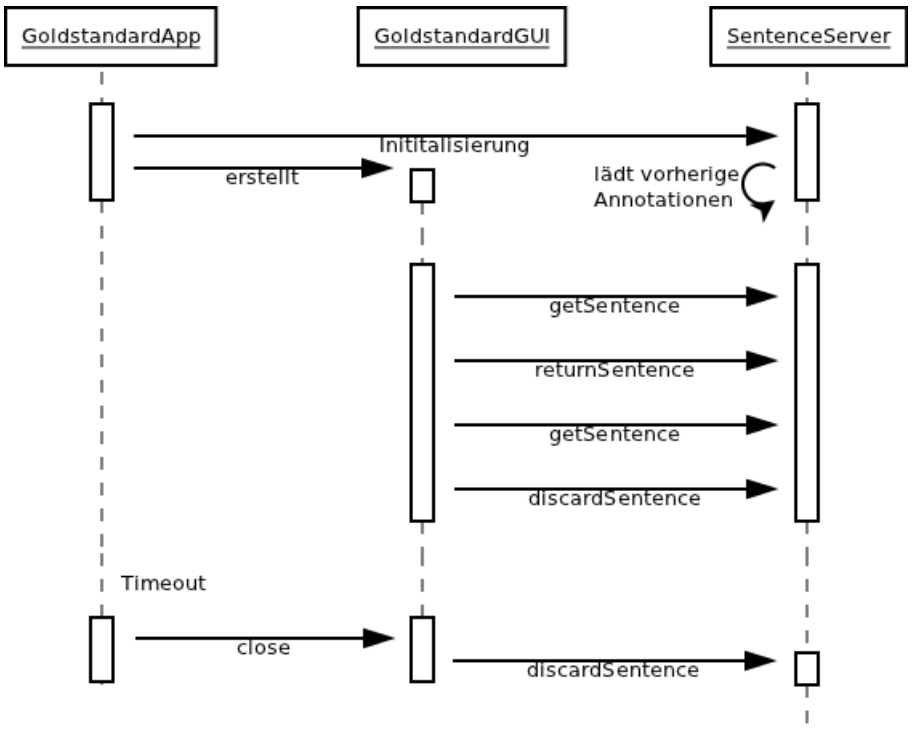

Abbildung 2: Aufrufe zwischen den Komponenten

4 Grundsätzliche Struktur- und Entwurfsprinzipien der einzelnen Pakete

4.1 SentenceServer

`static synchronized BoaSentence getSentence()`

Wandelt die Datei mit den Sätzen für unseren Goldstandard über die Hilfsfunktion fileToString in einen String um, extrahiert anschließend mittels nextSentence den nächsten Satz und versucht mit dem erhaltenen String ein Objekt vom Typ BoaSentence zu initialisieren. Bei einem Misserfolg wird null zurückgegeben.

`static synchronized void returnSentence(BoaSentence sentence)`

Zunächst wird die Repräsentation des Satzes als Stanford-XML in einem Objekt des Typs nu.xom.Document abgespeichert. Anschließend werden die von dieser Klasse bereitgestellten Funktionen zur Selbstmanipulation aufgerufen, um für jede Annotation des BoaSentence (die Array List<BoaAnnotation> wird mit einem Iterator abgearbeitet) in der XML ein Kindelement “Annotation“ bei “sentence“ hinzuzufügen, welches sich wiederum in mehrere Kindelemente “Token“ und einen “Typ“ aufteilt. Die Navigation innerhalb der XML Struktur erfolgt dabei über die Funktionen getRootElement und getFirstChildElement, das Hinzufügen selbst über appendChild. Wichtig ist dabei, dass neu hinzugefügte Objekte zuvor als nu.xom.Element initialisiert werden müssen.

Nachdem die XML Struktur mit unseren Attributen ergänzt wurde, wird die entsprechende XML

des Satzes zur Gesamt XML von unserem Goldstandard hinzugefügt. Dies erfolgt abermals über die appendChild Funktion von nu.xom.Document. Zuletzt wird die Hilfsfunktion xmlDocToFile aufgerufen, um die Java Object XML als Datei abzuspeichern.

`static synchronized void discardSentence(BoaSentence sentence)`

Ruft aodSentence auf, um die String-Repräsentation des eingegebenen Satzes wieder an den Anfang der Textdatei mit unseren (noch nicht annotierten) Goldstandardsätzen zu schreiben.

`private static String fileToString(File file)`

Eine UTF-8 encodierte Textdatei wird über einen InputStreamReader eingelesen und mit Hilfe eines Stringbuilders zeichenweise übertragen, wobei jeder char zunächst als Integer in einer lokalen Variable abgespeichert und, sofern sein ASCII Wert nicht -1 (end of file) entspricht, per typecast zum Stringbuilder hinzugefügt wird.

`private static String nextSentence(String text)`

Sofern der eingegebene String nicht leer ist, liefert die Methode den Teilstring bis zum ersten Auftreten eines Zeilenumbruchs. Dies wird über die Methode substring realisiert.

`private static void aodSentence(String sentence, File oldFile, Boolean direction)`

Der zu bearbeitende Text, wird mittels fileToString aus einer UTF-8 Textdatei eingelesen. Anschließend wird je nach gewählter Richtung sentence an den Beginn des so erhaltenen Strings hinzugefügt (true) oder alle Vorkommen von sentence mittels replace im Text entfernt (false). Der neue Text wird über einen OutputStreamWriter in einer temporären Datei abgespeichert, welche anschließend die alte Datei überschreibt (durch Umbenennung).

`private static void xmlDocToFile(Document doc)`

Nutzt einen Serializer aus dem nu.xom Packet, um unsere noch als Java Object (Document) vorliegende XML als Datei abzuspeichern. Auch hier wird zunächst eine temporäre Datei angelegt, bevor alte Versionen überschrieben werden.

4.2 Vaadin Applikation und GUI

Eine Instanz der Vaadin Applikation (GoldstandardApp.java) wird automatisch vom Servlet-Container erzeugt, sobald die Webseite aufgerufen wird. Anschließend wird eine Instanz der GUI konstruiert. Diese wird clientseitig mit Hilfe von HTML/CSS und JavaScript dargestellt. Die Ereignishandler werden direkt im Konstruktor als anonyme Klassen hinzugefügt.

Member von GoldstandardGUI.java mit der Annotation @AutoGenerated wurden von dem Visuellen Editor des Vaadin Eclipse-Plugins erstellt, und sollten nicht geändert werden.

5 Erstellung des Goldstandards

Für die Erstellung des Goldstandards wurden folgende Quellen benutzt:

Quelle	**Anteil**
en.wikipedia.org	66%
www.guinnessworldrecords.com	8%
andere Webseiten	7%
selbst verfasst	19%

Dabei wurde darauf geachtet möglichst viele verschiedenen Schreibweisen und Einheiten abzudecken. Zusätzlich gibt es einige Sätze ohne Annotationen. Diese sollen das auffinden von False Positives durch eine ungünstige Implementierung provozieren.

Der Goldstandard umfasst 513 Sätze mit 981 Annotationen (1,9 Annotationen/Satz).

Verteilung der Annotationen nach dem Typ:

Typ	**Anzahl**
DATE	249
LINEAR_MEASURE	350
TEMPERATURE	206
WEIGHT	176

Verteilung der Sätze nach der Anzahl n ihrer Annotationen:

n	**Anzahl**
0	36
1	188
2	174
3	55
4	42
>4	18