Entwurfsbeschreibung

Inhaltsverzeichnis

1 Struktur des Gesamtsystems

1.1 Sequenzdiagramm

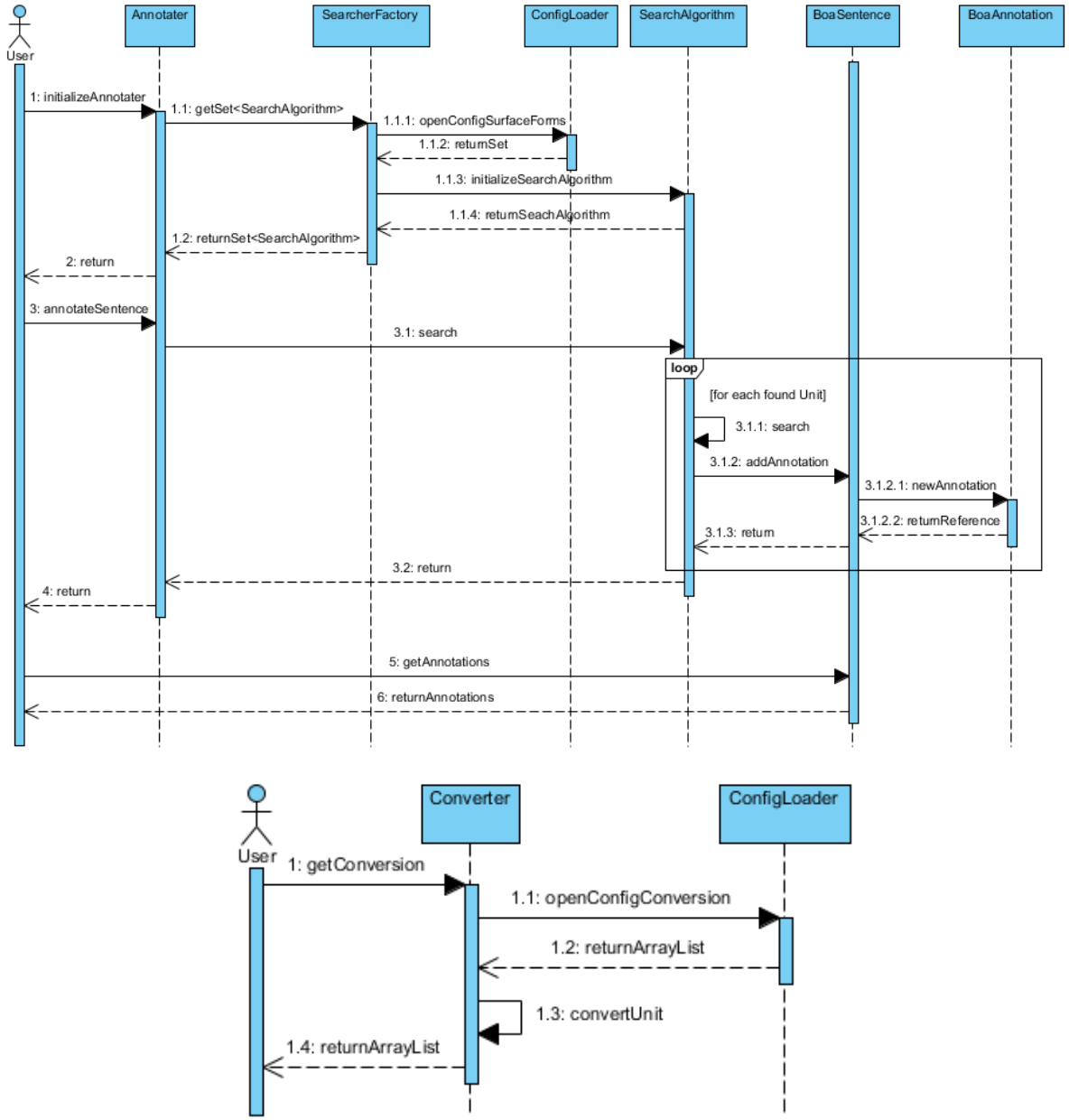

Abbildung 1: Sequenzdiagramm

Für das Gesamtprojekt gibt es zwei verschiedenen Abläufe: die Extraktion und die Konvertierung. Bei der Extraktion fordert die Klasse `Annotator` zuerst von der Klasse `SearcherFactory` die Suchalgorithmen für bekannte Einheiten an. Dafür lässt `SearcherFactory` die Oberflächenformen von der Klasse `ConfigLoader` laden, um mit denen die einzelnen von `SearchAlgorithm` abgeleiteten Suchklassen zu initialisieren.
Nachdem diese zurückgegeben wurden, sucht `Annotator` mithilfe dieser in dem Satz nach möglichen Einheitenformen. Wird eine Einheit gefunden, wird eine Annotation der Liste von Annotationen der Klasse `BoaSentence` hinzugefügt. Wurden alle Oberflächenformen gesucht, wird aus

dem Satz die Liste aller Annotationen extrahiert und zurückgegeben.
Bei der Konvertierung wird nur eine Annotation an die Klasse `Converter` übergeben. Diese lässt sich vom `ConfigLoader` die Umrechnungs- und Konvertierungstabellen zurückgeben. Anschließend wird die Annotation in alle möglichen Oberflächenformen der spezifischen Einheit umgewandelt und in einer Liste gespeichert, die zum Schluss an den Nutzer zurückgegeben wird.

Hinweis: Hilfsklassen, wie `SentenceLoader` oder ähnliche, wurden im Sequenzdiagramm nicht mit integriert. Denn der Ablauf würde durch sie nicht verändert werden.

1.2 Klassendiagramm

Abbildung 2: Klassendiagramm

2 Struktur der Einzelkomponenten

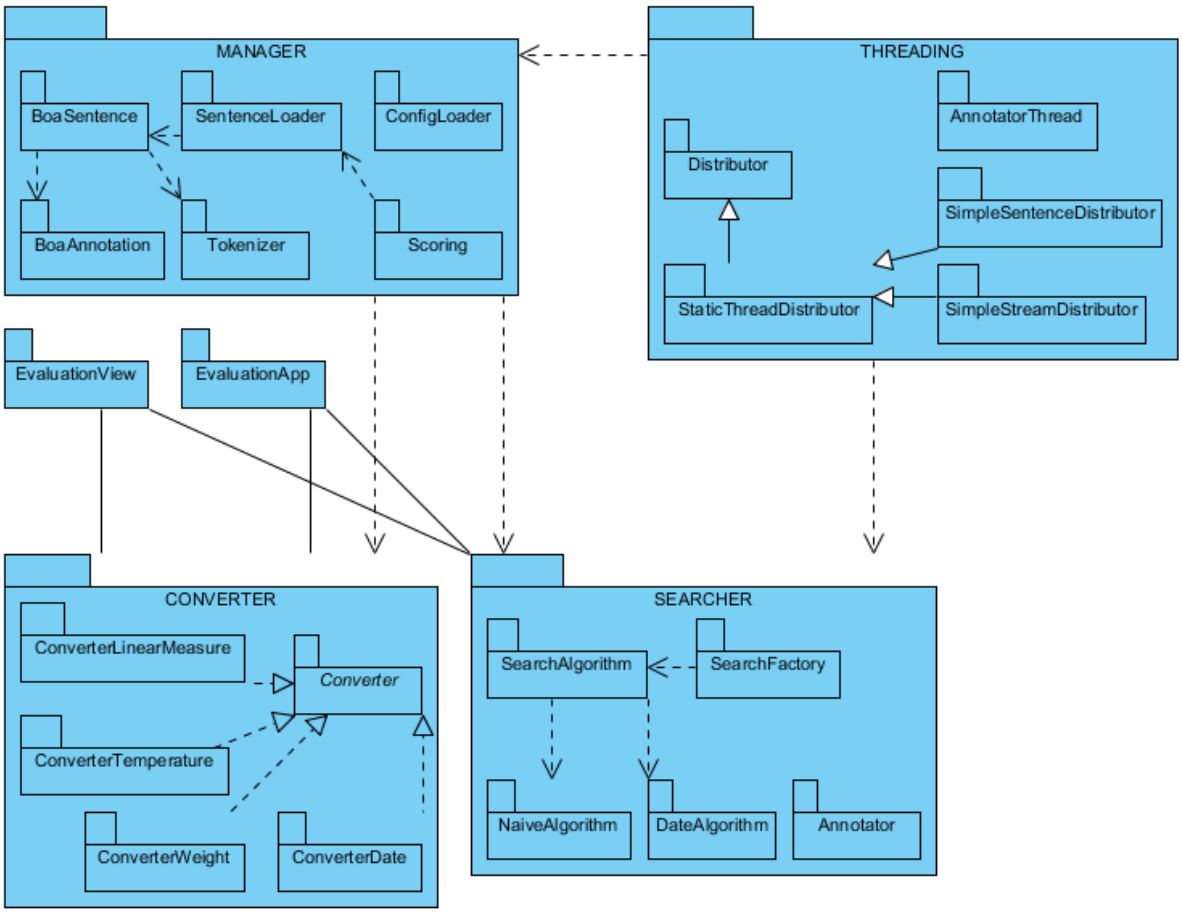

Abbildung 3: Einteilung in Pakete

2.1 Manager

Dieses Paket besteht aus den Verwaltungsklassen `ConfigLoader`, `SenctenceLoader`, `BoaAnnotation`, `BoaSentence`, `Tokenizer` und `Scoring`.

2.1.1 ConfigLoader

Diese Klasse dient grundlegend für das Laden von Daten aus der Config-Datei, in der die Oberflächenformen und Umrechnungsdaten gespeichert sind. Zurückgegeben wird entweder ein Liste aus Strings (für LabelSearcher) oder eine HashMap auf Strings und BigDecimals (für Converter).

2.1.2 SentenceLoader

Diese Klasse stellt verschiedene Methoden zum Auslesen von `BoaSentence` aus unseren `xml`-Dateien zur Verfügung.

2.1.3 BoaAnnotation

Dies ist die Grundklasse für das Annotationssystem. Sie speichert zu jedem Label, das aus verschiedenen Tokens besteht, den entsprechenden Typ ab.

2.1.4 BoaSentence

Diese Klasse beinhaltet die grundlegenden Informationen, die vom Programm bearbeitet werden sollen, d.h. den Satz, die dazugehörenden Tokens, die Liste von entstandenen Annotationen und der Verweis auf das dazugehörige `xml`-Dokument.

2.1.5 Tokenizer

Diese Klasse dient als Hilfswerkzeug für die Tokenisierung von Sätzen und kann `BoaSentence`-Objekte zurückgeben, die für weitere Verarbeitungen gebraucht werden.

2.1.6 Scoring

In dieser Klasse erfolgt größtenteils der Evaluationsprozess. Dabei werden Precision, Recall sowie F-Score berechnet.

2.2 Searcher

Dieses Paket besteht aus der Klasse `SearchFactory` sowie den Klassen `Annotater`, `SearchAlgorithm`, `NaiveAlgorithm` und `DateAlgorithm`. Es ist für das Durchsuchen des Eingabesatzes nach Einheitenvorkommen sowie deren Annotierung verantwortlich und beinhaltet damit die Implementierung der ersten Kernfunktion unseres Gesamtprojektes.

2.3 Converter

Dieses Paket besteht aus der abstrakten Klasse `Converter` und den davon abgeleiteten Klassen, welche die Funktionen für den zweiten Teil unserer Projektanforderung bereitstellen. Sie konvertieren übergebene Einheiten in alle weiteren Oberflächenformen, die zur jeweiligen Einheit in unseren Konfigurationsdateien gespeichert sind, zurück.

2.4 Threading

Um den Annotationsvorgang zu parallelisieren, gibt es die Klasse `AnnotatorThread`. Ein solcher Thread sorgt sowohl für die Tokenisierung als auch für das Annotieren einer bestimmten Teilmenge der Eingabedaten. Diese erhalten sie von einer Instanz der abstrakten Klasse `Distributor`, die Text oder bereits tokenisierte Sätze an die Threads verteilt, und das Ergebnis zurück erhält. Es existieren zwei Implementierungen von `Distributor`: `SimpleSenteceDistributor`, die `BoaSentence` Objekte verteilt, und `SimpleStreamDistributor`, die zerstückelten Text verteilt.
Da sämtliche Klassen nicht thread-sicher sind, konstruiert sich jeder Thread, mit Hilfe der übergebenen `SearcherFactory` eigene `Annotator`- bzw. `SearchAlgorithm` Objekte.

2.5 BoaGUI

Dieses Paket besteht nur aus der Klasse `BoaGUI` und entspricht dem View der *MVC-Architektur*. Hier wird die Anordnung der einzelnen Elemente des *Graphical User Interface* (kurz: GUI) gespeichert und dargestellt. Des Weiteren ist das Paket für die Entgegennahmen von Benutzerinteraktionen zuständig, wobei die Verarbeitung dieser in der Klasse `BoaApp` durchgeführt werden.

2.6 BoaApp

Dieses Paket besteht nur aus der Klasse `BoaApp` und entspricht dem Controller der *MVC-Architektur*. Sie ist für die Weiterverarbeitung der Benutzerinteraktionen zuständig und reagiert entsprechend auf diese.