Ein Eingabeassistent zum Programmieren in natürlicher Sprache

Dokumentenart: Exposé für eine Diplomarbeit
Autor: Jonathan Best
Studiengang: Informatik (Diplom)
Betreuer: Mathias Landhäußer
Datum: 24. April 2014

1 Motivation

Die Einführung der abstrakten Programmiersprachen in den 50er Jahren hätte jedermann dazu befähigen können, einen Computer zu programmieren. Dennoch ist das Programmieren auch heute noch eine vorwiegend handwerkliche Fertigkeit, die in intensivem Studium mühsam erworben wird. Die Schwierigkeit beim Programmieren liegt darin, die Beschreibung des Problems und seiner Lösung in die Formen der strukturierten Programmiersprache zu überführen.

Das AliceNLP-Projekt [TLK13] hat sich zum Ziel gesetzt, diese Schwierigkeit zu mildern. Dem Programmierer soll erlaubt sein, Problem und Lösung in natürlicher Sprache zu beschreiben und die Umsetzung in Programmcode der Maschine zu überlassen. Damit soll „jedermann“ befähigt werden, das eigene Gerät mit natürlicher Sprache zu steuern. Als exemplarische Domäne wurde dafür die Entwicklungsumgebung Alice [CDP00] gewählt. Alice bietet eine große Auswahl an Gegenständen (Entitäten), die auch ein Nicht-Programmierer kennt und manipulieren kann. Der Anwender soll in die Lage versetzt werden, mit einem englischen Text Alice-Szenen zu erzeugen und zu animieren. Damit soll das Projekt zeigen, dass die „Programmierung“ in natürlicher Sprache in prinzipiell machbar ist.

2 Zielsetzung

Obwohl die gesamte Fülle der englischen Sprache als Eingabe für AliceNLP akzeptiert werden soll, muss der Anwender sich trotzdem präzise ausdrücken. Ähnlich wie bei einer Software-Spezifikation muss der Anwender sich präzise und unmissverständlich ausdrücken. Vorangegangene Arbeiten [KB10] im Bereich des Requirement Engineering haben gezeigt, dass die Sprachverarbeitung hilfreich ist, um Spezifikationen zu verbessern. Wir möchten eine ähnliche Technik einsetzen, um bekannte Probleme bei der Eingabe zu

vermeiden. Der Eingabeassistent für AliceNLP wird den Benutzer darauf hinweisen, wenn sein Text bekannte Probleme aufweist und soweit möglich Vorschläge zur Korrektur geben.

Drei Sachverhalte sollen erkannt und korrigiert werden:

1. Die Initialisierung der Entitäten sollte vom Anwender beschrieben werden, weil in Alice bereits zum Beginn einer Szene alle Entitäten vorhanden sein müssen. Insbesondere Position, Blickwinkel und Sichtbarkeit sollte der Anwender spezifizieren.

2. Es soll erkannt werden, ob ein Satz entweder eine Beschreibung der Szene darstellt, einen Handlungsablauf beschreibt oder ein Ereignis in der Szene beschreibt. Fällt ein Satz in keine der drei Kategorien, hat er keinen Einfluss auf AliceNLP und sollte entfernt werden. (Beispielsweise Modalsätze)

3. Die Zuordnung von Personal-, Akkusativ- und Possesivpronomen in der Beschreibung (Koreferenz) soll festgestellt und angezeigt werden. Eine falsche Koreferenzanalyse ist eine potentielle Fehlerquelle.[1]

Die Art und Weise, wie diese Regeln überprüft werden, soll die Möglichkeit zur Erweiterung bieten. Zum jetzigen Zeitpunkt sind unsere Kenntnisse über Fehler und Fehlerquellen nur gering. Es ist abzusehen, dass im Verlauf des Projekts weitere Fehler offensichtlich werden, welche man mit dem Eingabeassistenten vermeiden oder entschärfen kann.

Für AliceNLP-Beschreibungen gelten ähnliche Regeln wie für Software-Spezifikation. In Anlehnung an die Arbeit von Körner und Brumm [KB10] sollen deshalb zusätzliche Eigenschaften erkannt werden. Darunter insbesondere:

- Nominalsätze (Nominalization)
- Mehrdeutigkeit (Polysemy)
- Unvollständige Argumentliste bei Verben (Incompletely specified process words)

Die Erkennung dieser Eigenschaften überschneidet sich allerdings mit anderen Arbeiten im Projekt und wird deshalb nachrangig behandelt.

[1] Die Koreferenzanalyse für AliceNLP ist ein besonderes Problem, da das Pronomen „it“ in den Beschreibungen häufig auftaucht und kein uns bekanntes Analysesysteme dieses zuverlässig behandelt.

2.1 Beispiel

Der Anwender möchte, dass eine Entität „Fluffy“ zu einer Straßenlaterne geht. Er verfasst die folgenden Zeilen:

> There is Fluffy.
> Fluffy walks to a lamppost.

AliceNLP benötigt für eine geeignete Initialisierung eine Positionsangabe, die entweder relativ zur Kamera oder zu einem anderen Objekt angegeben wird. Außerdem wird angenommen, dass alle genannten Dinge zu Beginn sichtbar und über die Szene verteilt sind. Der Anwender sollte sich dessen bewusst sein, damit er davon abweichende Anweisungen geben kann. Der Eingabeassistent würde folgende Dinge bemängeln:

1. Wo befindet sich „Fluffy“ zum Anfang der Szene? Vorschlag: Auf der linken Seite.
2. Wo befindet sich die Straßenlaterne zum Anfang der Szene? Vorschlag: Auf der rechten Seite.

Der Anwender hat die Möglichkeit, die Vorschläge mit einem Mausklick anzunehmen. Die automatische Korrektur sähe aus wie folgt:

> There is Fluffy to the left.
> There is a lamppost to the right.
> Fluffy walks to the lamppost.

Mit diesen expliziten Anweisungen kann AliceNLP die Entitäten positionieren. Dem Anwender sollte nun offensichtlich sein, wie AliceNLP seine Eingabe interpretieren wird und wo man ansetzen muss, um die Interpretation in seinem Sinne zu beeinflussen.

3 Evaluation

Der Zweck des Assistenten ist, den Anwender beim Editieren zu unterstützen. In der Evaluation soll der Verlauf der Änderungen an einem Dokument nachvollzogen und bewertet werden. An diesem Verlauf möchten wir erkennen, ob der Assistent eine nützliche Erweiterung des Projekts darstellt. Dazu gehen wir, wie bei Hampel [Ham12], von einem existierenden Video aus, das von allen Probanden beschrieben wird. Die dabei entstehenden Beschreibungen werden untereinander verglichen, sowie mit einer gesondert erstellten Musterlösung. Wir instrumentieren den Texteditor, sodass dieser Revisionen des Dokuments anlegt (konzeptionell wie bei [MH02]). Jedes Mal, wenn die Fehlererkennung des Assistenten läuft, wird gleichzeitig eine Revision erstellt.

An jeder Revision wird vermerkt, welche Hinweise der Assistent zu diesem Zeitpunkt angezeigt hat. Am Änderungsverlauf lässt sich nun nachvollziehen, welche Meldung vom Benutzer korrigiert wurde und ob diese Korrektur erfolgreich war. Diese Untersuchung soll mit fünf oder sechs Probanden durchgeführt werden, die mit dem Assistenten zum ersten Mal arbeiten.

Als Alternative können wir auch die bereits existierenden Beschreibungen aus dem Alice-Korpus [Ham12] mit ihrer Musterlösung zu vergleichen. Bei dieser Methode würde evaluiert, ob der Assistent die Abweichungen oder Fehler aus dem Korpus erkennt und ob er die Musterlösung behandelt, ohne Fehlermeldungen auszugeben.

Literatur

[CDP00] Cooper, Stephen ; Dann, Wanda ; Pausch, Randy: Alice: a 3-D Tool for Introductory Programming Concepts. (2000)

[Ham12] Hampel, Sina: *Programmieren in natürlicher Sprache: Aufbau eines Alice-Korpus*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Bachelor's Thesis, 2012. `http://www.ipd.kit.edu/Tichy/theses.php?id=199`

[KB10] Körner, Sven J. ; Brumm, Torben: Natural Language Specification Improvement with Ontologies. In: *IJSC* 3 (2010), Nr. 4, 445–470. `http://ps.ipd.kit.edu/backend/index.php/veroeffentlichungen-details/items/3527.html`

[MH02] Müller, Matthias M. ; Hagner, Oliver: Experiment about Test-First Programming. In: *IEE Proceedings – Software* Bd. 149, 2002, S. 131–136

[TLK13] Tichy, Walter F. ; Landhäusser, Mathias ; Körner, Sven J.: Universal Programmability - How AI Can Help. In: *2nd International NSF sponsored Workshop on Realizing Artificial Intelligence Synergies in Software Engineering*, 2013