Das Testen und Test-Driven Development von Conversational AI

Hochschule Darmstadt
Fachbereiche Mathematik und Naturwissenschaften & Informatik

Motivation

Das Thema dieser Arbeit ist die Erläuterung und Anwendung der agilen Softwareentwicklung mit dem Test-Driven Development (TDD) für die Entwicklung einer Conversational AI. Die Anpassung von Softwareentwicklungsmethoden an Machine Learning (ML) ist ökonomisch sinnvoll, da ML immer häufiger in Software verwendet wird. In der Softwareentwicklung werden standardisierte Vorgehensweisen wie das TDD genutzt, da diese eine simple, strukturierte und zügige Bearbeitung der Projekte erleichtern. Die Problematik, diese Vorgehensweisen für ML-Systeme zu verwenden, liegt im Wesentlichen darin, dass diese Vorgehensweisen für die Entwicklung von gleichbleibenden, eher deterministischen Systemen ausgelegt sind. Dem gegenüber stehen ML-Systeme, welche durch die Notwendigkeit des regelmäßigen Trainierens der Modelle eine sich verändernde und statistische Natur aufweisen. Für die Entwicklung einer Conversational AI wurde sich entschieden, da dies durch die praktische Anwendung erfordert wurde und die Verwendung von Transformern eine kürzliche, sehr interessante Neuerung in diesem Bereich bietet.

Ziel und Vorgehen

Die zentrale Fragestellung dieser Arbeit ist, ob und wie Test-Driven Development anwendbar ist, um eine Conversational AI zu entwickeln. Um die Fragestellung zu beantworten, wurde ein Vorgehen vorgestellt, welches nach den Prinzipien des TDD handelt und die Entwicklung einer Conversational AI ermöglicht. Konkret wurde versucht das zyklische Bearbeiten der Tests, das sog. Mantra des TDD, beizubehalten. Dieses Mantra besagt, dass jeder Test in drei Phasen implementiert und umgesetzt wird:

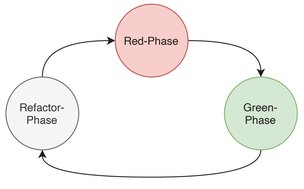

Figure 1: TDD Zyklus

Hierbei wird in der Red-Phase der Test gewählt und implementiert. Daraufhin wird erst in der Green-Phase der entsprechende Code umgesetzt, um den Test zu bestehen. Abschließend wird der Code in der Refactor-Phase bereinigt und es kann mit dem nächsten Test begonnen werden. Durch das Beibehalten dieses Vorgehens sollten möglichst viele der Vorteile des TDD auch für die Entwicklung einer Conversational AI erhalten werden. Bei dem Vorgehen mussten dabei sowohl die Anforderungen an die Conversational AI als auch die Unterschiede und Besonderheiten des Machine Learnings beachtet werden.
Bei der praktischen Umsetzung wurden zwei Frameworks genutzt. Zum einen wurde das Framework Rasa verwendet, um die Conversational AI zu entwickeln und zum anderen wurde das Framework CheckList verwendet, um das TDD mit Rasa zu ermöglichen.

Ergebnisse

In dieser Arbeit wurde eine Vorgehensweise der agilen Softwareentwicklung vorgestellt, welche ermöglicht eine Conversational AI zu entwickeln. Hierbei wurde ein Fokus darauf gelegt die Prinzipien und den Grundgedanken des TDD beizubehalten.
Im Allgemeinen bietet das TDD vor allem Vorteile im Bereich der gedanklichen Strukturierung, Implementierung und Kommunikation. Diese Vorteile wurden auch beim Entwickeln der Conversational AI festgestellt. Ein Beispiel dieser Vorteile ist das sorgenfreie Implementieren, da sowohl das Testen vor die Implementierung des funktionalen Codes gezogen wurde als auch dass diese bereits implementierten Tests dem Entwickler Sicherheit geben, dass die aktuelle Anwendung funktionstüchtig ist. Ein anderer Vorteil ist, dass durch die Fokussierung auf Tests die Kommunikation über die Anwendung verbessert werden kann. In einem Entwicklungsteam können anhand der Tests Probleme identifiziert und Anforderungen besprochen werden. Hierbei ist besonders hervorzuheben, dass bei dieser Kommunikation die Vorkenntnisse weitestgehend vernachlässigt werden können. Somit können sowohl Programmierer mit Nicht-Programmierern als auch ML-Verständige mit Nicht-ML-Verständigen besser und leichter kommunizieren.
Ein wesentlicher Unterschied bei dem Vorgehen ist, dass bei einem ML-System nicht alle möglichen Inputs getestet werden können. Daher können die Tests lediglich eine untere Schranke an die Funktionalität sicherstellen. Diese minimale Funktionalität muss daher die durch die Anforderungen definierten Testfälle für das Modell sicherstellen und gewährleisten.

Fazit

Die Anwendung wurde erfolgreich mit dem vorgestellten Vorgehen umgesetzt.
Hierbei wurden die Anforderungen an die Conversational AI erfüllt und die Funktionalitäten erfolgreich umgesetzt. Diese werden auch durch die bei dem Vorgehen entwickelten Tests sichergestellt. Besonders hervorzuheben sind die Vorteile, die das TDD mitbringt. Hierzu zählen sowohl eine gute gedankliche Struktur der Implementierung als auch die Unterstützung bei der Umsetzung der einzelnen Schritte. Des Weiteren ist hervorzuheben, dass das TDD bei der Entwicklung von ML besonders vorteilhaft für die langfristige Instandhaltung ist. Hierfür wird betont, dass die alleinige Entwicklung von ML-Modellen in der Praxis nicht ausreichend ist. Das TDD bietet hierfür den Vorteil, dass jegliche Voraussetzungen an die Anwendung bereits mit einem Test gesichert sind. Somit kann die, an die Entwicklung anschließende, dauerhafte Instandhaltung von ML-Modellen auf diese Tests zurückgreifen und durch wenige Ergänzungen eine dauerhafte Performanz des Modells in der Entwicklung sicherstellen.

References

[1] Ziv Carmon, Klaus Wertenbroch, Haiyang Yang, and Rom Schrift. Designing ai systems that customers won't hate. *MIT Sloan Management Review*, 12 2019.

[2] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, 2020.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Kontakt

Name Alexander Kniesz

Email alexander.kniesz@stud.h-da.de