Anforderungen-zu-Quelltext-Rückverfolgbarkeit mittels Wort- und Quelltexteinbettungen

Dokumentenart:	Exposé für eine Masterarbeit
Autor:	Fei Chen
Matrikel-Nr.:	1751217
Studiengang:	Informatik Master
Betreuer:	M.Sc. Tobias Hey
Datum:	14. Oktober 2020

1 Motivation

Anforderungsrückverfolgbarkeit bezeichnet allgemein das Abbilden und Zuordnen von Anforderungen auf andere Artefakte aus dem Entwicklungsprozess, wie zum Beispiel auf Testfälle oder auf Teile der Implementierung. Mit den Rückverfolgbarkeitsinformationen können Verfahren wie eine Abdeckungsanalyse oder eine Analyse über den Einfluss von Änderungen durchgeführt werden, was wiederum durch das bessere Verständnis Kosten und Zeit einspart [DP98].

Die Erzeugung der Rückverfolgbarkeitsverbindungen (Trace-Links) erfolgt oft manuell durch den Entwickler und ist aufwändig, daher werden Ansätze zur Automatisierung benötigt. Existierende Werkzeuge zur Anforderungsrückverfolgbarkeit erreichen nicht den vollen Automatisierungsgrad, der für einen produktiven Einsatz nötig wäre.

Bei der vollautomatischen Anforderungsrückverfolgbarkeit gibt es eine Reihe von Hürden, die es schwierig machen, Anforderungen und Quelltext direkt miteinander zu verbinden. Zum Beispiel beschreiben Anforderungsdokumente und Quelltext in unterschiedlichen Detailgraden. Außerdem können die zu verbindenden Artefakte vom Format her unterschiedlich aufgebaut und verfasst sein.

Als mögliche Lösung hierzu bietet sich ein Ansatz mit Wort- und Quelltexteinbettungen, die als Zwischenrepräsentation verwendet werden, an. Eine Worteinbettung repräsentiert ein Wort durch einen Vektor im Vektorraum. Analog verhält es sich mit Quelltextausschnitten und Quelltexteinbettungen. Durch die Einbettungen wird von der konkreten Beschreibung und Darstellung der Anforderungen und des Quelltextes abstrahiert und die Verbindung

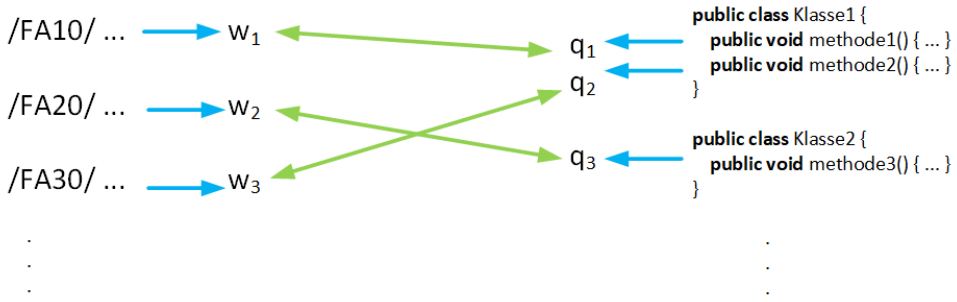

Abbildung 1: Ein Beispiel zur Zuordnung bei der Anforderungsrückverfolgbarkeit.

zwischen Anforderungen und Quelltext entspricht nun einer Abbildung zwischen Vektorräumen. Bei Worteinbettungen werden Informationen über die Kontextähnlichkeit erhalten, das heißt Vektoren von Wörtern, die in ähnlichen Kontexten auftauchen, liegen im Vektorraum nah beieinander. Dadurch lassen sich Wörter durch Einbettungen besser verallgemeinern, wovon man sich wiederum eine bessere Abbildung zwischen Einbettungen verspricht. Außerdem hängt die Größe der Vektoren nicht von der Größe des Vokabulars ab, was Einbettungen gut skalieren lässt [BDVJ03].

1.1 Beispiel

In Abbildung 1 ist ein Beispiel zur Anforderungsrückverfolgbarkeit mit Einbettungen zu sehen. Die Anforderungen, die in natürlicher Sprache geschrieben sind, werden auf Einbettungsvektoren abgebildet, ebenso der Quelltext. Die Einbettungen bieten nun die Möglichkeit, durch eine Abbildung zwischen den Vektorräumen in Verbindung gesetzt zu werden. Zum Beispiel könnte /FA10/ die Anforderung „Der Bildverarbeiter soll ein Bild nach Graustufen filtern können“ sein. Weiterhin sei „methode1()“ die passende Implementierung dazu. Die Anforderung und die Methode werden als Erstes auf die Worteinbettung w_1 bzw. Quelltexteinbettung q_1 abgebildet und danach werden die Einbettungen in Verbindung gesetzt.

2 Zielsetzung

Es soll untersucht werden, inwiefern sich Anforderung-zu-Quelltext-Rückverfolgbarkeit unter Einsatz von Wort- und Quelltexteinbettungen realisieren lässt. Die Einbettungen dienen als Zwischenrepräsentation für die Anforderungen und den Quelltext. Hierfür sollen zunächst verschiedene Möglichkeiten zur Generierung von Wort- und Quelltexteinbettungen geprüft werden.

Danach soll ein Verfahren erarbeitet werden, um korrespondierende Anforderungen und Quelltext einander zuzuordnen.

Das Programm erhält also Anforderungen in natürlicher Sprache und den passenden Quelltext als Eingabe und liefert als automatische Ausgabe die Rückverfolgbarkeitsverbindungen zurück.

3 Vorgehensweise

Für die Anforderung-zu-Quelltext-Rückverfolgbarkeit müssen zuerst Einbettungen erzeugt werden. Es existieren bereits Verfahren zur Generierung von Worteinbettungen. Zum Beispiel gibt es word2vec [MCCD13], welches Worteinbettungen durch Vorhersage des Wortkontextes trainiert. Außerdem existieren auch weitere Verfahren wie ELMo [PNI+18], die zusätzlich Polysemie berücksichtigen und für jede Bedeutung eine eigene Worteinbettung generieren.

Word2vec oder ELMo generieren *pro Wort* eine Einbettung. Im Rahmen der Anforderungen-zu-Quelltext-Rückverfolgbarkeit sollen Anforderungen zu Quelltext zugeordnet werden, aber eine Anforderung besteht in der Regel aus mehreren Wörtern oder Sätzen. Daher muss auch untersucht werden, wie man mehrere Wörter oder Sätze auf eine Einbettung abbilden kann. Mögliche Ansätze hierzu sind zum Beispiel die Berechnung einer Paragrapheinbettung [LM14], welche einen beliebig langen Textausschnitt repräsentiert. Eine weitere Möglichkeit wäre, die Anforderung durch ein Label zu repräsentieren, von dem dann die Worteinbettung berechnet wird.

Zur Generierung von Quelltexteinbettungen gibt es auch bereits Arbeiten: code2vec [AZLY19] ist in der Lage, einer ganzen Methode eine Quelltexteinbettung zuzuweisen. Da code2vec auf der Analyse des Syntaxbaums des Quelltextes aufbaut, ist dieser Ansatz programmiersprachenabhängig. Ein weiterer Ansatz ist IR2Vec [SAJ+19]. Hier wird der Quelltext zunächst in die LLVM-Zwischenrepräsentation überführt, worauf im zweiten Schritt die Einbettungen für einzelne Ausdrücke, aber auch für ganze Methoden oder Klassen durch Kombination der Einbettungen ihrer einzelnen Ausdrücke berechnet wird. Durch die Nutzung der LLVM-Zwischenrepräsentation ist dieser Ansatz unabhängig von der Programmiersprache.

Bei den Quelltexteinbettungen muss außerdem die Granularität der Zuordnung geprüft werden; das heißt ob man eine Anforderung zu einer Methode oder zu einer ganzen Klasse zuordnet.

Nachdem die Quelltext- und Anforderungseinbettungen erzeugt wurden, müssen diese einander passend zugeordnet werden. Hierzu gibt es aus dem Bereich der Übersetzung zwischen Sprachen ähnliche Arbeiten. Dort werden auch für die Wörter aus der Quell- und Zielsprache Worteinbettungen generiert; anschließend wird durch maschinelles Lernen eine lineare Transformation in Form einer Matrix trainiert, die die Worteinbettungen aus der

Quellsprache in die Zielsprache abbilden soll.

[AMD18] und [YLW+18] sind Arbeiten, in denen eine Transformationsmatrix auf einer Menge von Übersetzungswortpaaren durch ein unüberwachtes Lernverfahren trainiert wird. Je nach Arbeit wurden für das Training verschiedene Fehlermetriken wie die euklidische Distanz oder die maximale mittlere Diskrepanz (Maximum Mean Discrepancy) untersucht.

Da in der Sprachübersetzung für manche Sprachen nicht genügend große Mengen an Übersetzungswortpaaren vorhanden sind, wurden einige Verfahren entwickelt, die ohne viele Trainingsdaten funktionieren.

In [ALA17] wurde ein Verfahren vorgestellt, das nur ein Startwörterbuch mit 25 Einbettungspaaren (bestehend aus den Einbettungen eines Worts und ihrer Übersetzung) benötigt. Damit wird eine Transformationsmatrix berechnet, wobei der Fehler (gemessen als euklidische Distanz) zwischen der Multiplikation der Worteinbettung mit der Transformationsmatrix und der Einbettung der eigentlich Übersetzung minimiert wird. Anschließend wird mit Hilfe der Transformationsmatrix das Wörterbuch erweitert, indem neue (und alte) Einbettungen aus der Quellsprache mit der Transformationsmatrix multipliziert werden und die nächstliegende Einbettung aus der Zielsprache als korrespondierende Übersetzung festgelegt wird. Dies wird iterativ fortgesetzt.

Eine anderen Arbeit [CLR+17] benötigt gar keinen Trainingsdatensatz mit initialen Übersetzungspaaren. Stattdessen wird ein gegnerisches Netzwerk (adversarial network) verwendet. Dieses besteht aus zwei eigenständigen neuronalen Netzen, dem Generator und dem Diskriminator. Der Generator versucht, mit der Transformationsmatrix multiplizierte Einbettungen aus der Quellsprache wie Einbettungen aus der Zielsprache aussehen zu lassen. Der Diskriminator wird darauf trainiert, diese unterscheiden zu können. Da bekannt ist, ob eine Einbettung aus der Zielsprache kommt oder ob es eine mit der Transformationsmatrix multiplizierte Einbettung ist, können sich der Generator (und damit die Transformationsmatrix) und der Diskriminator schrittweise verbessern.

Für diese Arbeit soll untersucht werden, welche und inwiefern diese Verfahren zwischen Anforderungs- und Quelltexteinbettungen angewendet werden können. Insbesondere die Verfahren mit wenigen bzw. ohne Trainingsdaten sind interessant, da es für Anforderungsrückverfolgbarkeit keine sehr großen Trainingssätze gibt.

4 Evaluation

In der Evaluation soll überprüft werden, wie gut sich die Anforderungsrückverfolgbarkeit mit Einbettungen lösen lässt. Dafür können existierende Datensätze zur Anforderungsrückverfolgbarkeit genutzt werden. In [ZSMA17] wurden viele solcher Datensätze zusammengetragen und klassifiziert. Vor al-

lem solche, die quelloffen sind und Anforderungen zu Code, Klassen oder Methoden zuordnen, können genutzt werden.

Ein Teil davon wird für das Training des maschinellen Lernverfahrens verwendet und ein anderer Teil wird ausschließlich zur Evaluation verwendet, um zu prüfen, wie sich das Verfahren bei im Training ungesehenen Anforderungen schlägt.

Um diese Arbeit mit anderen Arbeiten vergleichbar zu machen, wird das Einbettungsverfahren auf die Datensätze angewendet und danach die üblichen Metriken - Präzision, Ausbeute und das F1-Maß - berechnet.

Literatur

[ALA17] Artetxe, Mikel ; Labaka, Gorka ; Agirre, Eneko: Learning Bilingual Word Embeddings with (Almost) No Bilingual Data. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada : Association for Computational Linguistics, Juli 2017, S. 451–462

[AMD18] Aldarmaki, Hanan ; Mohan, Mahesh ; Diab, Mona: Unsupervised Word Mapping Using Structural Similarities in Monolingual Embeddings. In: *Transactions of the Association for Computational Linguistics* 6 (2018), August, S. 185–196. `http://dx.doi.org/10.1162/tacl_a_00014`. – DOI 10.1162/tacl_a_00014

[AZLY19] Alon, Uri ; Zilberstein, Meital ; Levy, Omer ; Yahav, Eran: Code2Vec: Learning Distributed Representations of Code. In: *Proc. ACM Program. Lang.* 3 (2019), Januar, Nr. POPL, S. 40:1–40:29. `http://dx.doi.org/10.1145/3290353`. – DOI 10.1145/3290353. – ISSN 2475–1421

[BDVJ03] Bengio, Yoshua ; Ducharme, Réjean ; Vincent, Pascal ; Janvin, Christian: A Neural Probabilistic Language Model. In: *J. Mach. Learn. Res.* 3 (2003), März, S. 1137–1155. – ISSN 1532–4435

[CLR+17] Conneau, Alexis ; Lample, Guillaume ; Ranzato, Marc'Aurelio ; Denoyer, Ludovic ; Jégou, Hervé: Word Translation Without Parallel Data. (2017), Oktober

[DP98] DömgesRalf ; PohlKlaus: Adapting Traceability Environments to Project-Specific Needs. In: *Communications of the ACM* (1998), Dezember

[LM14] Le, Quoc ; Mikolov, Tomas: Distributed Representations of Sentences and Documents. In: *International Conference on Machine Learning*, 2014, S. 1188–1196

[MCCD13] Mikolov, Tomas ; Chen, Kai ; Corrado, Greg ; Dean, Jeffrey: Efficient Estimation of Word Representations in Vector Space. In: *arXiv:1301.3781 [cs]* (2013), September

[PNI+18] Peters, Matthew E. ; Neumann, Mark ; Iyyer, Mohit ; Gardner, Matt ; Clark, Christopher ; Lee, Kenton ; Zettlemoyer, Luke: Deep Contextualized Word Representations. In: *arXiv:1802.05365 [cs]* (2018), März

[SAJ+19] S, Venkata K. ; Aggarwal, Rohit ; Jain, Shalini ; Desarkar, Maunendra S. ; Upadrasta, Ramakrishna ; Srikant, Y. N.: IR2Vec: A Flow Analysis Based Scalable Infrastructure for Program Encodings. In: *arXiv:1909.06228 [cs]* (2019), September

[YLW+18] Yang, Pengcheng ; Luo, Fuli ; Wu, Shuangzhi ; Xu, Jingjing ; Zhang, Dongdong ; Sun, Xu: Learning Unsupervised Word Mapping by Maximizing Mean Discrepancy. In: *arXiv:1811.00275 [cs]* (2018), November

[ZSMA17] Zogaan, Waleed ; Sharma, Palak ; Mirahkorli, Mehdi ; Arnaoudova, Venera: Datasets from Fifteen Years of Automated Requirements Traceability Research: Current State, Characteristics, and Quality. In: *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017. – ISSN 2332–6441, S. 110–121