Zehner, Fabian; Andersen, Nico

ReCo: Textantworten automatisch auswerten. Methodenworkshop

formal und inhaltlich überarbeitete Version der Originalveröffentlichung in:
formally and content revised edition of the original source in:
Zeitschrift für Soziologie der Erziehung und Sozialisation 40 (2020) 3, S. 334-339

Methodenworkshop

ReCo: Textantworten automatisch auswerten

Fabian Zehner[1,2] & Nico Andersen[1]

[1]DIPF | Leibniz-Institut für Bildungsforschung und Bildungsinformation

[2]Zentrum für Internationale Bildungsvergleichsstudien (ZIB) e.V.

Sprachliche Antworten spielen im sozialwissenschaftlichen Kontext vielerorts eine wichtige Rolle: zum Beispiel in Leistungstests der Bildungsforschung, in Prüfungen, Unterricht oder der psychologischen Diagnostik. Während geschlossene Antworten wie bei *Multiple Choice* einfach auszuwerten sind, können offene Antworten die Konstruktvalidität eines Instruments verbessern (z. B. Bridgeman, 1991; Ihme, Senkbeil, Goldhammer & Gerick, 2017; Lim, 2019). Um offene Textantworten manuell auszuwerten, werden üblicherweise Kodierer*innen aufwändig geschult. Ihr Einsatz birgt speziell in Large-Scale Assessments – wie dem *Programme for International Student Assessment* (PISA; OECD, 2019) – das Risiko fehlerhafter Kodierungen aufgrund der umfangreichen Datensätze, der Ermüdung, Inkonsistenz oder der subjektiven Perspektiven der Kodierer*innen. Natürliche Sprachverarbeitung und Maschinenlernen hingegen ermöglichen inzwischen den Einsatz von Software, um die manuelle Kodierung zu unterstützen, zu ergänzen oder zu ersetzen (einen Überblick liefern Mieskes & Pado, 2018).

Es gibt verschiedene Softwarepakete, um Antworten automatisch auszuwerten: z. B. *c-rater* für Kurztextantworten (Leacock & Chodorow, 2003) oder *Coh-Metrix* für Aufsätze (Graesser, McNamara & Kulikowich, 2011). Allerdings sind sie selten in Forschung oder Praxis einsetzbar, da sie zumeist reine Forschungsentwicklungen, kommerziell oder aber nur auf englischsprachliche Antworten anwendbar sind.

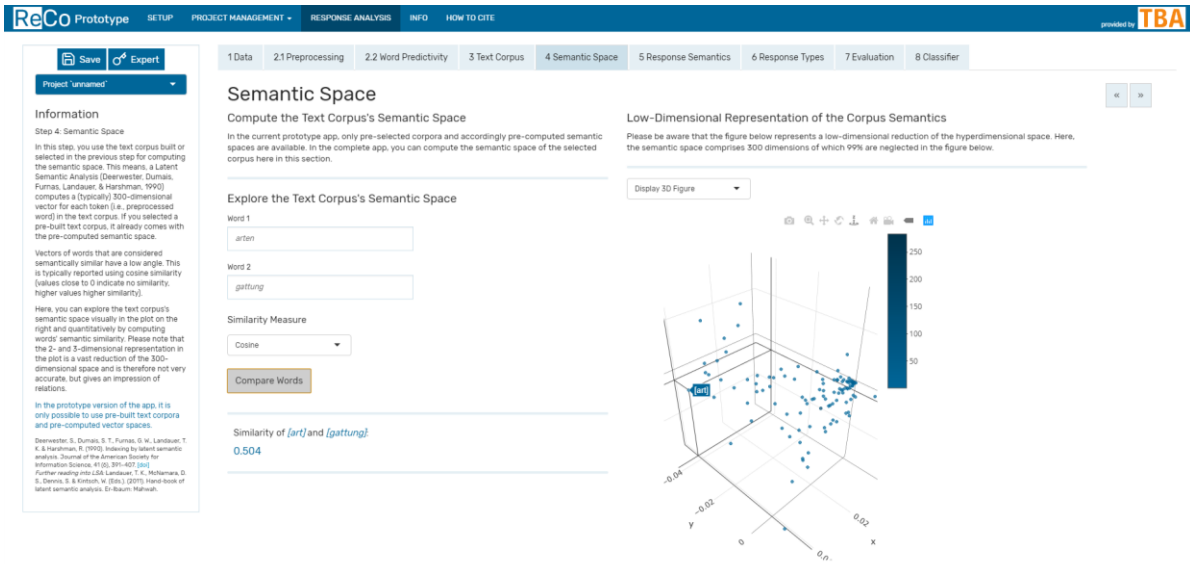

Abbildung 1: Blick auf die ReCo-Software (Automatic Text Response Coder)

Mit dem vorliegenden Beitrag wird erstmalig der Prototyp einer R- sowie Java-basierten und frei verfügbaren Software veröffentlicht (s. Abbildung 1), die für die Verwendung mit deutschen Textantworten evaluiert wurde und aktuell für weitere Sprachen weiter entwickelt wird: *ReCo* (*Automatic Text Response Coder;* Zehner, Sälzer & Goldhammer, 2016). ReCo ist auf Kurztextantworten spezialisiert und adressiert Semantik, weshalb auch von Inhaltsscoring die Rede ist.

Die hier vorgestellte Software enthält einen Demodatensatz, bei dem es wichtig ist, vorab anzumerken, dass dieser und die hier zitierten Beispielantworten lediglich eine sehr geringe Sprachvielfalt enthalten. Das liegt daran, dass dieser Datensatz auf empirischen Daten basiert und wegen deren Vertraulichkeit umfangreich manuell manipuliert wurde, was mit sprachlich komplexeren Items nicht möglich gewesen wäre. Die ReCo-Methodik selbst funktioniert hingegen auch bei komplexeren Antworten (so gezeigt etwa wenn Testpersonen schreiben sollen, was die Protagonistin einer Geschichte in einer bestimmten Situation sagen könnte; Zehner et al., 2016). ReCo wurde darüber hinaus bereits für weitere Anwendungszwecke eingesetzt, etwa zur Extraktion von Antwortmerkmalen, um den Mittelwertsunterschied in der Lesekompetenz zwischen Jungen und Mädchen zu beleuchten (Zehner, Goldhammer & Sälzer, 2018).

Dieser Beitrag skizziert kurz die ReCo-Methodik und stellt erstmals die Shiny-App vor, die automatisches Kodieren für eigene Daten flexibel anwendbar macht. Dafür wird skizziert, wie der aktuell verfügbare Prototyp installiert und auf einen Demodatensatz angewendet wird. Zuletzt gibt der Beitrag einen Ausblick, welche Funktionalitäten die App nach Verlassen der aktuellen Prototypenphase sowie in der langfristigen Entwicklung haben wird. Aktuelle Entwicklungen können auf der ReCo-Webseite verfolgt werden: www.reco.science

1 Von der Antwort zur Kodierung mittels Semantik

Der vorliegende Beitrag legt den Fokus auf die praktische Anwendung, weshalb die dahinterliegende Methodik lediglich grob skizziert wird. Details können Zehner et al. (2016) entnommen werden, wo auch die Performanz der Methode anhand deutscher PISA-Daten berichtet wird.

Beim automatischen Auswerten von Textantworten erhält die Software die Aufgabe, eine Antwort wie etwa die folgende auszuwerten: *„Die Passage bezieht sich auf die Tiere Delfine, Wildferde, Ratten und Paviane“*.[1] Die Software muss dann entscheiden, ob diese Antwort richtig ist (Scoring) oder – allgemeiner – welcher Kategorie die Antwort zuzuordnen ist (Kodierung). ReCo verwendet dazu ausschließlich die Semantik der in der Antwort enthaltenen Wörter. Techniken der natürlichen Sprachverarbeitung helfen dabei, die Antworten zu normalisieren (also redundante sprachliche Varianz zu reduzieren) und die Semantik zu quantifizieren. Mittels Maschinenlernen werden dann Kodiermodelle auf Basis von Trainingsdaten gebildet, um den Zusammenhang zwischen Semantik und Kodierung abzubilden.

In der ersten Phase (sog. Vorverarbeitung) trennt die Software die Antwort in Tokens (*[Die]*, *[Passage]*, …), korrigiert Rechtschreibfehler (*[Wildpferde]*), ersetzt Umlaute und Großschreibung (*[delfine]*) und entfernt Stoppwörter (~~*[die]*~~) sowie Affixe durch Stemming (*[tiere]*). Das

[1] Das zugrundeliegende Item erfragt, auf welche Tierarten sich ein am Textende befindlicher Satz bezieht. Es sind mind. zwei von vier Tierarten zu nennen, die in der Textmitte aufgelistet werden. Das Beispielitem und weitere Details zum Datensatz finden sich im Reiter *Info* der App.

Resultat für unser Beispiel wäre *[passag]*, *[bezieht]*, *[tier]*, *[delfin]*, *[wildpferd]*, *[ratten]*, *[pavian]*. Es kommt das reduktionistische Paradigma des *Bag of Words* zum Einsatz; das heißt, Reihenfolge und Syntax werden ignoriert. Weitere Schritte, wie zum Beispiel Kompositatrennung, oder das Ersetzen durch komplexere Techniken, wie zum Beispiel Lemmatisierung[2] statt Stemming, können durchgeführt werden.

Um die Semantik der so gewonnenen Tokens erkennen zu können, kommt ein zweites Paradigma zum Einsatz: das Kookkurrenzenparadigma. Dieses geht davon aus, dass Wörter, die in ähnlichen Kontexten auftauchen, sich auch semantisch ähnlich sind. Entsprechend können Nutzer*innen in ReCo einen für ein Item thematisch passenden Textkorpus aus der Wikipedia extrahieren und eine *Latente Semantische Analyse* durchführen, die genau diese beiden Paradigmen anwendet (LSA; Deerwester, Dumais, Furnas, Landauer & Harshman, 1990; guter Überblick in Landauer, McNamara, Dennis & Kintsch, 2011). Das Ergebnis der LSA ist eine Art Wörterbuch (semantischer Vektorraum), das jedem im Textkorpus enthaltenen Wort einen 300-dimensionalen Vektor zuordnet. Wörter, deren Vektoren in ähnliche Richtungen zeigen, werden als semantisch ähnlich angenommen. Technisch funktioniert die LSA ähnlich einer Faktorenanalyse, indem sie die Dimensionalität des Textkorpus über eine Singulärwertzerlegung reduziert. ReCo nutzt diesen semantischen Vektorraum, um die Semantik der Antworten zu quantifizieren. Dafür wird vereinfachend der Zentroidvektor aller Tokens einer Antwort verwendet.

Auf die Vektoren aller Antworten in einem Datensatz können anschließend statistische Analysen angewendet werden, um die Antworten zu kodieren. Da wie auf der Wortebene auch die Vektoren ähnlicher Antworten in ähnliche Richtungen weisen, kann in einem ersten nicht-überwachten Schritt ein Clustermodell gebildet werden, das Typen von Antworten ausweist. Durch Hinzuziehen von manuell erstellten Kodierungen kann im überwachten Maschinenlernen etwa ein *naïve Bayes*-Klassifikator trainiert werden, der einzelnen Antworttypen (Clustern) einen Code zuweist. Soll ReCo schließlich eine Antwort kodieren, identifiziert es den Antworttypen, der dieser Antwort am ähnlichsten ist, und weist den entsprechenden Code zu.

Da auf diese Weise ein Modell den Daten angepasst wird, muss zuletzt noch die Modellgültigkeit auf unabhängigen Daten geprüft werden. Üblicherweise kommt hier die Kreuzvalidierung zum Einsatz, bei der das Modell mehrfach an einem (variierenden) Großteil der Daten trainiert und an den restlichen Daten getestet wird.

2 Die ReCo-Software in R Shiny

ReCo ist eine Java-Software, die mit dem vorliegenden Beitrag mittels einer Shiny-App erstmals über R frei zugänglich gemacht wird. Die über die App verfügbaren Funktionalitäten sind im aktuellen Prototypen noch stark eingeschränkt. So können derzeit nur deutschsprachige Texte und vorberechnete semantische Räume verwendet werden, um die Methodik nachvollziehen zu können. Der Prototyp dient lediglich der Demonstration der allgemeinen methodischen Vorgehensweise.

Ein Vorteil der Nutzung einer R-basierten Oberfläche ist es, dass Nutzer*innen die mit ReCo berechneten Zwischenergebnisse auch direkt in R öffnen können, um von den in der Software

[2] Lemmatisierung führt Wörter auf ihre tatsächlichen Wortstämme zurück und ist insofern akkurater, allerdings nur für bestimmte Sprachen verfügbar und außerdem abhängig von der Sprachqualität, die in Assessments ohne individuelle Konsequenzen – wie PISA – sehr niedrig ist.

vorgesehenen Arbeitsschritten für die eigenen Zwecke abzuweichen. Die ReCo-Funktionen werden in späteren Versionen außerdem auch ohne die graphische Shiny-Oberfläche in R nutzbar sein.

Um die Shiny-App einmalig zu installieren, wird folgender Befehl in die Konsole eines aktuellen R (R Core Team, 2020) eingegeben:

```
source("https://www.reco.science/latest")
```

Damit wird das Paket shinyReCoR vom ReCo-Server heruntergeladen und installiert. Welche Befehle dabei zur Installation in R ausgeführt werden, kann eingesehen werden, indem dieselbe URL im Webbrowser aufgerufen wird. Anschließend muss bei Verwendung lediglich das Paket geladen und die Funktion recoApp() aufgerufen werden.

```
library(shinyReCoR)
recoApp()
```

Dieser Beitrag bezieht sich auf das Paket shinyReCoR in Version 0.1. Mit dem Prototypen werden auch ein Beispieldatensatz und ein vorberechneter semantischer Raum geliefert, die zur Demonstration verwendet werden. Informationen zum Beispieldatensatz (wie dem zugrundeliegenden Item) können dem Reiter *Info* in der App entnommen werden.

2.1 Schritt 1: Daten

Nachdem die App geöffnet ist und der Reiter *Response Analysis* ausgewählt ist, startet der Arbeitsprozess damit, die eigenen Textdaten zu importieren. Aktuell kann dabei ausschließlich eine CSV-Datei hochgeladen werden, die die Spalten *text* und *mancode* enthält. Erstere muss die Textantworten enthalten, zweitere den manuellen Code. Falls *mancode* nicht verfügbar ist, kann der Prozess nur bis Schritt *6, Response Types,* durchlaufen und keine automatischen Codes generiert werden. In späteren Versionen wird auch der Import von anderen (Datei-)Formaten mit beliebigen Spaltenbenennungen möglich sein. In der Vorschau sollte geprüft werden, dass alle Daten importiert wurden und auch Sonderzeichen wie Umlaute korrekt dargestellt werden.[3]

Um das im Beitrag beschriebene Beispiel durchlaufen zu können, wählen Sie den mitgelieferten Beispieldatensatz aus, der sich im selben Verzeichnis befindet wie das installierte Paket. Während der Installation gibt das Paket den Verzeichnispfad zum Beispieldatensatz auf die R-Konsole aus, um dessen Auffinden zu erleichtern. Um zum nächsten Schritt zu gelangen, wird der Pfeil rechts oben oder *2.1 Preprocessing* in der Menüleiste gewählt.

2.2 Schritt 2: Vorverarbeitung und Wordprädiktivität

Voraussetzung: Datenimport

Im Schritt *Preprocessing* werden die Techniken ausgewählt, die in der Vorverarbeitung der Textdaten zur Anwendung kommen. In der Vorschau rechts kann anhand eines Beispieltexts sowie zufällig ausgewählten Antworten aus dem Datensatz nachvollzogen werden, welchen Effekt die Verarbeitung auf die Texte hat. Hat man die gewünschten Techniken ausgewählt, muss noch der Button *Start Preprocessing* gewählt werden. Ein Ladebalken erscheint und die Ergebnisse können anschließend eingesehen werden. Für den Beispieldatensatz etwa kann

[3] Sollten Sonderzeichen nicht korrekt dargestellt werden, ist es in den meisten Fällen zu empfehlen, die Ausgangsdatei im UTF-8-Format zu speichern und das entsprechend im Auswahlmenü *Encoding* anzugeben.

rechts unter *Preview* der Eintrag *Random Response 4 (incorrect)* gewählt werden, um eine zufällige, falsche Antwort anzuzeigen. Das könnte etwa die Antwort *„das die Tiere öfters krank werden z.B. Ratten“* sein, die nach der Vorverarbeitung auf *„tier oft krank ratt“* reduziert ist.

In einer späteren App-Version werden weitere Wahlmöglichkeiten für die Vorverarbeitung zur Verfügung stehen; unter anderem wird es möglich sein, die Stoppwortliste anzupassen und eine automatische Rechtschreibkorrektur vorzunehmen.

Der Schritt *Word Predictivity* ist optional. Er gibt darüber Informationen, welche Wörter besonders häufig in den Antworten vorkommen und ob einzelne Wörter besonders prädiktiv dafür sind, ob eine Antwort einen bestimmten Code erhält (also z. B. richtig ist). Wörter, die hier als wichtig eingestuft werden, sollten auf jeden Fall reliabel im Textkorpus (vgl. Schritt 3) vertreten und auch nicht in der Stoppwortliste aufgenommen sein.

Abbildung 2 zeigt, wie die App mithilfe einer Wortwolke darstellt, welche Wörter als besonders prädiktiv für eine bestimmte Kodierung angenommen werden können. Wie leicht zu erkennen ist, weist die Verwendung der Wörter *delfine* und *ratten* auf eine richtige Antwort hin, während *flusspferd* eher mit falschen Antworten assoziiert ist.

Abbildung 2: Prädiktivität einzelner Wörter in den Antworten anhand einer Wortwolke (Schritt 2.2 der App; hier ein Ausschnitt)

2.3 Schritt 3: Textkorpus

Voraussetzung: keine

In einer späteren Version der App wird in diesem Schritt ein individuell auf das zu analysierende Item zugeschnittener Textkorpus zusammengestellt. Dafür kann der Titel eines Wikipedia-Artikels eingegeben werden, der für die Textantworten repräsentative Inhalte behandelt (vgl. Schritt 2.2). Die Software stellt dann automatisch einen Textkorpus aus den Wikipedia-Artikeln zusammen, die auf der gewählten Seite verlinkt sind oder die umgekehrt auf diese Seite verlinken. Je nach Einstellung werden auch solche Artikel hinzugenommen, die wiederum in diesen Artikeln verlinkt werden oder auf diese verlinken, usw.

Im aktuell vorliegenden Prototypen kann an dieser Stelle lediglich ein bereits mitgelieferter Korpus mit Namen *SozialeZootiere* ausgewählt werden, was die Anwendbarkeit des Prototypen auf andere Daten als den Demodatensatz einschränkt.

Der in diesem Schritt zusammengestellte Textkorpus dient als Basis für die Berechnung des semantischen Raums. Die Korpuswahl muss mithilfe des Buttons bestätigt werden. Nach dem Laden kann zum nächsten Schritt navigiert werden.

2.4 Schritt 4: Semantischer Raum

Voraussetzung: Wahl des Textkorpus

In diesem Schritt berechnet ReCo den semantischen Raum mithilfe einer Latenten Semantischen Analyse auf Basis des zuvor zusammengestellten oder gewählten Textkorpus. Bei nichtvorberechneten Räumen kann dieser Schritt je nach Korpusgröße und verfügbarer Rechenleistung länger dauern.

Im Falle unseres Beispiels ist der semantische Raum bereits vorberechnet und er kann unmittelbar exploriert werden. So können einerseits die Vektoren (vereinfacht dargestellt durch Punkte in einem auf drei Dimensionen reduzierten Raum) visuell in einer Abbildung untersucht werden. Entscheidend ist dabei die Richtung der Vektoren, die vom Koordinatenursprung aus zu betrachten sind. Wörter die so in eine ähnliche Richtung zeigen, werden als semantisch ähnlich angenommen. Mithilfe des Mauszeigers kann direkt mit der Abbildung interagiert werden. Außerdem ist die Auftrittshäufigkeit der im Korpus vorhandenen Wörter in den Antwortdaten farbkodiert. Quantitativ kann die Ähnlichkeit zwischen Wörtern in Form des Cosinus durch Eingabe in die Textfelder in dieser Oberfläche errechnet werden.

Während dieser Schritt ohne vorselektierte Textkorpora und entsprechend vorberechnete Räume entscheidend für die Berechnung des semantischen Raums ist, stellt er bei einem vorselektierten Korpus und vorberechneten Raum wie in unserem Beispiel lediglich die Möglichkeit zur Exploration des Raums dar.

Wichtig anzumerken ist es, dass die dreidimensionale Darstellung eine reduktionistische Darstellung des eigentlich 300-dimensionalen Raums ist. Die Abbildung zeigt außerdem nur jene Wörter, die auch im Antwortdatensatz vorkommen, falls dieser zuvor geladen wurde. Daher kann es teilweise zu unterschiedlichen Einschätzungen bezüglich der Ähnlichkeit von Wörtern anhand der visuellen Exploration und der (akkuraten) quantitativen Ähnlichkeit kommen.

In unserem beispielhaften semantischen Raum *SozialeZootiere* finden wir weit entfernt vom Rest der Datenpunkte, dass *[satz]* und *[text]* in ähnliche Richtungen weisen (s. Abbildung 3)[4]. Die Eingabe der Begriffe in die Textfelder zeigt, dass die Cosinus-Ähnlichkeit der beiden Begriffe in diesem Raum *cos* = 0,26 beträgt, *[satz]* und *[zoo]* hingegen bei *cos* = -0,01 liegt.

[4] Das Token [text] ist in der Abbildung aus technischen Gründen nicht dargestellt.

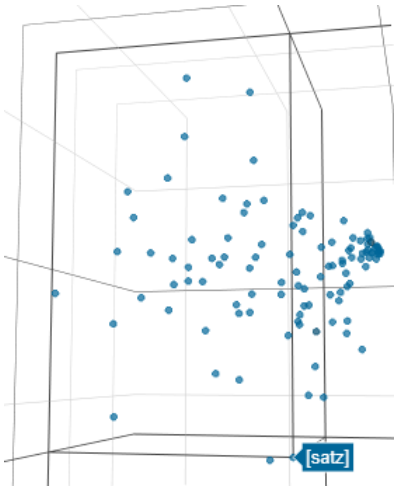

Abbildung 3: Wörter im semantischen Raum (Schritt 4 Semantic Space *in der App)*

2.5 Schritt 5: Antwortsemantik

Voraussetzungen: vorverarbeitete Antworten und semantischer Raum

Falls alle vorigen Schritte erfolgreich ausgeführt wurden, werden hier die Zentroidvektoren der Antworten berechnet, indem die im semantischen Raum hinterlegten Vektoren der einzelnen Wörter einer Antwort aggregiert werden. Analog zur Abbildung im vorangegangenen Schritt kann auch hier der semantische Raum exploriert werden. Allerdings stellen die Datenpunkte hier ganze Antworten dar, sodass die Ähnlichkeit zwischen diesen visuell eingeschätzt werden kann. Auch hier gilt, dass die dreidimensionale Darstellung eine markante Reduktion des 300-dimenionsalen Raums darstellt.

In unserem Beispiel sehen wir etwa, dass die Antworten „*Der Text bezieht sich auf Tiere wie: Delfine, Wildferde, Ratten und Pavians“* und „*Der Satz bezieht sich auf die Tiere: Delfine, Wildpferde, Ratten und Paviane“* als semantisch ähnlich angenommen werden.

2.6 Schritt 6: Antworttypen

Voraussetzung: Antwortvektoren

Wie im vorigen Schritt visualisiert, ordnen sich die Antworten im Raum gemäß ihrer Semantik an. Daher kann in diesem Schritt eine Clusteranalyse durchgeführt werden, um Antworttypen zu identifizieren. Die Oberfläche bietet verschiedene Clustermethoden und Distanzmaße an. In der aktuellen Prototypenfassung müssen sich die Nutzer*innen für eine bestimmte Clusteranzahl entscheiden. In einer späteren Version der App kann die Clusteranzahl gemäß der Evaluation des Klassifikators empirisch bestimmt werden.

Falls für den Antwortdatensatz manuelle Kodierungen vorliegen, kann den Antworttypen ein Code zugewiesen werden. Auf diese Weise kann die automatische Kodierung den Antworttypen identifizieren, der einer neuen Antwort am ähnlichsten ist, und dann den entsprechenden Code vergeben.

Falls keine manuellen Kodierungen für einen Antwortdatensatz vorliegen, kann ReCo bis zu dieser Stelle im Arbeitsprozess genutzt werden, um Textdaten in Typen zu unterteilen. Das kann etwa hilfreich sein, wenn Kodierrichtlinien erstellt werden sollen. Die Zuordnung von

Antworten zu Clustern, und falls verfügbar auch deren automatische Kodierung, kann in diesem Schritt exportiert werden.

Die Oberfläche zeigt die häufigsten Clusterzuordnungen mit ihren prototypischen Antworten sowie – falls manuelle Kodierungen verfügbar sind – die größeren Cluster mit hoher sowie niedriger Performanz an (d. h., mit stark ungleich- bzw. gleichverteilten Kodierverteilungen innerhalb der Cluster).

2.7 Schritt 7: Evaluation

Voraussetzung: Clustermodell und manuelle Kodierung

Sofern manuelle Kodierungen zum Datensatz vorliegen, kann die Übereinstimmung zwischen der automatischen und manuellen Kodierung anhand einer Kreuzvalidierung überprüft werden. Aktuell werden hierzu verschiedene Evaluationskoeffizienten wie die prozentuale Übereinstimmung (*Accuracy*) oder Kappa angezeigt.

2.8 Schritt 8: Klassifikator

Voraussetzung: Clustermodell und manuelle Kodierung

In diesem letzten Schritt kann der entwickelte Klassifikator exploriert werden. Durch Eingabe neuer Antworten im Textfeld kann geprüft werden, welche Kodierung der Klassifikator den neu eingegebenen Antworten zuweist. Es wird dabei auch angegeben, wie sicher diese Zuweisung gemäß den Trainingsdaten ist. Wird etwa die beispielhafte Antwort „*Es geht um Wildpferde und Ratten.“* eingegeben, klassifiziert die Software diese als richtig. Darüber hinaus werden auch diagnostische Detailinformationen angegeben, wie etwa, welche der eingegebenen Wörter nicht im Textkorpus enthalten sind, wie die Antwort verarbeitet wird und welche Antworten noch in dem Cluster enthalten sind, dem die neue Antwort zugeordnet wurde.

3 Ausblick zur Weiterentwicklung

Der beschriebene Prototyp ist naturgemäß lediglich die erste Entwicklungsstufe einer graphischen Oberfläche. Während die zugrundeliegende Java-Software bereits über viele weitere Funktionalitäten verfügt, werden in der hier beschriebenen App aktuell weitere grundlegende Funktionen ergänzt. So etwa die individuelle Zusammenstellung von Textkorpora und damit verknüpfte Berechnung des semantischen Raums. Auch allgemeinere Funktionen wie den Einsatz von parallelen Berechnungen, eine bessere Handhabung durch ein Projektmanagement, Itemgruppen sowie weitere Sprachen werden mittel- und langfristig verfügbar sein. Neue Entwicklungen werden auf der ReCo-Webseite (www.reco.science) bekannt gegeben und Interessierte können sich für einen Newsletter anmelden.

4 Literaturverzeichnis

Bridgeman, B. (1991). A comparison of open-ended and multiple-choice question formats for the quantitative section of the Graduate Record Examinations General Test. *ETS Research Report Series, 1991* (2), i-25. https://doi.org/10.1002/j.2333-8504.1991.tb01402.x

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science, 41* (6), 391–407. https://doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9

Graesser, A. C., McNamara, D. S. & Kulikowich, J. M. (2011). Coh-Metrix: Providing multilevel analyses of text characteristics. *Educational Researcher, 40* (5), 223–234. https://doi.org/10.3102/0013189X11413260

Ihme, J. M., Senkbeil, M., Goldhammer, F. & Gerick, J. (2017). Assessment of computer and information literacy in ICILS 2013. Do different item types measure the same construct? *European Educational Research Journal, 16* (6), 716–732. https://doi.org/10.1177/1474904117696095

Landauer, T. K., McNamara, D. S., Dennis, S. & Kintsch, W. (Eds.). (2011). *Handbook of latent semantic analysis*. Erlbaum: Mahwah.

Leacock, C. & Chodorow, M. (2003). C-rater. Automated scoring of short-answer questions. *Computers and the Humanities, 37,* 389–405.

Lim, H. (2019). Test format effects. A componential approach to second language reading. *Language Testing in Asia, 9:6*. https://doi.org/10.1186/s40468-019-0082-y

Mieskes, M. & Pado, U. (2018). Work smart - reducing effort in short-answer grading. In *Proceedings of the 7th Workshop on NLP for Computer Assisted Language Learning (NLP4CALL 2018) at SLTC, Stockholm, 7th November 2018* (S. 57–68). Linköping University Electronic Press, Linköpings universitet.

OECD. (2019). *PISA 2018 Results (volume I)* (PISA). Paris: OECD Publishing. https://doi.org/10.1787/5f07c754-en

R Core Team (2020). *R. A Language and Environment for Statistical Computing*. Vienna, Austria. Verfügbar unter https://www.R-project.org/

Zehner, F., Goldhammer, F. & Sälzer, C. (2018). Automatically analyzing text responses for exploring gender-specific cognitions in PISA reading. *Large-Scale Assessments in Education, 6:7*. https://doi.org/10.1186/s40536-018-0060-3

Zehner, F., Sälzer, C. & Goldhammer, F. (2016). Automatic coding of short text responses via clustering in educational assessment. *Educational and Psychological Measurement, 76* (2), 280–303. https://doi.org/10.1177/0013164415590022