Word-Embedding Benchmarking

Philipp Behnen[1], Felix Kruse[1], Jorge Marx Gómez[1]

[1]Carl von Ossietzky Universität Oldenburg, Department für Informatik,
Abt. Wirtschaftsinformatik / VLBA, Oldenburg, Deutschland
{philipp.behnen,felix.kruse,jorge.marx.gomez}@uol.de

Abstract. Die automatisierte Verarbeitung natürlicher Sprache bietet großes Potenzial, um neues Wissen aus unstrukturierten Daten zu gewinnen. Für die Bewältigung dieser Herausforderung haben sich in den vergangenen Jahren Word-Embedding-Verfahren bewährt, da diese semantische Ähnlichkeiten abbilden können. Durch die intensive Forschung entstehen stetig neue Verfahren. Je nach Aufgabe ist dabei aber nicht immer eindeutig, welches Verfahren für den Einzelfall empfehlenswert ist. Um dieses Problem zu adressieren, liefert dieser Beitrag einen Word-Embedding Benchmark, der die NLP-Tasks Ähnlichkeit, Analogie und (Multiclass-/Multilabel-) Klassifikation für einheitlich trainierte Word-Embeddings betrachtet und neben der Genauigkeit auch auf Kriterien wie die Trainingsdauer und die Hardwareressourcen eingeht. Zudem werden die Standardparametereinstellungen der Word-Embeddings modifiziert, um die Ergebnisse zu validieren. Die besten Ergebnisse lieferten die Word-Embeddings FastText und ELMo. Mit diesem Benchmark werden Data Scientisten bei der Auswahl eines Word-Embeddings für den Einsatz in entsprechenden NLP-Aufgaben unterstützt.

Keywords: Natural Language Processing, Machine Learning, Word Embedding, Character Embedding, Data Science

15th International Conference on Wirtschaftsinformatik,
March 08-11, 2020, Potsdam, Germany

https://doi.org/10.30844/wi_2020_c3-behnen

1 Einleitung

Die Big Data Ära sorgt für die vielen, umfangreichen und frei verfügbaren, heterogenen Datenquellen [1]. Der Wert eines Unternehmens wächst, wenn diese vorhandenen Datenquellen verknüpft und eine sehr gute Qualität aufweisen, sodass die Informationsqualität steigt und für optimierte Unternehmensentscheidungen sorgt [1]. Beispielsweise kann dies durch das Aufbereiten und Analysieren von Newsdaten passieren oder durch das Verknüpfen von mehreren Datenquellen durch Record Linkage Verfahren [1]. Für das Aufbereiten und Analysieren von Texten natürlicher Sprache (bspw. Newsdaten) oder das Vergleichen von Zeichenketten im Record Linkage können Word-Embeddings einen Vorteil generieren [2]. Um den Data Scientisten in den vielen Einsatzmöglichkeiten von Word-Embeddings bei der Auswahl eines geeigneten Word-Embedding Verfahrens zu unterstützen, wird in dieser Arbeit ein Benchmark erstellt.

Word-Embeddings gehören zu Sprachmodellierungsverfahren im Natural Language Processing [3]. Der Forschungsbereich Natural Language Processing untersucht, wie Computer Texte natürlicher Sprache interpretieren und in verschiedenen Anwendungen nutzen können [4]. Word-Embeddings bilden ein Wort oder eine Phrase aus einem gegebenen Vokabular auf einen Vektor mit reellen Zahlen ab [3]. Diese Vektoren umfassen in der Regel zwischen 100-1000 Dimensionen. Mit Hilfe der hochdimensionalen Vektoren sollen die semantischen Ähnlichkeiten der Wörter und Phrasen erhalten und abgebildet werden. Zu den Methoden zur Erzeugung der Word-Embeddings gehören beispielsweise Neuronale Netze oder die Wort Co-Occurence Matrix [3]. Für das Training der Word-Embeddings werden Datenquellen mit Texten natürlicher Sprache wie beispielsweise Wikipedia verwendet [4], [5]. Die Word-Embeddings können für verschiedene NLP-Tasks wie der Bestimmung von Ähnlichkeiten oder Analogien [5], Klassifikation [6], Part-of-Speech Tagging [7] oder Named Entity Recognition [8] genutzt werden. Sogenannte Out-of-Vocabulary (OOV) Wörter, die nicht in den Trainingsdaten vorkommen, können oftmals nur begrenzt abgebildet werden [9]. Ansätze wie Character-Embeddings können durch die Betrachtung von Zeichenketten innerhalb von Wörtern jedoch OOV Wörter abbilden, solange die Wortbestandteile bekannt sind [5], [6]. In diesem Paper werden Character-Embeddings und Word-Embeddings unter dem Begriff Word-Embeddings zusammengefasst. Word-Embeddings basieren als Machine-Lerning-Verfahren im Kern auf Neuronalen Netzen, dessen Architektur je nach Verfahren unterschiedlich ausgeprägt ist. Im Folgenden werden die dabei verwendeten Methoden Continuous Bag of Words (CBOW), Skip-Gram, Co-Occurence Matrix und Contextual Embedding näher beschrieben.

Die CBOW-Methode versucht die Wahrscheinlichkeit eines Wortes in einem Kontext vorherzusagen. Hierzu wird ein Kontext-Window bestimmt, durch das eine vorgegebene Anzahl an Wörtern oder Zeichen um das vorherzusagende Wort oder Zeichen berücksichtigt werden soll [10], [11].

Die Skip-Gram-Methode (SG) funktioniert umgekehrt zur CBOW Methode. Diese versucht von einem einzelnen Wort oder Zeichen auf die umliegenden Wörter und Zeichen, den Kontext, zu schließen [11], [12].

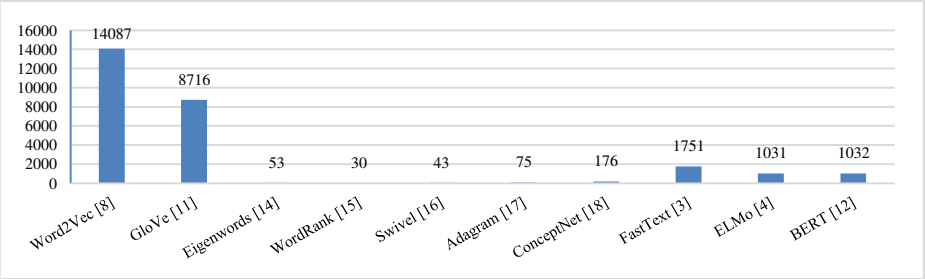
https://doi.org/10.30844/wi_2020_c3-behnen

Das Co-Ocurrence Verfahren basiert auf der Annahme, dass ähnliche Wörter nah beieinanderstehen. Für dieses Verfahren wird ein Kontext-Window definiert. Das Kontext-Window wird auf die sämtliche Worte des Textkorpus angewendet. Dabei wird in einer Matrix gespeichert, welche Wörter im Kontext des jeweiligen Wortes stehen und somit eine umfassende Statistik erstellt. Auf Basis dieser Statistik werden anschließend Wahrscheinlichkeiten berechnet und semantische Zusammenhänge gelernt [13].
Contextual Embeddings besitzen die Eigenschaft, dass jedes Wort in Abhängigkeit des gegebenen Kontext betrachtet wird und somit semantische (sowie kontextsensitive) Informationen berücksichtigt werden können [6], [14]. So kann das Wort *Bank* in den Sätzen *Ich sitze auf der Bank* und *Ich hole Geld von der Bank* dem Kontext entsprechend und auf die semantischen Unterschiede hin abgebildet werden. Voraussetzung zur Generierung dieser Vektoren ist, dass der jeweilige Input aus einem vollständigen Satz besteht. Die Bestimmung des Vektors wird zur Laufzeit durchgeführt [6].

1.1 Relevanz verschiedener Word-Embeddings

In Abbildung 1 ist die Relevanz bestehender Word-Embeddings gemessen an der Anzahl der Zitationen der zugehörigen Paper dargestellt. Es wurde jeweils das am häufigsten zitierte Paper für das jeweilige Word-Embedding ausgewählt. Basis für die Auswertung sind Google Scholar Daten vom 07.08.2019. Word2Vec mit 14087 [10], GloVe mit 8716 [13], FastText mit 1751 [15] und ELMo mit 1031 Zitationen [6] werden als die relevanten Word-Embeddings für den Benchmark ausgewählt. Das aktuellste Word-Embedding mit Namen BERT (1032 Zitationen), ist zum Zeitpunkt der Erstellung des Beitrages von keiner frei zugänglichen Bibliothek bereitgestellt worden und ist demnach nicht für ein Benchmarking geeignet.

Abbildung 1. Anzahl der Zitationen verschiedener Word-Embeddings



Word2Vec und FastText unterstützen CBOW und Skip-Gram, während GloVe eine Co-Occurence Matritx verwendet. ELMo ist das einzige kontextbasierte Word-Embedding der ausgewählten Verfahren. Word2Vec und GloVe bilden als klassische Word-Embeddings vollständige Wörter in Vektoren ab [13]. Im Gegensatz dazu arbeiten FastText und ELMo auf Zeichenbasis (Character-Embedding) und können

https://doi.org/10.30844/wi_2020_c3-behnen

deshalb auch das Problem von OOV-Words (herkömmlicher) Word-Embeddings umgehen. [5].

1.2 Related Work

In Tabelle 1 ist die Abgrenzung dieses Beitrags zu vorhandenen Word-Embedding Benchmark Papern dargestellt. Die Abgrenzung wird über die in den Papern betrachteten Embeddings, die NLP-Tasks, die dargestellten und modifizierten Parametereinstellungen und die betrachteten Effizienzkriterien vorgenommen.
Aufgrund der wenig vorhandenen Benchmarks besteht speziell für die Verfahren von ELMO und FastText ein Forschungsbedarf. ELMo wurde bisher nicht hinsichtlich der NLP-Tasks Ähnlichkeiten und Analogien untersucht. Der Einsatz von Word2Vec und GloVe in der Klassifikation beschränkt sich auf binäre Textklassifikation (Sentiment Analysis) [21] und der Klassifikation von einzelnen Wörtern in verschiedene Klassen [22]. Die Textklassifikation mit mehreren Klassen (Multiclass) in einem Word-Embedding Benchmark bedarf weiterer Forschung. Des Weiteren ist zu nennen, dass nicht immer alle Parameter gleich eingestellt werden (siehe Tabelle 1). Um sicherzustellen, ob die eingestellten Parameter für jedes getestete Word-Embedding aussagekräftige Ergebnisse liefern, sollte eine Validierung durchgeführt werden. In Bezug auf die Parametereinstellung und Validierung (z. B. mehrere Tests bzgl. verschiedener Iterationen) ist ebenfalls Forschungsbedarf zu vermerken. Dieser Beitrag grenzt sich außerdem von vergleichbaren Arbeiten ab, indem für jede Anpassung auch die Effizienz, gemessen an Trainingsdauer und RAM-Verbrauch während des Trainings, betrachtet wird.

Tabelle 1. Abgrenzung des Beitrages zu vorhandenen Word-Embedding-Benchmarks

Publikation	Embedding				NLP-Task			Parametereinstellungen					Effizienz	
	Word2Vec	GloVe	FastText	ELMo	Ähnlichkeit	Analogy	Klassifikation	Trainingsdaten	Dimensionalität	Iterationen	Context-Window	Validierung	Trainingsdauer	RAM-Verbrauch
Pennington et al. (2014) [13]	x	x			x	x		x	x					
Baroni et al. (2014) [22]	x				x	x	x	x	x		x	x		
Levy et al. (2015) [23]	x	x			x	x		x	x		x			
Ji et al. (2015) [17]	x	x			x	x		x	x		x			
Schnabel et al. (2015) [21]	x	x			x	x	x	x	x					
Ghannay et al. (2016) [7]	x	x			x	x		x	x		x		x	
Bojanowski et al. (2017) [5]	x		x		x	x		x	x	x	x			
Peters et al. (2018) [6]				x			x							
Dieser Beitrag	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**

https://doi.org/10.30844/wi_2020_c3-behnen

2 Erstellung und Durchführung des Benchmarks

2.1 Auswahl der Word-Embeddings für den Benchmark

Entsprechend der als relevant eingestuften Word-Embeddings (siehe Kapitel 1.1) werden in diesem Beitrag Word2Vec, GloVe, FastText und ELMo für den Benchmark ausgewählt. Für die Implementierungen werden die Python-Packages *glove-python*[1] für GloVe, *gensim* [24] für Word2Vec und FastText sowie *bilm-tf* / *allennlp* [25] für ELMo verwendet. Für eine möglichst hohe Transparenz und Aussagekraft werden alle Verfahren auf den gleichen Trainingsdaten trainiert, die im folgenden Kapitel beschrieben werden. Die einheitlichen Parametereinstellungen und -anpassungen werden in Kapitel 2.3 erläutert. Ferner ist zu nennen, dass für die Auswertungen mit ELMo das namensgebende ELMo-Layer (First Representation Layer) zur Bestimmung von Vektoren verwendet wird. Die Auswahl des Layers kann je nach Aufgabentyp einen Einfluss auf das Ergebnis haben [6]. ELMo ist das einzige Benchmark-Objekt, das mehrere Vektor-Repräsentationen unterstützt, weshalb die Wahl des Layers bei den anderen Architekturen entfällt.

2.2 Auswahl der Trainingsdaten

Als Datenquelle wird ein Dump der englischen Wikipedia vom 20.03.2019 verwendet. Damit wird gewährleistet, dass die Ergebnisse des Benchmarks reproduzierbar sind. Die Domänenunabhängigkeit sowie die freie Verfügbarkeit der Daten stellen weitere Argumente für die Wahl der englischen Wikipedia als Trainingsdatensatz dar. Bevor der Trainingsdatensatz genutzt werden kann muss dieser bereinigt werden. Dabei wird der Text aus dem vorhandenen XML-Schema extrahiert und es werden die MediaWiki Markup Language Tags, die HTML-Tags, die Satzzeichen, die Sonderzeichen und die Stop-Words entfernt. Zudem werden Akzente ersetzt (z. B. è zu e) und Zahlen durch das Tag *<NUM>* ausgetauscht. Eine Case-Normalisierung wird ebenfalls durchgeführt. Auf Verfahren wie Stemming oder Lemmatization wird aufgrund der Notwendigkeit von Wortendungen für die verwendeten NLP-Tasks verzichtet. Der resultierende Textkorpus ist ca. 9,3 GB groß und enthält 1,2 Mrd. Wörter.

2.3 Parametereinstellungen der Word-Embedding Verfahren

Das Training wird auf einem Rechner im AI-Lab der Abteilung VLBA der Universität Oldenburg ausgeführt, der mit 32 GB RAM, einer i7 8700k CPU und einer GTX 1080Ti GPU ausgestattet ist. Die gemeinsamen Parameter aller ausgewählten Word-Embeddings[2] werden in Tabelle 2 dargestellt. Alle weiteren

[1] https://github.com/maciejkula/glove-python
[2] Ausnahme: ELMo besitzt kein Context-Window

https://doi.org/10.30844/wi_2020_c3-behnen

Einstellungen werden nicht berücksichtigt und auf den Empfehlungswerten der jeweiligen Entwickler belassen.

Tabelle 2. Parametereinstellungen der Benchmark-Objekte. Standardparameter sind fett dargestellt - (S) = Standardparameter

Korpusgröße	Dimension	Iterationen	Context-Window
0,1 GB	100	1	5 (S)
1 GB	300 (S)	3 (S)	10
9,3 GB (S)	1000	10	

Für den ersten Benchmark werden alle Word-Embeddings mit den Standardparametereinstellungen trainiert (siehe Tabelle 2). Im zweiten Teil des Benchmarks werden die Parameterwerte in den aufgeführten Wertebereichen variiert (siehe Tabelle 2), um die Standardparametereinstellungen zu validieren. Dazu wird jeweils ein Parameter variiert und die übrigen werden auf den Standardparametereinstellungen belassen. Durch die Parametervariationen wird sichergestellt, dass die bestmöglichen Ergebnisse je Word-Embedding für den Benchmark herangezogen werden.
Neben selbst trainierten Word-Embeddings werden auch vortrainierte Modelle im Benchmark betrachtet. Die ausgewählten vortrainierten Word-Embeddings sind in Tabelle 3 aufgeführt.

Tabelle 3. Eigenschaften vortrainierte Word-Embeddings als Referenz

Embedding	Dim.	Wörter	Architektur	Quelle
Word2Vec	300	100 Mrd.	Skip-Gram	https://code.google.com/archive/p/word2vec/
GloVe	300	840 Mrd.		https://nlp.stanford.edu/projects/glove/
FastText	300	600 Mrd.	Skip-Gram	https://fasttext.cc/docs/en/english-vectors.html
ELMo	512	5,5 Mrd.		https://allennlp.org/elmo

2.4 Benchmark-Kriterien und Benchmark-Sets

Als Bewertungskriterien werden die (1) Trainingsdauer, (2) der RAM-Verbrauch während des Trainings sowie die (3) Genauigkeit für die NLP-Tasks (3a) Ähnlichkeiten, (3b) Analogien und (3c) Klassifikation ausgewählt. Die Auswahl der NLP-Tasks wird mit den identifizierten Forschungslücken aus vorhandenen Benchmarks begründet.
Die Trainingsdauer und der RAM-Verbrauch werden beim Training der verschiedenen Word-Embeddings gemessen. Die Genauigkeit wird anhand von bestehenden Benchmark-Sets ermittelt und für jeden Task einzeln bestimmt. Im Folgenden werden die verwendeten Benchmark-Sets zur Messung der Genauigkeit beschrieben.

https://doi.org/10.30844/wi_2020_c3-behnen

Ähnlichkeiten Für die Ähnlichkeits-Aufgabe wurden die Benchmark-Sets MEN Test Collection [26] und WS353 [9] ausgewählt. Beide Sets sind fester Bestandteil vergleichbarer Benchmarks [17], [12], [23], [13] und können demnach als relevante Benchmark-Sets eingestuft werden. Es handelt sich um Listen von Wortpaaren, die mit einem numerischen Ähnlichkeitswert versehen sind. Dieses Label soll möglichst genau bestimmt werden. Aus beiden Benchmark-Sets wird ein einheitliches, kombiniertes Set erstellt. Die Bestimmung der Genauigkeit erfolgt nach dem relativen Unterschied der Vorhersage zum vorhandenen Label.

Analogien Bei dem Analogie-Task geht es darum, ausgehend von einer Relation eine zweite zu komplettieren [11]. Dabei soll eine passende Antwort auf eine Frage wie *deutschland ist zu berlin wie frankreich zu X* gefunden werden. Vorhandene Benchmarks wie [7], [13], [17] oder [21-23] verwenden dafür das Google Analogy Test Set [27]. In diesem Benchmark wird stattdessen das BATS [28] verwendet, da dieses durch eine höhere Anzahl an Kategorien und der damit einhergehende Komplexitätssteigerung die Möglichkeit bietet, aussagekräftigere Ergebnisse für die Praxis zu liefern. Während das Google Analogy Test Set zu über 56 % aus Land-Hauptstadt-Relationen (z. B. *deutschland-berlin*) besteht [28], enthält das BATS u. a. auch ableitende (z. B. *argue-argument*) oder Teil-Ganzes-Relationen (z. B. *car-engine*), auch Meronyme genannt [28]. Die Genauigkeit wird mit den Berechnungsvorschriften 3CosAdd oder 3CosMul ausgewertet, wobei ein geringer (ca. 1 %) Vorteil zugunsten 3CosMul erwartet wird [23], [29]. Für die Berechnung der Ergebnisse in diesem Beitrag wurden sowohl das 3CosAdd-, als auch das 3CosMul-Verfahren verwendet. Da während allen Tests keine hinreichenden Unterschiede festgestellt werden konnten, wird im Folgenden die jeweils bessere Genauigkeit dargestellt.

Klassifikation Für die Auswertung der Klassifikationsaufgabe wird die Reuters 21578 Text Categorization Test Collection [30] verwendet, bei der Nachrichtendaten in 90 verschiedene Klassen eingeordnet werden. Eine Besonderheit des Sets ist, dass es sich um ein Multiclass und Multilabel Set für Klassifikation handelt. Das bedeutet, dass es mehr als zwei Klassen gibt und ein Text ggf. mehreren Klassen zugeordnet werden kann [31]. Die Nachrichtentexte des Benchmark-Sets müssen für die weitere Verarbeitung aufbereitet werden. Dafür wird die bereits beschriebene Bereinigungsstrategie angewandt. Der Split zwischen Trainings- und Validationsdaten beträgt 70 % zu 30 %. Mit den trainierten Word-Embeddings wird für jeden der zu klassifizierenden Texte je ein Vektor bestimmt, indem von allen enthaltenen Wort-Vektoren ein Mittelwert gebildet wird. ELMo liefert für einen Text als Input direkt einen Vektor [6], sodass der zusätzliche Schritt für ELMo ausgelassen wird. Der Vektor wird als Input für einen Log-Linearen Klassifikator aus dem *sklearn* Package[3] für Python benutzt. Die Genauigkeit des Benchmark-Sets berechnet sich aus dem gewichteten Mittel des F1-Scores aller verwendeten Klassen.

[3] https://scikit-learn.org, Es wurden Standardeinstellungen verwendet.

https://doi.org/10.30844/wi_2020_c3-behnen

3 Evaluation der Ergebnisse

3.1 Ergebnisse der Word-Embeddings mit Standardparametern

Die Ergebnisse des durchgeführten Benchmarks für die selbsttrainierten (self) Word-Embeddings mit Standardparametereinstellungen sind im direkten Vergleich zu den vortrainierten (pre) Word-Embeddings in Tabelle 4 dargestellt.

Tabelle 4. Benchmark-Ergebnisse der ausgewählten Word-Embeddings mit Standardparametern

Word-Embedding	Zeit (h)	RAM (GB)	Ähnlichkeit (%)		Analogie (%)		Klassifikation (%)	
			Self	Pre	Self	Pre	Self	Pre
W2V SG	2,33	2,1	97,06	96,06	27,40	26,03	79,97	79,02
W2V CB	0,9	2,1	96,92		27,31		80,02	
GloVe	2,48	17,5	98,78	99,30	18,62	30,97	78,87	79,21
FT SG	7,5	8,3	99,53	99,67	28,31	37,16	80,14	80,04
FT CB	3,3	8,3	97,39		37,07		80,19	
ELMo	165	-	99,61	99,55	29,36	2,51	78,78	80,29

Trainingsdauer Die geringste Trainingsdauer benötigte Word2Vec CBOW mit 0,9h. Die längste Trainingsdauer benötigte ELMo mit 165h. ELMo wird als einziges von den verwendeten Word-Embeddings auf der GPU statt auf CPU trainiert und benötigt dennoch die längste Trainingsdauer. Das Skip-Gram Verfahren benötigt für das Trainieren von Word2Vec und FastText mehr Zeit als das CBOW Verfahren.

RAM-Verbrauch Word2Vec belegt während des Trainings mit 2,1 GB am wenigsten RAM. Für das Training von ELMo ist es notwendig, die Daten in kleinere Teile (Batches) aufzuteilen und sequenziell zu trainieren. Der RAM-Verbrauch von ELMo kann daher nicht unter den gleichen Bedingungen wie für die anderen Word-Embedding-Verfahren gemessen werden. Für ELMo wird keine konkrete Zahl erfasst, da der RAM-Verbrauch durch die Batch-Größe flexibel angepasst werden kann. FastText benötigt 8,3 GB und GloVe 17,5 GB RAM. Innerhalb von Word2Vec und FastText sind zwischen CBOW und Skip-Gram keine Unterschiede feststellbar.

Ähnlichkeit Unter den selbst trainierten Word-Embeddings kann ELMo mit 99,61 % die höchste Genauigkeit für die Bestimmung von Ähnlichkeiten erzielen. Damit ist dieses um 0,06 Prozentpunkte genauer als das vortrainierte ELMo. Für FastText Skip-Gram kann mit 99,53 % die zweithöchste Genauigkeit der selbst trainierten Word-Embeddings festgestellt werden. Die Genauigkeit mit Skip-Gram ist für FastText im direkten Vergleich um 2,14 Prozentpunkte höher als für CBOW. Die geringste Genauigkeit der selbst trainierten Word-Embeddings liegt für Word2Vec CBOW mit 96,92 % insgesamt 2,69 Prozentpunkte unter dem Wert von ELMo. Verglichen mit

https://doi.org/10.30844/wi_2020_c3-behnen

den vortrainierten Word-Embeddings ist die Genauigkeit der selbst trainierten Verfahren im Schnitt 0,4 Prozentpunkte geringer.

Analogie Die höchste Analogie-Genauigkeit erreicht FastText Skip-Gram (vortrainiert) mit 37,16 %. Das selbst trainierte FastText CBOW liegt mit 37,07 % auf Platz Zwei aller getesteten Verfahren. Der maximale Unterschied zwischen den Benchmark-Objekten beträgt 34,65 %, wobei ELMo (vortrainiert) mit 2,51 % die geringste Genauigkeit erzielt. Für das selbst trainierte ELMo sind 29,36 % Genauigkeit festzustellen. Der Unterschied zwischen dem vortrainierten und selbsttrainierten ELMo von 26,85 Prozentpunkten kann durch eine abweichende Bereinigungsstrategie erklärt werden. Für GloVe beträgt der Unterschied zwischen selbst und vortrainierten Modellen 12,35 Prozentpunkte zugunsten des vortrainierten GloVe. Die erhöhte Genauigkeit kann durch die erhöhte Menge an Trainingsdaten (siehe Tabelle 3) erklärt werden. GloVe scheint damit stärker von der Korpusgröße abhängig zu sein als die anderen Word-Embeddings.

Klassifikation Für den Klassifikations-Task erreicht das vortrainierte ELMo mit 80,29 % den höchsten F1-Score der getesteten Verfahren. Selbst trainiert erzielt ELMo die geringste Genauigkeit von 78,78 %. Der Unterschied zwischen minimaler und maximaler Genauigkeit beträgt demnach 1,51 Prozentpunkte. Im direkten Vergleich erreichen die selbst trainierten Word-Embeddings im Schnitt um 0,02 Prozentpunkte bessere Ergebnisse als die vortrainierten Gegenstücke. Die Beobachtungen können nach weiteren Tests mit einem zweiten Klassifikator – dem Multilayer-Perceptron (MLP) aus dem *sklearn* Package[4] – bestätigt werden. Jedes Benchmark-Objekt weist mit dem MLP-Klassifikator eine um ca. 2 Prozentpunkte erhöhte Genauigkeit auf. Das Verhältnis zwischen den Verfahren bleibt gleich. Die Auswahl des Klassifikators hat demnach einen ebenfalls entscheidenden Einfluss auf das Ergebnis wie die Wahl des Word-Embeddings.

3.2 Ergebnisse mit Parameteranpassungen

Um die Belastbarkeit des Benchmarks in Bezug auf die Auswahl der Parameter zu gewährleisten und gleichzeitig ein möglichst transparentes Ergebnis zu liefern, werden weitere Tests mit Änderungen an den eingestellten Parametern durchgeführt (siehe Kapitel 2.3).

Anpassen der Context-Window-Size In Tabelle 5 sind die Veränderungen der Benchmark-Performance zu sehen, die durch das Verändern der Context-Window Size von 5 auf 10 für die Word-Embeddings entstehen. Alle Benchmark-Objekte weisen eine erhöhte Trainingsdauer auf. GloVe benötigt 31,43 % mehr RAM während des Trainings. Die übrigen Word-Embeddings benötigen keinen zusätzlichen RAM. Die Genauigkeit der Ähnlichkeitsaufgabe schwankt von einer Verschlechterung um -

[4] https://scikit-learn.org, Es wurden Standardeinstellungen verwendet.

https://doi.org/10.30844/wi_2020_c3-behnen

0,51% bis zu einer Verbesserung von 2,26%. Die Genauigkeit der Analogieaufgabe schwankt von einer Verschlechterung um -3,82% bis zu einer Verbesserung von +1,03%. Die Genauigkeit der Klassifikationsaufgabe ist bei allen Word-Embeddings gestiegen. Die Verbesserung liegt zwischen 0,12% und 1,51%.

Tabelle 5. Absolute Differenz der Performance bei Vergrößerung des Context-Windows von 5 auf 10 in %

Embedding	Zeit	RAM	Ähnlichkeit	Analogie	Klassifikation
W2V SG	+58,96	+0	+2,26	-3,82	+0,54
W2V CB	+57,41	+0	-0,14	-0,84	+0,12
GloVe	+70,47	+31,43	-0,51	+1,03	+1,51
FT SG	+12,89	+0	+0,12	-1,97	+0,71
FT CB	+109,2	+0	-0,45	+0,26	+0,46

Anpassen der Trainingsdaten-Menge In Tabelle 6 sind die Veränderungen der Benchmark-Performance zu sehen, die durch das Verändern der Trainingsdatenmenge von 9,3 GB auf 0,1 GB und 1 GB für die Word-Embeddings entstehen. Mit der Reduktion der Trainingdatenmenge auf 0,1 GB und 1 GB benötigen alle Word-Embeddings zwischen 70 % - 99,9 % weniger Zeit und RAM für das Training. Für die NLP-Tasks Ähnlichkeit und Klassifikation ist die Genauigkeit zwischen -8,15 % und -0,04 % geringer bei 0,1 GB Trainingsdaten. Bei einer Änderung auf 1 GB Trainingsdaten liegt die Abweichung der Genauigkeit zwischen -1,09 % und +2 %. Für die NLP-Task Analogie sind die Abweichungen der Genauigkeit höher. Für 0,1 GB liegt die Abweichung der Genauigkeit zwischen -18,02 % und +0,59 %. Für 1 GB liegt die Abweichung zwischen -8,09 % und + 1,91 %. Bei allen Word-Embeddings ist zu erkennen, dass mit 1 GB Trainingsdaten Genauigkeiten erreicht werden, die nahe denen der Word-Embeddings liegen, die mit 9,3 GB trainiert worden sind.

Tabelle 6. Absolute Differenz der Performance für verschiedene GB an Trainingsdaten in %

Embedding	Zeit		RAM		Ähnlichkeit		Analogie		Klassifikation	
	0,1	1	0,1	1	0,1	1	0,1	1	0,1	1
W2V SG	-99,25	-82,84	-90,48	-71,43	-7,29	+2	-15,74	-4,78	-0,48	+0,26
W2V CB	-98,15	-85,19	-90,48	-71,43	-8,15	-0,16	-18,02	-5,62	-3,99	-1,09
GloVe	-97,99	-91,28	-92,57	-80,57	-0,66	-0,89	-17,69	-8,09	-6,86	-0,59
FT SG	-99,11	-91,78	-90,36	-75,90	-0,04	-0,05	+0,59	+1,91	-0,28	+0,21
FT CB	-97,98	-79,29	-90,36	-75,90	-0,97	-0,6	-6,68	+0,2	-3,85	-0,21
ELMo	-98,51	-84,75			-0,04	-0,06	-2,09	-0,01	-2,13	+0,21

Anpassen der Dimensionsgröße In Tabelle 7 sind die Veränderungen der Benchmark-Performance zu sehen, die durch das Verändern der Dimensionsgröße von 300 auf 100 und 1000 Dimensionen für die Word-Embeddings entstehen. Bei einer Veränderung auf 100 Dimensionen reduziert sich die Trainingszeit bei allen Word-Embeddings. Die Veränderung liegt zwischen -13,81 % und -66,66 %. Bei einer Veränderung auf 1000 Dimensionen steigt die Trainingsdauer zwischen +15,7

https://doi.org/10.30844/wi_2020_c3-behnen

% und +385,3 %. Die Veränderung des RAM-Verbrauchs liegt zwischen -67,47 % und +0 % für 100 Dimensionen. Bei 1000 Dimensionen liegt die Veränderung zwischen +47,4 % und 217 %. Für die NLP-Task Ähnlichkeit liegt die Veränderung der Genauigkeit für 100 und 1000 Dimensionen zwischen -1,03 % und +1,31 %. Die Veränderung der Genauigkeit für die NLP-Task Analogie liegt zwischen -5,69 % und – 1,81 % für 100 Dimensionen und zwischen -1,99 % und +1,89% für 1000 Dimensionen. Bei der Klassifikation verändert sich die Genauigkeit zwischen -4,35 % und -2,54 % bei 100 Dimensionen und zwischen +1,47 % und 3,87 % bei 1000 Dimensionen. Sowohl für die Analogie als auch die Klassifikations Task steigt die Genauigkeit bei höherer Dimensionalität für alle Word-Embeddings.

Tabelle 7. Absolute Differenz der Performance für verschiedene Dimensionsgrößen in %

Embedding	Zeit		RAM		Ähnlichkeit		Analogie		Klassifikation	
	100	1000	100	1000	100	1000	100	1000	100	1000
W2V SG	-14,71	+385,3	+0	+152	+0,87	+1,31	-4,4	-1,99	-3,57	+3,87
W2V CB	-13,81	+196,3	+0	+152	-1,03	-1,21	-4,37	+1,89	-3,93	+2,96
GloVe	-14,09	+40,9	-8,0	+47,4	+0,43	-0,37	-4,09	+0,08	-3,07	+2,47
FT SG	-42,22	+124,0	-67,47	+217	+0,15	-0,35	-5,69	-0,91	-2,72	+3,73
FT CB	-35,12	+214,6	-67,47	+209	+0,8	-0,82	-2,89	+0,45	-4,35	+3,13
ELMo	-66,66	+15,7			-0,1	-0,06	-1,81	+1,02	-2,54	+1,47

Anpassen der Iterationen In Tabelle 8 sind die Veränderungen der Benchmark-Performance zu sehen, die durch das Verändern der Iterationen von 3 auf 1 und 10 für die Word-Embeddings entstehen. Bei 1 Iteration reduziert sich die Trainingszeit für alle Word-Embeddings zwischen -81,56 % und -38,89 %. Bei 10 Iterationen steigt die Trainingszeit bei allen Word-Embeddings zwischen +47,7 % und 496,5 %. Der RAM-Verbrauch verändert sich zwischen -8 % und +23,81 % für die Veränderung auf 1 Iteration. Bei der Veränderung auf 10 Iterationen verändert dieser sich bei allen Word-Embeddings gegenüber 3 Iterationen nicht. Die Veränderung der Genauigkeit für die Ähnlichkeits Task liegt zwischen -2,32% und +2,01 % bei Veränderung der Iterationen auf 1 und 10. Für die Analogie Task liegt die Veränderung der Genauigkeit zwischen -7,8% und +2,11 % bei Veränderung der Iterationen auf 1 und 10. Für die Klassifikations-Tasks liegt die Veränderung zwischen -0,12 % und + 1,6 % bei Veränderung der Iterationen auf 1 und 10.

Tabelle 8. Absolute Differenz der Performance für verschiedene Iterationen in %

Embedding	Zeit		RAM		Ähnlichkeit		Analogie		Klassifikation	
	1	10	1	10	1	10	1	10	1	10
W2V SG	-56,73	+496,5	+23,81	+0	-2,32	+2,01	-2,98	-0,66	+0,14	+0,07
W2V CB	-38,89	+377,8	+23,81	+0	+0,21	+0,1	-2,37	+0,89	-0,04	-0,06
GloVe	-57,72	+47,7	-8	+0	+0,46	-0,5	-7,8	+2,11	+0,14	+1,6
FT SG	-81,56	+106,0	-4,82	+0	+0,11	+0,03	+0,45	+0,77	+0,55	-0,1
FT CB	-58,08	+430,8	-4,82	+0	+0,46	-0,63	-0,44	+0,73	-0,12	+0,63
ELMo	-66,36	+233,3			+0,01	-0,05	-1,18	+0,99	-0,08	+1,29

https://doi.org/10.30844/wi_2020_c3-behnen

3.3 Diskussion der Benchmark Ergebnisse

Die Benchmark Ergebnisse zeigen, dass die selbsttrainierten Word-Embeddings von der Genaugikeit sehr nah an den vortrainierten Word-Embeddings liegen. Dieser Benchmark hilft bei der Auswahl eines Word-Embeddings für die NLP-Tasks Ähnlichkeit, Analogie und Klassifikation. Für die Ähnlichkeit und Analogie wird das vortrainierte FastText Skip-Gram empfohlen, da es die höchste Genauigkeit liefert und nicht mehr trainiert werden muss. Aus denselben Gründen wird für die Klassifikation das vortrainierte ELMo empfohlen. Sowohl FastText als auch ELMo können das Problem der OOV Wörter bewältigen. Dies ist ein Vorteil, wenn domänenspezifische Wörter, die nicht im Trainingsdatensatz repräsentiert sind, in einen Vektor gewandelt werden sollen. Reicht die Genauigkeit der zur Verfügung stehenden Word-Embeddings für eine spezifische Aufgabe nicht aus, kann ein Word-Embedding mit domänenspezifischen Trainingsdaten erstellt werden. Mit den Ergebnissen aus Tabelle 4 ist Word2Vec Skip-Gram oder CBOW zu empfehlen, da die Trainingsdauer gering und die Genauigkeit hoch ist. Wenn zusätzlich die Ergebnisse der Parameteranpassungen berücksichtigt werden, sollten die domänenspezifischen Word-Embeddings mit FastText Skip-Gram oder CBOW auf der Basis von 1 GB Trainingsdaten erstellt werden. Bei 1 GB Trainingsdaten reduziert sich die Trainingsdauer bei FastText Skip-Gram um 91,78% und bei CBOW um 79,29% und ist somit genauso gering wie bei Word2Vec. Die Genauigkeit für die FastText-Embeddings bei 1 GB Trainingsdaten bleibt ebenfalls konstant und der Vorteil der OOV Wörter wird ebenfalls genutzt. ELMo sollte bei der Auswahl für das Training mit domänenspezifischen Daten ebenfalls beachtet werden, da es ein Fine-Tuning unterstützt. Durch Fine-Tuning kann ein bereits trainiertes ELMo Word-Embedding mit neuen Daten nachtrainiert werden. Als Nachteil ist die hohe Trainingsdauer von ELMo zu nennen.

4 Zusammenfassung und Ausblick

In diesem Beitrag wurde ein Word-Embedding Benchmark durchgeführt, der sich von bisherigen Benchmarks in folgenden Kriterien unterscheidet: (1) die Anzahl der verschiedenen Word-Embeddings, (2) die NLP-Tasks, (3) die Anpassung der Parameter beim Training der Word-Embeddings und (4) die Betrachtung der Effizienz in Form von Trainingsdauer und RAM-Verbrauch. Anhand der Anzahl von Zitationen wurden für den Benchmark die Word-Embeddings Word2Vec, GloVe, FastText und ELMo ausgewählt. Für jedes der vier Word-Embeddings wurde ein vortrainiertes Model für den Benchmark ausgewählt. Zusätzlich wurden eigene Modelle für die vier Word-Embeddings trainiert. Für das Training dieser wurde die englische Wikipedia ausgewählt. Die Parameter Dimensionsanzahl, die Anzahl an Iterationen und die Größe des Context-Window können bei den ausgewählten Word-Embeddings angepasst werden. Diese Parameter wurden modifiziert und die Ergebnisse fließen ebenfalls in den Benchmark ein. Die Word-Embeddings wurden durch die Benchmark-Kriterien (1) Trainingsdauer, (2) RAM-Verbrauch, (3) Genauigkeit NLP-Task Ähnlichkeit, (4) Genauigkeit NLP-Task Analogie und (5)

https://doi.org/10.30844/wi_2020_c3-behnen

Genauigkeit NLP-Task Klassifikation (für Multiclass-/Multilabel-Tasks) bewertet. Für die Ähnlichkeit wurden die Benchmark-Sets *MEN Test Collection* und *WS353* verwendet. Für die Analogie wurde das Benchmark-Set *BATS* verwendet und für die Klassifikation das Benchmark-Set *Reuters 21578 Text Categorization Test Collection*. Die beste Genauigkeit für die NLP-Tasks Ähnlichkeit und Analogie liefert das vortrainierte FastText Skip-Gram. Für die NLP Task Analogie liefert ELMo die höchste Genauigkeit. Beide Word-Embeddings können mit Out-of-Vocabulary Wörtern umgehen. Dies ist ein weiterer Vorteil gegenüber Word2Vec und GloVe. Für das trainieren individueller Word-Embeddings eignet sich FastText ebenfalls. Bereits mit einer Trainingsdatenmenge von 1 GB werden die Genauigkeitswerte der vortrainierten Word-Embeddings für die getesteten NLP-Tasks erreicht und die Trainingsdauer und der RAM-Verbrauch sind verhältnismäßig gering. Hingegen konnte für GloVe, verglichen mit den anderen Verfahren, ein relativ hoher Unterschied zwischen den Ergebnissen des selbst-trainierten und vortrainierten Models festgestellt werden. Die Unterschiede in der Benchmark-Performance zwischen den NLP-Tasks zeigen außerdem, dass nur in geringem Maße eine Aussage für andere Tasks wie z. B. Part-of-Speech-Tagging getroffen werden kann.

Weitere Forschungsaktivitäten sollten für das Trainieren von domänenspezifischen Word-Embeddings initiiert werden. Die These, dass in Bezug auf domänenzpezifische Tasks selbsttrainierte Models schnell bessere Ergebnisse liefern könnten als die vortrainierten Gegenstücke, sollte untersucht werden. Forschungsbedarf besteht dabei u. a. bei der Berücksichtigung von Fine-Tuning, das z. B. von dem in dieser Arbeit vorgestellten ELMo unterstützt wird. Mit Hilfe des Fine-Tuning könnte es mit verhältnismäßig wenigen neuen Trainingsdaten möglich sein, domänenspezifische Informationen zu erfassen und in Vektoren abbilden zu können. Ein besonderes Augenmerk sollte außerdem auf transformer-basierte Ansätze wie BERT [14] oder GPT-2 [32] gelegt werden, die während der Bearbeitung des Beitrages entwickelt wurden. Die Aktualität des Themas und der in den letzten Jahren feststellbare Architekturwandel von Skip-Gram und CBOW über Recurrent Neural Network und LSTMNN bis zu Transformern zeigt auf, dass im Themengebiet des (textuellen) NLP weiterhin Forschungsbedarf besteht. Zudem existieren bisher keine domänenspezifischen Benchmark-Sets, um die domänenspezifischen und domänenunspezifischen Word-Embeddings auf diesen zu bewerten.

References

1. Enríquez, J.G., Domínguez-Mayo, F.J., Escalona, M.J., Ross, M., Staples, G.: Entity reconciliation in big data sources. A systematic mapping study. Expert Systems with Applications 80, 14–27 (2017)
2. Kooli, N., Allesiardo, R., Pigneul, E.: Deep Learning Based Approach for Entity Resolution in Databases. In: Nguyen, N.T., Hoang, D.H., Hong, T.-P., Pham, H., Trawiński, B. (eds.) Intelligent Information and Database Systems, 10752, pp. 3–12. Springer International Publishing, Cham (2018)

https://doi.org/10.30844/wi_2020_c3-behnen

3. Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., Krishnan, G., Deep, R., Arcaute, E., Raghavendra, V.: Deep Learning for Entity Matching. In: Das, G., Jermaine, C., Bernstein, P. (eds.) Proceedings of the 2018 International Conference on Management of Data - SIGMOD '18, pp. 19–34. ACM Press, New York, New York, USA (2018)
4. Blazquez, D., Domenech, J.: Big Data sources and methods for social and economic analyses. Technological Forecasting and Social Change 130, 99–113 (2018)
5. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching Word Vectors with Subword Information. In: Transactions of the Association for Computational Linguistics, 5, pp. 135–146
6. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations (2018)
7. Ghannay, S., Favre, B., Esteve, Y., Camelin, N.: Word embedding evaluation and combination. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), pp. 300–305 (2016)
8. Lebret, R., Collobert, R.: Word Emdeddings through Hellinger PCA (2013)
9. Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Pasca, M., Soroa, A.: A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. In: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 19–27. Association for Computational Linguistics (2009)
10. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed Representations of Words and Phrases and their Compositionality. In: Advances in neural information processing systems, pp. 3111–3119 (2013)
11. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space (2013)
12. Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., Mikolov, T.: FastText.zip: Compressing text classification models (2016)
13. Pennington, J., Socher, R., Manning, C.: Glove: Global Vectors for Word Representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532–1543 (2014)
14. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (2018)
15. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of Tricks for Efficient Text Classification (2016)
16. Dhillon, P.S., Foster, D.P., Ungar Lyle H.: Eigenwords: Spectral word embeddings. In: The Journal of Machine Learning Research, vol. 16(1)vol. , pp. 3035–3078 (2015)
17. Ji, S., Yun, H., Yanardag, P., Matsushima, S., Vishwanathan, S.V.N.: WordRank: Learning Word Embeddings via Robust Ranking (2015)
18. Shazeer, N., Doherty, R., Evans, C., Waterson, C.: Swivel: Improving Embeddings by Noticing What's Missing (2016)

https://doi.org/10.30844/wi_2020_c3-behnen

19. Bartunov, S., Kondrashkin, D., Osokin, A., Vetrov, D.P.: Breaking Sticks and Ambiguities with Adaptive Skip-gram. In: Artificial Intelligence and Statistics, pp. 130–138 (2016)
20. Speer, R., Chin, J., Havasi, C.: ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. In: Thirty-First AAAI Conference on Artificial Intelligence (2017)
21. Schnabel, T., Labutov, I., Mimno, D., Joachims, T.: Evaluation methods for unsupervised word embeddings. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 298–307 (2015)
22. Baroni, M., Dinu, G., Kruszewski, G.: Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 238–247 (2014)
23. Levy, O., Goldberg, Y., Dagan, I.: Improving Distributional Similarity with Lessons Learned from Word Embeddings. In: Transactions of the Association for Computational Linguistics, 3, pp. 211–225 (2015)
24. Rehurek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pp. 45–50. ELRA, Valletta, Malta (2010)
25. Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N.F., Peters, M., Schmitz, M., Zettlemoyer, L.S.: AllenNLP: A Deep Semantic Natural Language Processing Platform (2018)
26. Bruni, E., Boleda, G., Baroni, M., Khanh Tran, N.: Distributional Semantics in Technicolor. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1, pp. 136–145 (2012)
27. Mikolov, T., Yih, W.-t., Zweig, G.: Linguistic Regularities in Continuous Space Word Representations. In: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 746–751 (2013)
28. Gladkova, A., Drozd, A., Matsuoka, S.: Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In: Proceedings of the NAACL Student Research Workshop, pp. 8–15 (2016)
29. Rogers, A., Drozd, A., Li, B.: The (too Many) Problems of Analogical Reasoning with Word Vectors. In: Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (* SEM 2017), pp. 135–148 (2017)
30. Lewis, D.: Reuters-21578 text categorization test collection. Distribution 1.0, AT&T Labs-Research (1997)
31. Cady, F.: The data science handbook. Wiley, Hoboken NJ (2017)
32. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language Models are Unsupervised Multitask Learners (2019)

https://doi.org/10.30844/wi_2020_c3-behnen