



Bachelorarbeit

im Studiengang Computerlinguistik

an der Ludwig-Maximilians-Universität München

Fakultät für Sprach- und Literaturwissenschaften

Untersuchung von Embedding-Arithmetik in Transformer-Modellen

vorgelegt von
Antonina Gruzdeva

Betreuer: M.Sc. Sebastian Gerstner
Prüfer: Prof. Dr. Hinrich Schütze
Bearbeitungszeitraum: 02.04.2025 – 11.06.2025

Abstract

Diese Arbeit untersucht die geometrischen Eigenschaften der Embedding- und Unembedding-Matrizen klassischer und moderner Sprachmodelle anhand linguistischer Analogieaufgaben. Analysiert werden *Word2Vec* sowie die Transformer-Modelle *BERT*, *GPT-2* und *Pythia 410M*. Zur Evaluation kommen Analogie-Metriken wie *3CosAdd*, *3CosAvg*, *PairDistance* und *OnlyB* zum Einsatz. Die Ergebnisse zeigen, dass alle Modelle syntaktisch reguläre Relationen wie Plural- oder Tempusbildung zuverlässig abbilden, während semantisch komplexere Relationen nur uneinheitlich erfasst werden. *Word2Vec* erzielt in vielen Fällen vergleichbare oder sogar bessere Ergebnisse als die Transformer-Modelle – trotz geringerer Vektordimension. Besonders leistungsfähig ist die Methode *3CosAvg*, die sowohl in morphologischen als auch in semantischen Kategorien robuste Resultate liefert. Unembedding-Matrizen schneiden insgesamt besser ab als die entsprechenden Embeddings, insbesondere bei derivationsmorphologischen Relationen. Methodische Eingriffe wie Vokabularnormalisierung und systematisches Lowercasing könnten die Ergebnisse beeinflusst haben – etwa durch den Verlust distinktiver Wortformen und die Reduktion der Vergleichsdaten in bestimmten Kategorien. Die Befunde verdeutlichen, dass sowohl Modellarchitektur und Repräsentationsebene als auch das Preprocessing durch die Python-Library *TransformerLens* – etwa das Entfernen der Mittelwertkomponente aus dem Word Embedding – die Kodierung linguistischer Relationen im Vektorraum messbar beeinflussen.

Inhaltsverzeichnis

Abstract	i
1. Einleitung und Motivation	1
2. Theoretischer Hintergrund	3
2.1. Word Embeddings und semantische Vektorräume	3
2.2. Embedding-Arithmetik und Analogie-Tests	5
2.2.1. Lineare Verfahren zur Analogieerkennung	5
2.2.2. Einschränkungen linearer Analogieverfahren	8
2.3. Transformer-Modelle: Grundlagen der Architektur und Embeddings	10
3. Methodik und Experimentenaufbau	13
3.1. Auswahl der Sprachmodelle	13
3.2. Datenbasis	15
3.3. Experimentelle Durchführung	16
3.4. Evaluationsergebnisse und -methoden	17
4. Analyse der Ergebnisse und Fazit	19
4.1. Methoden im Vergleich	19
4.2. Modelle im Vergleich	21
4.3. Diskussion der Ergebnisse und methodische Einschränkungen	24
4.4. Fazit	25
Literaturverzeichnis	27
Abbildungsverzeichnis	31
Tabellenverzeichnis	33
A. Ergänzendes Material	35
A.1. Kategorien und Subkategorien der Datenbasis	35
A.2. Kategoriale Unterschiede in der Modell-Accuracy bei vier linearen Analogieverfahren	37
A.3. Top-1 und Top-5 Accuracy pro Methode, Subkategorie und Modellvariante	38
A.4. Top-1 und Top-5 Accuracy pro Kategorie und Modellvariante	41
B. Eingereichte Software and Daten	43

1. Einleitung und Motivation

In den letzten Jahren haben sich *Word Embeddings* als eine der effektivsten Methoden zur Repräsentation von Bedeutung in der maschinellen Sprachverarbeitung etabliert. Insbesondere die Arbeiten von [Mikolov et al. \(2013a,c\)](#) zeigten, dass in Modellen wie *Word2Vec* erlernte Wortvektoren in der Lage sind, linguistische Regularitäten syntaktischer und semantischer Natur zu erfassen.

Eine der bekanntesten Demonstrationen dieser Eigenschaft ist das von [Mikolov et al. \(2013c\)](#) eingeführte *Vektor-Offset*-Verfahren (auch *3CosAdd* genannt). Ein häufig zitiertes Beispiel ist die Gleichung:

$$\vec{\text{king}} \approx \vec{\text{queen}} - \vec{\text{woman}} + \vec{\text{man}}$$

Dieses Verfahren legt nahe, dass die semantische Struktur der Sprache im geometrischen Aufbau des Vektorraums abgebildet wird und dass sich bestimmte linguistische Regularitäten durch einfache arithmetische Vektoroperationen aufdecken lassen.

Der Ansatz von [Mikolov et al.](#) hatte einen weitreichenden Einfluss auf die Forschung zu Analogien mit Word Embeddings und etablierte das Lösen von Analogie-Aufgaben als einen der beliebtesten Benchmarks für die Evaluation von Wort-Vektoren. Im Anschluss entstanden verschiedene Erweiterungen und Alternativen zu *3CosAdd*, etwa *PairDirection* (oder auch *PairDistance* genannt) und *3CosMul* ([Levy and Goldberg, 2014](#)), *3CosAvg* ([Drozd et al., 2016](#)) sowie *3CosWeight* ([Kafe, 2019](#)), die eine differenziertere Analyse semantischer und morphologischer Relationen ermöglichen.

Während sich ein Großteil der bisherigen Forschung auf klassische *Vector Space Models* (VSMs) wie *Word2Vec*, *GloVe* und *fastText* konzentriert hat, ist bislang weitgehend offen geblieben, inwiefern neuere Transformer-Modelle vergleichbare Regularitäten in ihren Embedding- und Unembedding-Matrizen aufweisen. Gerade vor dem Hintergrund der breiten Anwendung von Transformern in der maschinellen Sprachverarbeitung stellt sich die Frage, ob deren Word Embeddings dieselben strukturellen Eigenschaften aufweisen wie frühere Modelle. Die Untersuchung dieser Eigenschaften ist nicht nur theoretisch bedeutsam, sondern liefert auch wichtige Erkenntnisse für die Interpretierbarkeit und Vergleichbarkeit moderner Sprachmodelle.

Zielsetzung und Forschungsfragen

Das Ziel dieser Arbeit besteht darin, zu untersuchen, ob und in welchem Umfang sich semantische und syntaktische Regularitäten, wie sie in klassischen Word Embeddings-Modellen identifiziert wurden, auch in den Embedding- und Unembedding-Matrizen moderner Transformer-Modelle widerspiegeln.

Es soll untersucht werden, ob die in der Literatur etablierten Analogie-Verfahren wie *3CosAdd*, *3CosAvg* und *PairDistance* in der Lage sind, ähnliche Regularitäten in den Matrizen von Transformer-Modellen aufzudecken wie im Fall klassischer VSMs.

Die zentralen Forschungsfragen lauten:

1. Weisen die Embedding- und Unembedding-Matrizen von Transformer-Modellen (*GPT-2*, *BERT*, *Pythia 410m*) ähnliche semantische und syntaktische Vektorstrukturen auf wie klassische Modelle (*Word2Vec*)?
2. Können Analogie-Metriken wie *3CosAdd*, *PairDistance* und *3CosAvg* auf diese Matrizen sinnvoll angewendet werden, um linguistische Relationen zu identifizieren?

1. Einleitung und Motivation

3. Inwiefern unterscheiden sich Transformer-Modelle in ihrer Fähigkeit, bestimmte Relationen zu kodieren, und wie variieren die Ergebnisse je nach Modellarchitektur und Matrizen-Typ (Embedding vs. Unembedding)?

Mit der Beantwortung dieser Fragen soll ein Beitrag zur Evaluation und Vergleichbarkeit von Word Embeddings in klassischen und modernen Sprachmodellen geleistet werden.

2. Theoretischer Hintergrund

2.1. Word Embeddings und semantische Vektorräume

Die Frage nach der Bedeutung von Wörtern und deren maschineller Repräsentation gehört zu den zentralen Herausforderungen der computerlinguistischen Forschung. Eine inzwischen etablierte Lösung besteht in der Nutzung sogenannter Word Embeddings, also numerischer Repräsentationen von Wörtern als Vektoren in einem mehrdimensionalen Raum. Diese Technik basiert auf der distributionellen Hypothese, die bereits in den 1950er Jahren von Linguisten wie Firth (1957), Harris (1954) und Joos (1950) formuliert wurde. Sie besagt, dass Wörter, die in ähnlichen Kontexten auftreten, tendenziell ähnliche Bedeutungen haben. Die Lage eines Wortes in diesem Raum wird aus der statistischen Verteilung seiner Kontexte im Textkorpus abgeleitet.

Solche Vektoren zur Wortrepräsentation werden allgemein als Embeddings bezeichnet. In der Literatur wird der Begriff jedoch nicht immer einheitlich verwendet: Während er im weiteren Sinne auch auf spärliche, zählbasierte Repräsentationen wie *TF-IDF*- oder *PPMI*-Vektoren angewendet wird, ist in jüngeren Arbeiten häufig eine engere Bedeutung gemeint – nämlich dichte, lernbasierte Repräsentationen wie die von *Word2Vec* (Jurafsky and Martin, 2025, S.107). Im Folgenden werden die zentralen Unterschiede zwischen den genannten Vektorrepräsentationen sowie der Übergang von zählbasierten, spärlichen Vektoren zu dichten, lernbasierten Embeddings näher erläutert.

Frühe Repräsentationen in der distributionellen Semantik basierten auf den Vektorraum-Modellen, bei denen Wörter als hochdimensionale, spärlich besetzte Vektoren (*sparse vectors*) dargestellt wurden. In diesen Modellen entspricht jede Dimension typischerweise einem spezifischen Kontext – etwa einem Wort im Vokabular oder einem Dokument in einem Korpus. Die Einträge in diesen Vektoren ergeben sich aus Kookkurrenzstatistiken, also der Häufigkeit, mit der ein Zielwort gemeinsam mit einem Kontextwort auftritt. Solche Repräsentationen entstehen zum Beispiel in Term-Dokument-Matrizen oder Wort-Wort-Kookkurrenzmatrizen, in denen die meisten Werte Null sind, da jedes Wort nur mit einem Bruchteil aller möglichen Kontexten tatsächlich gemeinsam vorkommt (Jurafsky and Martin, 2025, S.107-109).

Um die informativsten Kookkurrenzen hervorzuheben und gleichzeitig triviale Häufigkeiten zu normalisieren, kommen häufig gewichtende Maße zum Einsatz, etwa die *Term Frequency–Inverse Document Frequency* (*TF-IDF*) oder die *Positive Pointwise Mutual Information* (*PPMI*). Diese Maße verbessern die Qualität der semantischen Repräsentationen, indem sie seltene, aber bedeutungstragende Wort-Kontext-Paare stärker gewichten als häufige, aber wenig differenzierende Kombinationen (Jurafsky and Martin, 2025, S.112-116).

Obwohl solche sparse Vektoren eine wichtige Grundlage für die Semantikforschung bildeten, weisen sie mehrere Nachteile auf: Sie sind sehr groß, rechnen sich ineffizient und erfassen semantische Ähnlichkeit nur begrenzt, da sie einzelne Kontexte isoliert behandeln. Zudem führen sie zu sehr hohen Dimensionalitäten, was in vielen maschinellen Lernverfahren problematisch ist.

Ein früher Ansatz zur Überwindung dieser Probleme war das Modell der *Latent Semantic Analysis* (*LSA*), das aus der *Latent Semantic Indexing* (*LSI*)-Methode hervorging. LSA nutzt *Singulärwertzerlegung* (*SVD*), um aus einer gewichteten Term-Dokument-Matrix eine niedrigdimensionale Repräsentation semantischer Strukturen zu erzeugen. Dabei werden die wichtigsten latenten Dimensionen extrahiert, entlang derer die Daten am stärksten variieren. Die so entstandenen dichten Vektoren (z.B. auf 300 Dimensionen reduziert) bildeten eine frühe Form von Word Embeddings und kamen in verschiedensten NLP-Anwendungen zum Einsatz – etwa zur Wortbedeutungsdisambiguierung (Schütze, 1992), morphologischen Analyse, Sprachmodellierung oder automatisierten

Bewertung von Texten (Jurafsky and Martin, 2025, S.130).

Die in LSA erzeugten Vektoren markieren den Übergang von zählbasierten, spärlich besetzten Repräsentationen zu dichten, kontinuierlichen Vektoren, wie sie später in lernbasierten Modellen eingesetzt wurden. Diese Vektoren sind typischerweise kurz (50–1000 Dimensionen), bestehen aus reellen Zahlen und weisen keine direkt interpretierbare Struktur pro Dimension auf. Im Vergleich zu spärlichen, oft zehntausende Dimensionen umfassenden Vektoren verbessern diese kompakteren Repräsentationen die Performanz in nahezu allen NLP-Aufgaben, da sie weniger Parameter erfordern und somit besser generalisieren können (Jurafsky and Martin, 2025, S.117–118).

Ein prominentes Verfahren zur Berechnung solcher dichten Wortvektoren ist das Word2Vec-Framework, das von Mikolov et al. (2013a) vorgestellt wurde und zwei Architekturen umfasst: *CBOW* (*Continuous Bag-of-Words*) und *Skip-Gram*. Wie in Abbildung 2.1 gezeigt, unterscheidet sich das Skip-Gram-Modell vom CBOW-Ansatz in der Richtung der Prädiktion: Während CBOW aus einem gegebenen Kontext das Zielwort zu rekonstruieren versucht, generiert Skip-Gram aus einem gegebenen Zielwort die umgebenden Kontextwörter.

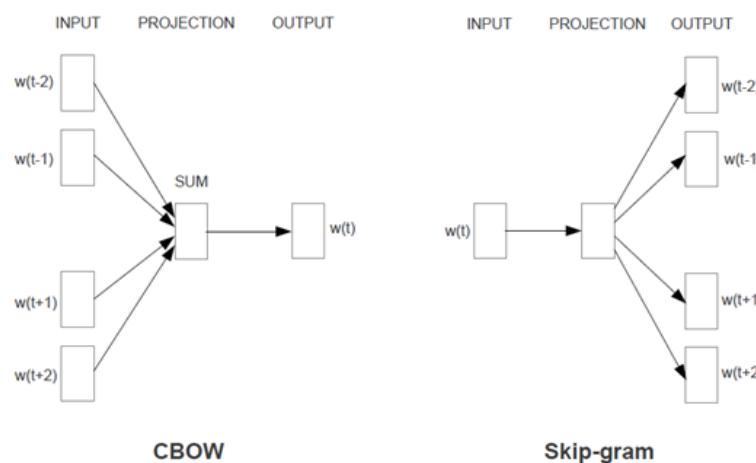


Abbildung 2.1.: CBOW- und Skip-Gram-Architektur (Mikolov et al., 2013a).

In der Praxis hat sich das *Skip-Gram-Modell mit Negative Sampling (SGNS)* als besonders leistungsfähig bei der Modellierung semantischer Relationen erwiesen Mikolov et al. (2013a). Die zugrunde liegende Idee besteht darin, einen binären Klassifikator zu trainieren, der entscheidet, ob ein Wort im Kontext eines Zielwertes auftritt oder nicht. Die beim Training dieses Klassifikators gelernten Gewichtungen werden anschließend als semantisch interpretierbare Wortvektoren verwendet.

Besonders bemerkenswert ist, dass dieses Verfahren keine manuelle Annotation benötigt, sondern lediglich auf selbstüberwachtem Lernen basiert: Kontextwörter, die in natürlicher Sprache vorkommen, dienen implizit als Trainingssignal. Damit schließt Word2Vec an frühere Arbeiten zum neuronalen Sprachmodellieren an (Bengio et al., 2003; Collobert et al., 2011), vereinfacht jedoch sowohl die Zielaufgabe (binäre Klassifikation statt Wortvorhersage) als auch die Modellarchitektur (logistische Regression statt tiefer neuronaler Netzwerke).

Neben Word2Vec wurden weitere Modelle für Word Embeddings entwickelt, die spezifische Schwächen klassischer Verfahren adressieren. Ein prominentes Beispiel ist *fastText* (Bojanowski et al., 2017), das Word2Vec um Subword-Informationen erweitert. Dies ermöglicht die Modellierung seltener oder unbekannter Wörter, indem jedes Wort als Kombination aus sich selbst und einer Menge von Zeichen-n-Grammen dargestellt wird. Für jedes n-Gramm wird ein Embedding gelernt, und die Wortrepräsentation ergibt sich aus der Summe dieser Teil-Embeddings. Dadurch eignet sich fastText besonders gut für morphologisch reiche Sprachen (Jurafsky and Martin, 2025, S.123).

Ein weiteres verbreitetes Modell ist *GloVe* (Pennington et al., 2014), das auf der globalen Kookkurrenzstatistik eines Korpus basiert. Es verbindet Elemente zählbasierter Modelle (wie PPMI) mit den linearen Struktureigenschaften prädiktiver Verfahren wie Word2Vec. Dabei wird das Verhältnis von Kookkurrenzwahrscheinlichkeiten genutzt, um semantische Relationen abzubilden.

Die Fähigkeit von Word Embeddings, semantische Analogien durch einfache Vektoroperationen abzubilden, wurde in der Folgezeit zu einem zentralen Benchmark bei der Evaluation solcher Modelle. Im folgenden Kapitel wird daher näher untersucht, wie solche Analogie-Aufgaben formalisiert und ausgewertet werden können. Dabei stehen lineare Verfahren der Analogieauflösung wie *3CosAdd*, *PairDistance* und *3CosAvg* im Fokus. Aufbauend auf den in diesem Kapitel erläuterten Grundlagen bildet diese Analyse die methodische Basis für die spätere Untersuchung statischer Embedding-Strukturen in Transformer-Modellen.

2.2. Embedding-Arithmetik und Analogie-Tests

Ein wichtiger Aspekt semantischer Word Embeddings ist ihre Fähigkeit, relationale Bedeutungen durch arithmetische Operationen im Vektorraum zu modellieren. Diese Eigenschaft wurde ursprünglich durch das sogenannte Parallelogramm-Modell beschrieben, das von Rumelhart and Abrahamson (1973) als kognitives Modell zur Lösung einfacher Analogien vorgeschlagen wurde. Dabei wurde angenommen, dass Analogie-Relationen wie a ist zu a^* wie b zu b^* geometrisch durch Vektoroperationen dargestellt werden können.

Für eine Analogieaufgabe der Form $a : a^* :: b : b^*$ erhält das Modell drei Vektoren – \vec{a} , \vec{a}^* und \vec{b} – und soll den fehlenden Vektor \vec{b}^* bestimmen. Die zugrunde liegende Methode besteht darin, die Differenz zwischen \vec{a}^* und \vec{a} zu berechnen und diese auf \vec{b} zu übertragen. Gesucht wird dann das Wort, dessen Vektor möglichst nahe bei der berechneten Position liegt:

$$\vec{b}^* = \arg \min_x \text{distance}\left(x, \vec{a}^* - \vec{a} + \vec{b}\right) \quad (2.1)$$

Als Distanzmaß kann beispielsweise die *euklidische Distanz* verwendet werden (Jurafsky and Martin, 2025, S.124-125).

In moderneren Word-Embedding-Modellen wie Word2Vec (Mikolov et al., 2013a) oder GloVe (Pennington et al., 2014) wurde dieser Ansatz weiterentwickelt und systematisch evaluiert. Dort konnte gezeigt werden, dass Relationen wie *MALE-FEMALE* oder *CAPITAL-CITY-OF* überraschend präzise modelliert werden können (Jurafsky and Martin, 2025, S.125). Dadurch wurde die Analogie-Aufgabe zu einem der wichtigsten Benchmarks für die Evaluation semantischer Vektorräume.

Allerdings ist dieser Ansatz nicht ohne Einschränkungen. So neigen Modelle dazu, bei Analogie-Aufgaben zunächst eines der Eingabewörter oder deren morphologische Varianten als nächstliegende Lösung zurückzugeben – ein Effekt, der durch gezieltes Ausschließen von Eingabewörtern kompensiert werden muss. Zudem zeigen empirische Untersuchungen (Linzen, 2016; Gladkova et al., 2016; Schlüter, 2018; Peterson et al., 2020), dass die Vektorarithmetik zwar gut für häufige Wörter und einfache Relationen funktioniert, jedoch bei komplexeren semantischen Strukturen, abstrakten Konzepten oder kognitiv anspruchsvoller Analogien deutlich schlechter abschneidet.

Trotz dieser Einschränkungen ist die parallelogramm-basierte Analogieauflösung bis heute ein zentraler Untersuchungsgegenstand in der semantischen Evaluation von Embedding-Modellen. Im folgenden Abschnitt werden verschiedene Metriken vorgestellt, mit denen sich die Fähigkeit von Embedding-Räumen zur Modellierung solcher Analogie-Relationen präziser erfassen lässt.

2.2.1. Lineare Verfahren zur Analogieerkennung

Die im Weiteren beschriebene Klasse von Methoden zur Analogieauflösung in semantischen Vektorräumen stützt sich auf sogenannte lineare Verfahren, bei denen angenommen wird, dass sich semantische Relationen durch einfache algebraische Operationen wie Addition, Subtraktion und

Mittelwertbildung modellieren lassen: das *3CosAdd*-Verfahren von Mikolov et al. (2013c), die *PairDistance*-Methode von Levy and Goldberg (2014), sowie die von Drozd et al. (2016) eingeführte Methode *3CosAvg*.

3CosAdd

Ein erster einflussreicher Ansatz zur Lösung der Analogieaufgaben wurde von Mikolov et al. (2013c) vorgeschlagen und bildet die Grundlage für viele spätere Evaluationsmetriken.

Ein zentrales Ergebnis der Arbeit von Mikolov et al. (2013c) ist die Beobachtung, dass kontinuierliche Word Embeddings in der Lage sind, linguistische Regularitäten durch einfache Vektoroperationen zu modellieren. Diese Regularitäten – sowohl syntaktischer als auch semantischer Natur – lassen sich als konstante Vektor-Offsets zwischen Wortpaaren darstellen, die eine bestimmte Beziehung teilen.

Um Analogiefragen der Form $a : a^* :: b : b^*$ zu beantworten, schlagen die Autoren ein einfaches Rechenverfahren vor: Ausgehend von drei bekannten Vektoren \vec{a} , \vec{a}^* und \vec{b} wird ein Zielvektor \vec{b}^* approximiert, indem der Vektorunterschied $\vec{a}^* - \vec{a}$ auf \vec{b} übertragen wird. Gesucht wird dann das Wort, dessen Vektor am nächsten zur resultierenden Position liegt – gemessen an der Kosinusähnlichkeit (Gleichung 2.2).

$$\arg \max_{\vec{b}^* \in V} \left(\cos \left(\vec{b}^*, \vec{a}^* - \vec{a} + \vec{b} \right) \right) \quad (2.2)$$

In ihrer Arbeit untersuchen Mikolov et al. (2013c) kontinuierliche Wortvektoren, die mit einem rekurrenten neuronalen Sprachmodell (RNNLM) trainiert wurden. Die Evaluation erfolgt durch zwei speziell konstruierte Testsets: ein syntaktisches, das u.a. Flexionsformen von Adjektiven, Nomen und Verben prüft, und ein semantisches, basierend auf der SemEval-2012-Aufgabe zur relationalen Ähnlichkeit. Die Ergebnisse zeigen, dass das RNNLM – trotz rein unüberwachter Trainingsmethode – etwa 40% der syntaktischen Fragen korrekt beantwortet und im semantischen Teil die damals besten Systeme übertrifft.

Levy and Goldberg (2014) haben die von Mikolov et al. eingeführte Methode zur Lösung von Wortanalogien als *3CosAdd* bezeichnet. Der Name leitet sich von einer Umformulierung der ursprünglichen Formel ab, die sie vorgenommen haben: Bei der Berechnung der Zielantwort werden drei Kosinuswerte additiv miteinander kombiniert. Diese Variante (vgl. Gleichung 2.3) ist mathematisch äquivalent zur ursprünglichen Vektoroperation (vgl. Gleichung 2.2), sofern alle Wortvektoren zuvor auf Einheitslänge normalisiert wurden.

$$\arg \max_{\vec{b}^* \in V} \left(\cos(\vec{b}^*, \vec{a}^*) - \cos(\vec{b}^*, \vec{a}) + \cos(\vec{b}^*, \vec{b}) \right) \quad (2.3)$$

Levy and Goldberg argumentieren, dass diese Variante gegenüber der ursprünglichen Formulierung einen klareren Einblick in die Funktionsweise des Analogieschlusses gibt, auch wenn beide rechnerisch dasselbe Ergebnis liefern.

Aufbauend auf Mikolovs Methode wurden zahlreiche Varianten und Erweiterungen entwickelt. Eine davon ist *PairDistance*, ein Ansatz von Levy and Goldberg (2014).

PairDistance

Dieser alternative Ansatz geht ebenfalls von der Annahme aus, dass semantische Relationen als Vektoroperationen im Word-Embedding-Raum abgebildet werden können – baut jedoch auf einer anderen geometrischen Intuition auf als der klassische Vektor-Offset-Ansatz (*3CosAdd*) nach Mikolov et al. (2013c).

Diese Methode ignoriert die absolute Lage der Vektoren im Raum und konzentriert sich stattdessen ausschließlich auf die semantische Verschiebung, die durch die Analogie definiert ist. Ziel ist es, ein Wort \vec{b}^* zu finden, das sich zu \vec{b} in etwa so verhält wie \vec{a}^* zu \vec{a} . Die Methode legt also den

Fokus auf die Richtung zwischen den Vektoren und ist besonders nützlich, wenn es darum geht, Relationen anhand ihrer Richtung im Vektorraum zu erkennen – unabhängig von der konkreten Position der beteiligten Wörter.

Die Berechnung erfolgt nach folgender Formel:

$$\arg \max_{\vec{b}^* \in V} \left(\cos(\vec{b}^* - \vec{b}, \vec{a}^* - \vec{a}) \right) \quad (2.4)$$

Dabei umfasst der Suchraum V alle Wörter im Vokabular, ausgenommen die bereits in der Analogie vorkommenden Wörter – also \vec{a} , \vec{a}^* und \vec{b} , um zu vermeiden, dass eines dieser Wörter aufgrund seiner Nähe zum Suchvektor fälschlicherweise als Lösung gewählt wird.

In ihren Experimenten zeigen [Levy and Goldberg \(2014\)](#), dass PairDistance insbesondere bei kontrollierten Aufgaben mit geschlossenem Vokabular – wie im SemEval 2012 Task – gut funktioniert und dort sogar bessere Ergebnisse liefert als 3CosAdd. Bei offenen Vokabularaufgaben (wie den Google- oder MSR-Datensätzen), bei denen aus dem gesamten Vokabular ein Zielwort gefunden werden muss, schneidet die Methode hingegen deutlich schlechter ab.

Neben 3CosAdd und PairDistance schlagen [Levy and Goldberg \(2014\)](#) zudem eine weitere Variante vor: *3CosMul*. Sie zeigt in vielen Fällen bessere Resultate als 3CosAdd – allerdings wird diese Methode hier nicht im Detail betrachtet, da der Fokus dieser Arbeit ausschließlich auf den linearen Verfahren zur Analogieauflösung liegt.

Zusammenfassend bietet PairDistance eine neue Möglichkeit, semantische Relationen über Richtungsinformationen im Vektorraum zu erfassen – eignet sich jedoch vorrangig für Aufgaben mit eingeschränkter Wortauswahl. Im experimentellen Teil dieser Arbeit wird untersucht, wie gut sich diese und verwandte Methoden zur Analyse von Relationstypen in Transformer-Modellen anwenden lassen.

3CosAvg

Eine weitere alternative Methode zur Auflösung von Analogien stellt *3CosAvg* dar, vorgeschlagen von [Drozd et al. \(2016\)](#).

In ihrer Arbeit setzen sich die Autoren kritisch mit der von [Mikolov et al.](#) eingeführten 3CosAdd-Methode auseinander und zeigen, dass sie in vielen Fällen nicht zuverlässig funktioniert, insbesondere bei komplexeren oder morphologischen Relationen. Grund dafür ist die Sensitivität gegenüber idiosynkratischen Eigenschaften einzelner Wörter, wie etwa Polysemie oder thematisch abweichender Gebrauch (z.B. *queen* als Königin vs. Musikband). Diese Probleme führen dazu, dass die Methode systematisch fehlschlägt oder durch triviale Nachbarschaftseffekte in den Vektorräumen verzerrt wird ([Drozd et al., 2016](#)).

Die Autoren schlagen zwei alternative Verfahren vor, die nicht auf Ein-Paar-Vergleiche, sondern auf die Auswertung mehrerer analoger Wortpaare beruhen: *3CosAvg* und *LRCos*. Die Methode *LRCos* wird hier im Weiteren nicht näher betrachtet, weil es ein lernbasiertes Verfahren zur Analogieauflösung darstellt.

3CosAvg ist eine Erweiterung von 3CosAdd, bei der der durchschnittliche Offset aus mehreren Beispieldpaaren verwendet wird, um die Zielrelation zu modellieren. Das resultierende Zielwort \vec{b}^* wird dann anhand der *Kosinusähnlichkeit* zu $\vec{b} + \vec{o}_{avg}$ bestimmt:

$$\arg \max_{\vec{b}^* \in V} \left(\cos \left(\vec{b}^*, \vec{b} + \vec{o}_{avg} \right) \right) \quad (2.5)$$

Der durchschnittliche Offset \vec{o}_{avg} wird wie folgt berechnet:

$$\vec{o}_{avg} = \frac{\sum_{i=0}^m \vec{a}_i^*}{m} - \frac{\sum_{i=0}^n \vec{a}_i}{n} \quad (2.6)$$

Dabei stehen \vec{a}_i für die Vektoren der Quellklasse und \vec{a}_i^* für die Vektoren der Zielklasse.

Die Methoden werden auf dem Google Analogy Test Set (Mikolov et al., 2013a) und dem Bigger Analogy Test Set (BATS) (Gladkova et al., 2016) evaluiert. Trotz seiner Einfachheit übertrifft 3CosAvg in vielen Fällen die klassische 3CosAdd-Methode.

2.2.2. Einschränkungen linearer Analogieverfahren

Lineare Verfahren zur Analogieauflösung wie 3CosAdd, PairDistance oder 3CosAvg beruhen, wie schon erwähnt, auf der Grundannahme, dass semantische Relationen durch Vektoroperationen im Repräsentationsraum abgebildet werden können. Diese Methoden haben in vielen frühen Studien beeindruckende Ergebnisse gezeigt, insbesondere bei klassischen Benchmarks wie dem Google Analogy Test Set (Mikolov et al., 2013a). Dennoch weisen sie in der Praxis erhebliche Einschränkungen auf – sowohl im Hinblick auf die Bandbreite erkennbarer Relationen als auch auf ihre Robustheit gegenüber unterschiedlichen Sprachphänomenen. In diesem Kapitel werden empirische und theoretische Arbeiten vorgestellt, die diese Limitationen aufzeigen.

Die Arbeit von Gladkova et al. (2016) hinterfragt kritisch die Leistungsfähigkeit linearer Verfahren wie 3CosAdd bei der Erkennung linguistischer Relationen mit Word Embeddings. Zu diesem Zweck führen die Autoren eine umfassende Evaluation auf einem selbst entwickelten, ausgewogenen Datensatz durch – dem *Bigger Analogy Test Set (BATS)*, das 40 Kategorien morphologischer und semantischer Relationen umfasst.

Die wichtigsten Befunde lassen sich wie folgt zusammenfassen:

- **Unterschiedliche Leistung je nach Relationstyp**

Lineare Verfahren zeigen deutliche Unterschiede in ihrer Leistungsfähigkeit je nach Art der zu modellierenden Relation. Während sie bei regelmäßigen flektierenden Relationen (z.B. Pluralbildung) relativ zuverlässig funktionieren, schneiden sie bei derivationaler Morphologie oder semantisch komplexeren Relationen wie Synonymie, Antonymie oder Meronymie deutlich schlechter ab.

Ein häufig beobachteter Fehler ist, dass die Modelle statt einer inhaltlich richtigen Lösung eine formal ähnliche Wortform vorhersagen – etwa *potato* → *potatoes* statt *potato* → *brown*. Dies deutet darauf hin, dass nicht die semantische Relation erkannt wird, sondern lediglich formale Gemeinsamkeiten dominieren. Besonders bei lexikalisch oder morphologisch mehrdeutigen Relationen stellt dies eine zentrale Schwäche linearer Verfahren dar.

- **Hohe Abhängigkeit von Modellparametern**

Die Studie zeigt, dass die Performanz linearer Methoden von der Art der Relation und den Modellparametern (Fenstergröße, Dimensionalität) stark abhängt. Es gibt keine universell optimalen Einstellungen, was die Anwendung linearer Verfahren in der Praxis erschwert. Entgegen gängiger Annahmen brachte eine Vergrößerung der Fensterweite oder Dimensionalität nicht durchgängig Leistungssteigerungen, und es ließ sich kein systematischer Zusammenhang zwischen Relations- typ und optimalen Parametern feststellen.

- **Geringe Gesamtergebnisse auf BATS**

Selbst das leistungsstärkste Modell (GloVe) erreichte auf BATS lediglich 28,5% durchschnittliche Genauigkeit, während ein SVD-Modell nur 22,1% erzielte.

Die Ergebnisse von Gladkova et al. (2016) zeigen deutlich, dass lineare Analogieverfahren wie 3CosAdd nur eine eingeschränkte Modellierungskapazität besitzen. Insbesondere bei komplexen morphologischen und semantischen Relationen stoßen sie an ihre Grenzen. Die Studie unterstreicht die Notwendigkeit, die Aussagekraft solcher Verfahren kritisch zu hinterfragen und differenzierte Evaluationsdatensätze zu verwenden, um die tatsächliche Modellleistung zu beurteilen.

Während Gladkova et al. (2016) empirisch zeigen, dass lineare Analogieverfahren inhaltlich begrenzt sind, setzt sich Linzen (2016) grundsätzlicher mit der Interpretierbarkeit und Aussagekraft

der Methode selbst auseinander. Er kritisiert, dass der sogenannte Offset-Ansatz in der Praxis nicht das misst, was er vorgibt zu messen.

[Linzen](#) zeigt, dass die Methode 3CosAdd in vielen Fällen nicht deshalb erfolgreich ist, weil das Modell tatsächlich eine Relation als konsistente Richtung abbildet, sondern weil das Zielwort b^* ohnehin der nächste Nachbar im Vektorraum zu b ist. Er führt mehrere einfache Baselines ein – etwa *OnlyB* (gibt den nächsten Nachbarn von b zurück) oder *IgnoreA* (nutzt nur b und a^*) – und zeigt, dass diese in vielen Fällen ähnlich gute oder bessere Ergebnisse erzielen als der eigentliche Offset-Ansatz. In der vorliegenden Arbeit wird die von Linzen eingeführte Methode *OnlyB* als Baseline unter den Methoden benutzt.

Ein weiteres kritisches Experiment betrifft die Richtungsumkehr: Wenn statt $base \rightarrow gerund$ die umgekehrte Analogie $gerund \rightarrow base$ gestellt wird, bricht die Leistung der Methode teilweise massiv ein. Das wäre nicht zu erwarten, wenn es wirklich um konstante semantische Relationen ginge – eine konsistente Relation müsste in beiden Richtungen gleich zuverlässig funktionieren. Der starke Leistungsabfall zeigt, dass auch hier Nachbarschaftseffekte eine dominante Rolle spielen.

Die Analyse von [Linzen](#) (2016) ergänzt die Ergebnisse von [Gladkova et al.](#) (2016) um eine tiefgreifende methodische Kritik: Lineare Analogieverfahren wie 3CosAdd messen in vielen Fällen nicht die Konsistenz semantischer Relationen, sondern profitieren von trivialen Nachbarschaftsstrukturen im Vektorraum. Baselines ohne Offset erzielen oft vergleichbare Ergebnisse, und bereits kleine Änderungen im Aufbau der Analogieaufgabe führen zu drastischen Leistungsunterschieden.

Die Kritik am linearen Analogieparadigma wird durch die Arbeit von [Rogers et al.](#) (2017) weiter gestützt und vertieft. In einer Reihe kontrollierter Experimente demonstrieren die Autoren, dass viele der bereits von [Gladkova et al.](#) (2016) und [Linzen](#) (2016) beobachteten Schwächen – etwa die starke Abhängigkeit von Nachbarschaftsstruktur, die fehlende Konsistenz der Offsets sowie die Sensitivität gegenüber Aufgabenformaten und Parameterwahl – systematisch auftreten.

Besonders hervorzuheben ist ihr Einwand gegen die Grundannahme semantischer Regelmäßigkeit: [Rogers et al.](#) zeigen, dass semantische Relationen im Vektorraum nicht durch stabile, übertragbare Richtungen repräsentiert werden. Vielmehr variieren Offsets selbst innerhalb gleichartiger Relationen stark, was die Vorstellung linearer Analogien als allgemein gültige semantische Gesetzmäßigkeiten infrage stellt. Auch methodisch argumentieren sie gegen die Aussagekraft gängiger Benchmarks, da hohe Scores häufig eher auf strukturelle Nähe als auf tatsächliche Relatiionserkennung zurückzuführen sind.

Insgesamt untermauert die Studie die These, dass lineare Verfahren wie 3CosAdd kein zuverlässiges Instrument zur Modellierung semantischer Relationen darstellen.

Die methodischen Zweifel an der Validität linearer Analogieverfahren werden in der Arbeit von [Schluter](#) (2018) noch weiter vertieft. Die Autorin argumentiert, dass die Annahme konsistenter Analogierichtungen im Vektorraum nicht aus dem distributionellen Lernprinzip abgeleitet werden kann. Lineare Relationen wie *king - man + woman = queen* erscheinen damit methodisch unbegründet, selbst wenn sie empirisch gelegentlich erfolgreich sind.

Zudem werden konventionelle Testpraktiken (wie die Normalisierung von Vektoren oder das gezielte Ausschließen von Prämissen-Wörtern aus dem Zielvokabular) als verzerrnd und aussagearm kritisiert. Auch gängige Visualisierungen, etwa PCA-Projektionen auf zwei Dimensionen, seien nicht geeignet, um echte Analogierichtungen im Raum sichtbar zu machen.

Die Arbeit von [Schluter](#) unterstreicht damit die zuvor geäußerten Einwände (vgl. [Linzen](#) (2016), [Rogers et al.](#) (2017)) und betont, dass Analogieaufgaben, wie sie in vielen Benchmarks verwendet werden, methodisch fragwürdig und nur begrenzt interpretierbar sind.

[Peterson et al.](#) (2020) erweitern die bestehende Kritik an linearen Analogieverfahren um eine kognitionspsychologische Perspektive. Sie untersuchen, ob das parallelogrammbasierte Modell menschliche Analogieurteile tatsächlich abbildet. Ihre Ergebnisse zeigen, dass dies nur für wenige Relationstypen der Fall ist. Besonders bei konzeptuell anspruchsvoller Relationen versagt das

Modell häufig.

Zudem weisen sie nach, dass Menschen oft einfachere Strategien verwenden – etwa die Wahl des semantisch ähnlichen Wortes (*Item-Ähnlichkeit*) – und dass diese Strategie in vielen Fällen besser mit den tatsächlichen Urteilen übereinstimmt als die Vektoroperation. Diese Ergebnisse stehen in engem Zusammenhang mit der Kritik von Linzen (2016), laut der der Erfolg von 3CosAdd häufig auf bloßer Nachbarschaftsähnlichkeit beruht. Beide Arbeiten stellen damit die interpretative Tiefe vektorbasierten Analogieschlusses infrage.

Trotz der in diesem Kapitel dargestellten Einschränkungen linearer Analogieverfahren bleibt das Interesse an diesen Methoden bestehen – nicht zuletzt, weil sie einfache und interpretierbare Einblicke in die Struktur semantischer Repräsentationen ermöglichen. Besonders im Kontext aktueller Sprachmodelle stellt sich die Frage, ob und inwiefern sich semantische Relationen auch in den Embedding- und Unembedding-Matrizen moderner Transformer-Architekturen erkennen lassen. Das folgende Kapitel führt daher in die Grundlagen der Transformer-Architektur ein und beschreibt die spezifische Form der statischen Embeddings, die in der anschließenden Analyse verwendet werden.

2.3. Transformer-Modelle: Grundlagen der Architektur und Embeddings

Transformer-Modelle haben sich in den letzten Jahren als Standardarchitektur in der Verarbeitung natürlicher Sprache etabliert. Ihre Fähigkeit, Sprachverständnis auf Grundlage großer Textkorpora zu modellieren, beruht maßgeblich auf der Art und Weise, wie sie Wörter in hochdimensionalen Vektorräumen repräsentieren. Dabei unterscheidet sich die Struktur dieser Repräsentationen von klassischen Wort-Embeddings wie denen aus Word2Vec – sowohl hinsichtlich ihrer Erzeugung als auch ihrer Einbettung in die Modellarchitektur.

Dieses Kapitel führt in die zentralen Komponenten der Transformer-Architektur ein, wobei der Schwerpunkt auf der Rolle der statischen Word Embeddings liegt. Ziel ist es, ein funktionales Verständnis der Embedding- und Unembedding-Schichten zu vermitteln, die im weiteren Verlauf dieser Arbeit im Rahmen der experimentellen Analyse untersucht werden.

Die Transformer-Architektur wurde von Vaswani et al. (2017) eingeführt und markiert einen fundamentalen Paradigmenwechsel in der Verarbeitung natürlicher Sprache. Anders als frühere Modelle, die auf rekurrente oder konvolutionale Mechanismen setzten, basiert der Transformer vollständig auf *Attention*, genauer gesagt auf dem sogenannten *Self-Attention-Mechanismus*. Dabei handelt es sich um ein Verfahren, das es dem Modell erlaubt, bei der Verarbeitung eines Tokens gezielt auf andere relevante Positionen in der Eingabesequenz zu „achten“ (Vaswani et al., 2017). Da sich diese Arbeit jedoch vorrangig mit anderen architekturellen Aspekten des Transformers beschäftigt, wird auf den Attention-Mechanismus an dieser Stelle nicht näher eingegangen.

Abbildung 2.2 veranschaulicht die grundlegende Struktur eines Transformer-Modells, das sich aus drei Hauptkomponenten zusammensetzt. Im Zentrum befinden sich *Stapel von Transformer-Blöcken* (*Stacked Transformer Blocks*), die jeweils aus mehreren Schichten bestehen – typischerweise aus einer *Multi-Head-Attention-Schicht*, einer *Feedforward-Komponente* sowie *Layer - Normalisierung* und *Residualverbindungen*. Jeder dieser Blöcke verarbeitet einen Eingabe-Vektor \vec{x}_i zu einem kontextualisierten Ausgabe-Vektor \vec{h}_i . Der gesamte Stapel von Blöcken verarbeitet auf diese Weise ein Fenster von Eingabevektoren ($\vec{x}_i \dots \vec{x}_n$) zu einer entsprechenden Sequenz von Ausgabevektoren ($\vec{h}_i \dots \vec{h}_n$), wobei ein Stapel je nach Modellgröße zwischen 12 und 96 oder sogar mehr solcher Blöcke enthalten kann (Jurafsky and Martin, 2025, S.184-185).

Vor den Transformer-Blöcken erfolgt eine *Eingabekodierung* (*Input Encoding*), bei der jedes Token über eine *Embedding-Matrix E* in einen kontinuierlichen Vektorraum abgebildet und zusätzlich um Positionsinformationen ergänzt wird. Nach den Transformer-Blöcken folgt der sogenannte *Language Modeling Head*, der den finalen Vektor \vec{h}_i eines Tokens – also das letzte Hidden State der Transformer-Schicht an Position i – übernimmt, ihn mit einer *Unembedding-Matrix U*

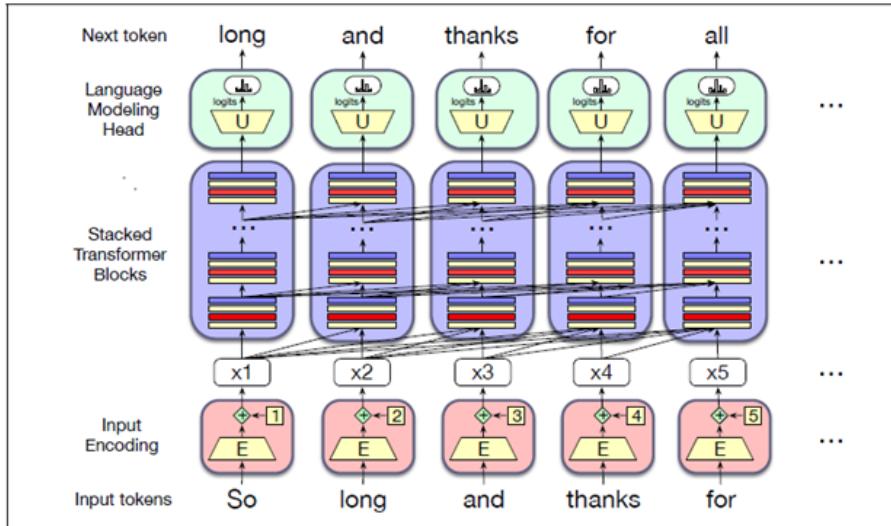


Abbildung 2.2.: Die Architektur eines (left-to-right) Transformers. Sie zeigt, wie jedes Eingabetoken kodiert, durch eine Abfolge gestapelter Transformer-Blöcke weiterverarbeitet und schließlich über einen Language-Modeling-Head zur Vorhersage des nächsten Tokens genutzt wird (Jurafsky and Martin, 2025, S.184).

multipliziert und anschließend mithilfe einer Softmax-Funktion in eine Wahrscheinlichkeitsverteilung über das Vokabular umwandelt. Auf diese Weise wird ein nächstes Token für jede Position vorhergesagt.

In der vorliegenden Arbeit werden genau diese beiden *Matrizen* E und U untersucht: die Embedding-Matrix zu Beginn und die Unembedding-Matrix am Ende des Modells. Beide liefern statische Vektorrepräsentationen von Wörtern und stellen damit eine Schnittstelle dar, an der sich untersuchen lässt, ob und inwieweit moderne Transformer-Modelle semantische Relationen in linearer Form kodieren.

Dabei ist anzumerken, dass diese zwei Matrizen in manchen Modellen tatsächlich identisch sind. In dem Fall spricht man vom sogenannten *Weight Tying*. In Sprachmodellen mit Weight Tying wird dieselbe Embedding-Matrix $E \in \mathbb{R}^{|V| \times d}$ sowohl im Eingabe- als auch im Ausgabelayer verwendet. Dabei bezeichnet $|V|$ die Größe des Vokabulars und d die Dimensionalität des Vektorraums. Anstelle einer separaten Ausgabematrix $U \in \mathbb{R}^{d \times |V|}$, die zur Projektion des letzten Hidden States in den Vokabularraum verwendet wird, nutzt man bei Weight Tying einfach die Transponierte der Embedding-Matrix, also $E^\top \in \mathbb{R}^{d \times |V|}$. Dadurch entsteht ein gemeinsamer Vektorraum für Eingabe und Ausgabe, was Parameter spart und häufig die Modellleistung verbessert. (Press and Wolf, 2017; Jurafsky and Martin, 2025, S.164-165).

Die Transformer-Architektur wird in verschiedenen Varianten eingesetzt: als *Encoder-Decoder-Modell* (auch bekannt als *Sequence-to-Sequence-Model*) etwa in der maschinellen Übersetzung (Vaswani et al., 2017), als rein encoderbasiertes Modell wie BERT (Devlin et al., 2019) oder als rein decoderbasiertes Modell wie GPT (Radford et al., 2019). Auf diese Architekturen wird im nächsten Kapitel, in dem die für das Experiment ausgewählten Modelle vorgestellt werden, näher eingegangen.

3. Methodik und Experimentenaufbau

3.1. Auswahl der Sprachmodelle

Im Rahmen des Experiments kamen folgende Sprachmodelle zum Einsatz: *Word2Vec*, *GPT-2 small* (im Folgenden *GPT-2*), *BERT base uncased* (im Folgenden *BERT*) sowie *Pythia 410M* (im Folgenden *Pythia*).

Word2Vec

Das Word2Vec-Modell, eingeführt von [Mikolov et al. \(2013a\)](#), gehört zu den grundlegenden Methoden zur Erstellung von Word Embeddings.

Word2Vec umfasst zwei Trainingsarchitekturen: Continuous Bag-of-Words (CBOW) und Skip-Gram, auf die im Kapitel 2.1 der vorliegenden Arbeit bereits eingegangen worden ist (vgl. Abbildung 2.1). Beide Modelle sind rein log-linear und verzichten auf versteckte Schichten, was sie besonders effizient und skalierbar macht.

In diesem Experiment wird das vortrainierte Vektorset *GoogleNews-vectors-negative300.bin*¹ verwendet, das von Google Research auf ca. 100 Milliarden Tokens aus Google News mithilfe des Skip-Gram-Modells mit Negative Sampling trainiert wurde. Es umfasst rund drei Millionen Wörter und Phrasen, wobei jeder Vektor eine Dimension von 300 besitzt. Das Modell dient in diesem Experiment als Baseline, anhand derer die Qualität modernerer Modelle wie GPT-2, BERT und Pythia verglichen wird.

GPT-2

Das Modell GPT-2 wurde von [Radford et al. \(2019\)](#) vorgestellt. Es handelt sich um ein auf der Transformer-Architektur basierendes unidirektionales Sprachmodell, das durch unüberwachtes Training auf einer sehr großen Textmenge generelle Sprachrepräsentationen lernt.

GPT-2 nutzt ausschließlich den Decoder-Teil des Transformer-Modells, wobei Mechanismen wie *Self-Attention*, *Maskierung (Causal Masking)* und *Positionsemmbeddings* zum Einsatz kommen, um sequenzielle Textvorhersagen zu ermöglichen. Die Architektur folgt einem strikt autoregressiven Ansatz, bei dem der nächste Token jeweils ausschließlich auf Basis des vorhergehenden Kontexts vorhergesagt wird.

Die ursprüngliche GPT-2-Modellreihe umfasst vier Varianten unterschiedlicher Größe: *small*, *medium*, *large*, *extra large*. Diese unterscheiden sich nicht in ihrer grundlegenden Architektur, sondern in der Anzahl an Layern, der Breite der Hidden-Dimensionen und der Gesamtanzahl an Parametern. In dieser Arbeit wird das kleinste Modell, *GPT-2 small*, verwendet. Es besteht aus 12 Transformer-Blöcken, nutzt Word Embeddings mit einer Dimension von 768 sowie ein Vokabular mit 50 257 Token und umfasst insgesamt 117 Millionen Parameter. Das Modell wurde auf dem Datensatz *WebText* mit rund 40 GB Text aus dem Internet trainiert.

BERT

Das Modell BERT (*Bidirectional Encoder Representations from Transformers*) wurde von [Devlin et al. \(2019\)](#) eingeführt. Im Gegensatz zu unidirektionalen Modellen wie GPT-2, die lediglich

¹<https://code.google.com/archive/p/word2vec/>

auf den vorhergehenden Kontext zugreifen, basiert BERT auf einem bidirektionalen Transformer-Encoder. Dieser ermöglicht es, bei der Repräsentation eines Tokens gleichzeitig sowohl den linken als auch den rechten Kontext zu berücksichtigen. Auf diese Weise können tiefere syntaktische und semantische Abhängigkeiten innerhalb eines Satzes erfasst werden.

Für das Training von BERT wurde eine große Textmenge verwendet, bestehend aus dem *Books-Corpus* (800 Millionen Wörter) und der *English Wikipedia* (2,5 Milliarden Wörter).

In dieser Arbeit kommt die Variante *BERT base uncased* zum Einsatz. Sie umfasst 12 Encoder-Layer, nutzt Word Embeddings mit einer Dimension von 768 und ein Vokabular mit 30 522 Token. Insgesamt enthält das Modell rund 110 Millionen Parameter.

Pythia

Die Pythia-Modellreihe wurde von *EleutherAI*² als offene Plattform für systematische Untersuchungen an großen Sprachmodellen entwickelt und von [Biderman et al. \(2023\)](#) vorgestellt. Ziel des Projekts ist es, eine kontrollierte und vollständig dokumentierte Suite autoregressiver Sprachmodelle bereitzustellen, mit der Forschungsfragen zur Trainingsdynamik, Skalierung und zum Modellverhalten untersucht werden können.

Ein zentrales Merkmal der Pythia-Reihe ist die konsistente Trainingsstruktur: Alle Modelle wurden auf denselben Daten, in derselben Reihenfolge und mit identischer Architektur- sowie Hyperparameterstrategie trainiert. Dies erlaubt direkte Vergleiche entlang der Skalierungsdimension – von 70 Millionen bis 12 Milliarden Parametern – ohne konfundierende Faktoren wie unterschiedliche Trainingsdaten oder Modellvarianten. Die verwendeten Daten stammen aus *The Pile* ([Gao et al., 2020](#)), einem öffentlich zugänglichen, kuratierten Korpus englischsprachiger Texte mit hoher thematischer Vielfalt.

Die Modelle basieren auf einem Decoder-only-Transformer-Design, vergleichbar mit GPT-2, nutzen jedoch einige moderne Erweiterungen. In der vorliegenden Arbeit wird das Modell *Pythia-410M* eingesetzt. Es weist folgende Spezifikationen auf: 24 Transformer-Blöcke, 1 024 - dimensionale Word Embeddings bei einer Vokabulargröße von 50 257 Token sowie rund 410 Millionen Gesamtparameter.

Dieses Modell stellt eine mittlere Größe innerhalb der Pythia-Suite dar und wurde – wie alle Modelle der Reihe – auf etwa 300 Milliarden Tokens trainiert. Die Architektur und das Training erfolgten mit Hilfe der Open-Source-Bibliothek *GPT-NeoX*, was eine vollständige Reproduzierbarkeit sicherstellt.

Auswirkung von Weight Tying und der Vorverarbeitung durch TransformerLens auf die Anzahl der zu analysierenden Matrizen der Sprachmodelle

Ein zentraler Aspekt bei der Auswahl der Transformer-Modelle für das Experiment war die Frage, ob das jeweilige Modell Weight Tying verwendet – also ob die Unembedding-Matrix lediglich die Transponierte der Embedding-Matrix ist. Diese Eigenschaft beeinflusst direkt die Anzahl der zu untersuchenden Matrizen, da bei geteilten Gewichten die Analyse auf eine gemeinsame Struktur reduziert werden kann.

In ihrer ursprünglichen Architektur nutzen GPT-2 und BERT Weight Tying, während Pythia über getrennte Embedding- und Unembedding-Matrizen verfügt. Während des Experiments stellte sich jedoch heraus, dass die zum Laden der Modelle verwendete Bibliothek *TransformerLens*³ standardmäßig bestimmte Vorverarbeitungsschritte an den Embedding- und Unembedding-Gewichten durchführt. Dadurch kommt es vor, dass auch Modelle, die ursprünglich Weight Tying implementieren, nach dem Laden intern mit getrennten Matrizen dargestellt werden.

²<https://github.com/EleutherAI>

³<https://transformerlens.org.github.io/TransformerLens/>

Diese Vorverarbeitung umfasst unter anderem das sogenannte *Centering der Embedding-Matrix (center_writing_weight)*⁴. Dabei wird von jeder Zeile der Embedding-Matrix der Mittelwert ihrer Einträge subtrahiert. Dies ist möglich, weil alle Komponenten, die im Modell aus dem Residual Stream lesen, zuvor eine Layer Normalization anwenden – und diese entfernt ihrerseits den Mittelwert des Eingabevektors. Die Mittelwertkomponente ist daher nicht relevant für die Berechnung und kann ohne Auswirkung auf das Modellverhalten entfernt werden. Dieser Schritt vereinfacht jedoch die spätere Analyse linearer Strukturen.

Beim *Centering der Unembedding-Matrix (center_unembed)*⁵ wird von jeder Spalte der Matrix der Mittelwert aller Ausgabewerte abgezogen. Diese Anpassung hat keinen Einfluss auf die Ausgabe des Modells, da die Unembedding-Matrix in eine Softmax-Funktion eingespeist wird. Da Softmax invariant gegenüber gleichmäßiger Verschiebung aller Eingabewerte ist, bleiben die resultierenden Wahrscheinlichkeiten unverändert. Auch dieser Schritt dient primär der mathematischen Vereinfachung bei der Analyse.

Durch diese Vorverarbeitung entstehen jeweils neue Varianten der ursprünglichen Modelle, die im Rahmen dieser Arbeit als *cleaned* bezeichnet werden. TransformerLens bietet jedoch die Möglichkeit, das Preprocessing gezielt zu deaktivieren. Dadurch konnten auch die Originalgewichte der Modelle untersucht werden; diese Varianten werden im Folgenden als *raw* bezeichnet.

Je nach Kombination von ursprünglichem Weight Tying und dem Vorverarbeitungsstatus (*cleaned* oder *raw*) wurde festgelegt, welche Matrizen bei welchem Modell analysiert werden. Ein wichtiger Hinweis betrifft hierbei das Modell BERT: Die Vorverarbeitungsschritte von TransformerLens sind aktuell nur für Decoder-Modelle implementiert, weshalb BERT ausschließlich in seiner ursprünglichen, unmodifizierten Form untersucht wurde.

Die folgende Tabelle fasst die relevanten Modelle, ihre Konfigurationen sowie die jeweils untersuchten Matrizen zusammen:

Eigenschaft	Word2Vec	GPT-2 cleaned	GPT-2 raw	Pythia cleaned	Pythia raw	BERT
Vokabulargröße	3 Mio	50 257	50 257	50 257	50 257	30 522
Vektor-Dimensionalität	300	768	768	1024	1024	768
Anzahl Parameter	300 Mio	117 Mio	117 Mio	410 Mio	410 Mio	110 Mio
Anzahl Layer	-	12	12	24	24	12
Weight Tying	-	nein	ja	nein	nein	ja
untersuchte Matrizen	Embeddings	Embeddings Unembeddings	Embeddings	Embeddings Unembeddings	Embeddings Unembeddings	Embeddings

Tabelle 3.1.: Vergleich zentraler Eigenschaften der in diesem Experiment untersuchten Modelle.

3.2. Datenbasis

Für das Experiment wurde der *Bigger Analogy Test Set (BATS)* verwendet (Gladkova et al., 2016). Im Gegensatz zu früheren, oft unausgewogenen Benchmarks wie dem Google Analogy Dataset, zeichnet sich BATS durch eine ausgewogene Abdeckung unterschiedlicher sprachlicher Relativenstypen sowie eine hohe Anzahl an Einzelbeispielen aus.

Der BATS-Datensatz umfasst insgesamt 99 200 Analogiefragen zu den 40 sprachlichen Subkategorien, die sich in vier größere Kategorien unterteilen. Die detaillierte Struktur des Datensatzes ist in der Tabelle A.1 abgebildet.

Jede Subkategorie enthält 50 Wortpaare, die im Analogietest zu $\binom{50}{2} = \frac{50 \cdot 49}{2} = 1,225$ mögliche Analogiefragen pro Richtung, also insgesamt $2 \cdot 1,225 = 2,450$ Fragen pro Subkategorie kombiniert werden.

⁴https://github.com/TransformerLensOrg/TransformerLens/blob/main/further_comments.md#centering-writing-weights-center_writing_weight

⁵https://github.com/TransformerLensOrg/TransformerLens/blob/main/further_comments.md#centering-writing-weights-center_writing_weight

Ein wesentliches Ziel bei der Erstellung von BATS war es, Ambiguitäten durch Homonyme zu reduzieren – insbesondere in den morphologischen Kategorien. So wurden beispielsweise alle Wörter ausgeschlossen, die laut WordNet mehreren Wortarten zugeordnet sind. Gleichzeitig wurde die Tokenfrequenz im Korpus als Kriterium für die Auswahl der 50 Wortpaare innerhalb jeder Kategorie verwendet, um sicherzustellen, dass ausreichend häufige Wörter zur Evaluation verwendet werden. In bestimmten Fällen erlaubt BATS mehrere richtige Antworten, was insbesondere für hyponymische oder hyperonymische Relationen sinnvoll ist (Gladkova et al., 2016).

Im Rahmen dieser Arbeit wurde eine modifizierte Version des ursprünglichen BATS-Datensatzes erstellt, die ausschließlich auf den Token basiert, die in allen untersuchten Modellen gemeinsam vorkommen. Diese Variante wird im Folgenden als *BATS_shared* bezeichnet.

Die gefilterte Version *BATS_shared* umfasst insgesamt 34 Subkategorien mit 848 gültigen Vergleichspaaren. Diese lassen sich zu insgesamt 29 088 möglichen Analogiefragen kombinieren. Die genaue Verteilung der Vergleichspaare und daraus resultierenden Analogiefragen pro Subkategorie ist in Tabelle A.2 dargestellt.

3.3. Experimentelle Durchführung

Laden und Vorverarbeitung der Modelle

Für die Untersuchung kamen vier Sprachmodelle zum Einsatz: *GPT-2*, *BERT*, *Pythia* und *Word2Vec*. Die Transformer-Modelle wurden mithilfe der Python-Bibliothek *TransformerLens* geladen. TransformerLens bietet eine modulare Infrastruktur zur Analyse, Manipulation und Visualisierung neuronaler Sprachmodelle auf der Ebene einzelner Layer, Token und Gewichtsmatrizen. Es eignet sich besonders gut für den gezielten Zugriff auf Embedding- und Unembedding-Matrizen, da es eine einheitliche Schnittstelle zur Modellarchitektur bereitstellt. Beim Laden der Transformer-Modelle führt TransformerLens sämtliche Preprocessing-Schritte an den Embedding- und Unembedding-Matrizen durch, die im Unterkapitel 3.1 im Detail beschrieben sind.

Das *Word2Vec*-Modell diente in diesem Experiment als Referenzmodell (Baseline). Es wurde über die Python-Bibliothek *Gensim* geladen. Gensim eignet sich insbesondere für die Arbeit mit vortrainierten binären Vektorformaten und bietet darüber hinaus robuste Werkzeuge zur Verarbeitung von Wortvektoren.

Datenbasis- und Token-Normalisierung

Um eine faire Vergleichbarkeit zwischen den Modellen sicherzustellen, wurde eine gemeinsame Vokabularbasis konstruiert.

Hierzu wurden zunächst die Tokenlisten aller untersuchten Modelle extrahiert. Da die Modelle unterschiedliche Tokenisierungssysteme verwenden, war eine Normalisierung der Vokabulare erforderlich, um eine vergleichbare Grundlage zu schaffen.

Die Normalisierungsmaßnahmen wurden modellabhängig angepasst: Bei *GPT-2* und *Pythia* wurde jeweils das führende Leerzeichen entfernt und alle Tokens in Kleinschreibung überführt. Für *Word2Vec* erfolgte ebenfalls eine Umwandlung in Kleinschreibung. Im Fall von *BERT* wurden ausschließlich vollständige Tokens berücksichtigt; Subtoken-Anteile mit dem Präfix „#“ wurden verworfen.

Zusätzlich wurde für alle Modelle geprüft, ob ein Token ausschließlich aus alphabetischen Zeichen besteht und eine Mindestlänge von drei Zeichen aufweist.

Auf Grundlage der so bereinigten und normalisierten Vokabulare wurde die Schnittmenge der Token berechnet: Insgesamt konnten 14 181 gemeinsame Tokens identifiziert werden. Diese wurden alphabetisch sortiert und in der Datei *shared_tokens.txt* gespeichert, die als Grundlage für die weitere Datenfilterung diente.

Anschließend wurde der ursprüngliche BATS-Datensatz auf jene Analogiepaare reduziert, bei denen alle vier beteiligten Wörter (a, a^*, b, b^*) in der Liste *shared_tokens* enthalten sind. Auf diese

Weise wurde sichergestellt, dass ausschließlich solche Beispiele in die Evaluation einfließen, die von allen untersuchten Modellen abgebildet werden können.

Extraktion der gefilterten Submatrizen

Für jedes der untersuchten Modelle wurden die relevanten Vektormatrizen extrahiert, die als Grundlage für die Analogie-Tests dienten. Bei den Transformer-Modellen (GPT-2, BERT, Pythia) wurden ausschließlich die statischen Token-Embeddings sowie die Unembedding-Matrizen berücksichtigt, die modellarchitektonisch am Eingang bzw. Ausgang der Transformer-Schichten liegen. Für das Word2Vec-Modell stand nur eine einzige Embedding-Matrix zur Verfügung.

Die extrahierten Matrizen wurden im Anschluss auf das gemeinsame Token-Vokabular *shared_tokens* reduziert: Es wurden nur die Vektoren derjenigen Tokens übernommen, die von allen Modellen repräsentiert werden konnten. Anschließend wurden die gefilterten Matrizen mit Hilfe der Python-Library Gensim in Binärformat überführt, sodass sie für die Analogieauswertung genutzt werden konnten.

Durchführung der Analogie-Tests

Die Evaluation erfolgte mit der Python-Bibliothek *Vecto*⁶, die speziell für die systematische Untersuchung von Vektorraum-Modellen entwickelt wurde. Vecto stellt ein modulares Framework bereit, mit dem sich Analogie-Tests auf Basis verschiedener Rechenverfahren automatisiert durchführen und vergleichen lassen. Für jeden Vektorsatz wurden Analogieaufgaben mithilfe von vier Verfahren berechnet: *OnlyB* (ein Verfahren, das lediglich den Vektor zurückgibt, der am nächsten zu Wort *b* liegt), *3CosAdd*, *3CosAvg* und *PairDistance*.

Die linearen Verfahren sind in Kapitel 2.2.1 im Detail erläutert. Dabei fungierte *OnlyB* im Rahmen des Experiments als Baseline unter den Rechenverfahren. Auf jede der neun extrahierten Submatrizen wurden die vier Verfahren separat angewendet, sodass insgesamt 36 vollständige Analogie-Experimente durchgeführt wurden. Die Ergebnisse wurden im JSON-Format gespeichert und enthalten verschiedene Metriken, auf die im nächsten Kapitel eingegangen wird.

3.4. Evaluationsergebnisse und -methoden

Die Ergebnisse der Analogie-Tests wurden in separaten JSON-Dateien gespeichert, wobei jede Datei ein einzelnes Experiment repräsentiert. Ein solches Experiment ist durch eine spezifische Kombination aus Modell und Analogie-Methode definiert. Der Ordner mit den Vecto-Ergebnissen *anomaly_results_pythia_cleaned* enthält beispielsweise folgende JSON-Dateien:

- pythia_cleaned_embs_3CosAdd
- pythia_cleaned_embs_3CosAvg
- pythia_cleaned_embs_OnlyB
- pythia_cleaned_embs_PairDistance
- pythia_cleaned_unembs_3CosAdd
- pythia_cleaned_unembs_3CosAvg
- pythia_cleaned_unembs_OnlyB
- pythia_cleaned_unembs_PairDistance

Dabei steht *embs* für die Embedding-Matrix und *unembs* für die Unembedding-Matrix.

⁶<https://vecto.space/>

Jede JSON-Datei enthält Ergebnisse sowohl auf Makro- als auch auf Mikroebene. Auf Makroebene werden Metriken pro Subkategorie gespeichert, darunter:

- Anzahl der Fragen (*cnt_questions_total*),
- Anzahl der korrekt beantworteten Fragen (*cnt_questions_correct*),
- durchschnittliche Accuracy,
- Metadaten zum Modell und Datensatz.

Auf Mikroebene – also pro einzelner Analogieaufgabe – unterscheiden sich die verfügbaren Metriken je nach verwendeter Methode. Gemeinsam dokumentieren alle Verfahren:

- die vollständige Aufgabenstellung (*question_verbose*),
- das Zielwort (*expected_answer*),
- eine geordnete Liste der Modellvorhersagen mit zugehörigen Scores (Kosinus-Ähnlichkeit zum richtigen Wort) und *hit*-Markierungen,
- den Rang des korrekten Zielworts (*rank*).

Darüber hinaus dokumentieren die Methoden wie 3CosAdd und PairDistance zusätzliche, spezifische Metriken, darunter Kosinus-Ähnlichkeiten zwischen Wortpaaren (*similarity a to a_prime*, *similarity b to b_prime*), euklidische Distanzen und einige weitere Nachbarschaftsrelationen.

Diese Unterschiede spiegeln die konzeptionellen Grundlagen der jeweiligen Methoden wider: Während 3CosAdd und PairDistance auf expliziten Vektorrechnungen zwischen Wortpaaren basieren und dadurch vielfältige Zwischenwerte berechenbar machen, arbeiten OnlyB und 3CosAvg mit vereinfachten Annahmen oder Mittelwerten, die weniger komplexe Zwischeninformationen erzeugen.

Die Auswertung der Analogieergebnisse erfolgte in Anlehnung an das etablierte Verfahren in früheren Studien zu wortanalogie-basierten Evaluierungen von Wortvektormodellen (Gladkova et al., 2016; Drozd et al., 2016; Rogers et al., 2017). Bei der Berechnung der mittleren Accuracy wurde wie folgt vorgegangen: Die durchschnittliche Accuracy pro Subkategorie wurde aus den JSON-Dateien extrahiert, anschließend wurde für jede der vier Hauptkategorien (*Inflectional Morphology*, *Derivational Morphology*, *Encyclopedic Semantics*, *Lexicographic Semantics*) der ungewichtete Mittelwert der Subkategorien berechnet. Jede Subkategorie ging dabei mit gleichem Gewicht in den Kategoriewert ein, unabhängig von der Anzahl der enthaltenen Aufgaben. Schließlich wurde für jedes Modell ein globaler Accuracy-Wert berechnet, indem die vier Hauptkategorien erneut ungewichtet gemittelt wurden.

Nach demselben Prinzip wurde auch die mittlere Kosinus-Ähnlichkeit pro Kategorie berechnet. Dabei wurde ausschließlich die Ähnlichkeit derjenigen Modellvorhersagen berücksichtigt, die korrekt waren. Dieser Fokus auf erfolgreiche Vorhersagen dient dazu, die Präzision der semantischen Projektion zu erfassen – also zu beurteilen, wie nah das Modell im Vektorraum tatsächlich an das erwartete Ziel heranführt, wenn es die Analogie korrekt löst.

Dabei muss angemerkt werden, dass Kosinus-Ähnlichkeit durch Vecto Library normalisiert wird: In der zugrundeliegenden Implementierung wird zunächst der standardisierte Wert zwischen zwei Vektoren im Bereich [-1,1] berechnet, anschließend jedoch in den Wertebereich [0,1] transformiert.

4. Analyse der Ergebnisse und Fazit

Eine erste Auswertung der Accuracy-Werte über verschiedene Sprachmodelle hinweg, aggregiert nach den vier BATS-Hauptkategorien (*Inflectional Morphology*, *Derivational Morphology*, *Encyclopedic Semantics*, *Lexicographic Semantics*), zeigt deutliche Unterschiede in der Leistungsbeurteilung zwischen den eingesetzten Analogie-Methoden. Diese Unterschiede sind exemplarisch in Abbildung 4.1 am Beispiel der Modell-Varianten *GPT-2 Raw Embeddings* und *GPT-2 Cleaned Unembeddings* veranschaulicht: Während Methoden wie 3CosAvg und 3CosAdd in nahezu allen Kategorien höhere Trefferquoten erzielen, schneiden Verfahren wie OnlyB und insbesondere PairDistance deutlich schlechter ab. Dabei bewerten die Methoden eher inkonsistent und liefern in Einzelfällen schlechtere Ergebnisse als die Baseline-Methode OnlyB (vgl. Abbildungen A.1, A.2, A.3).

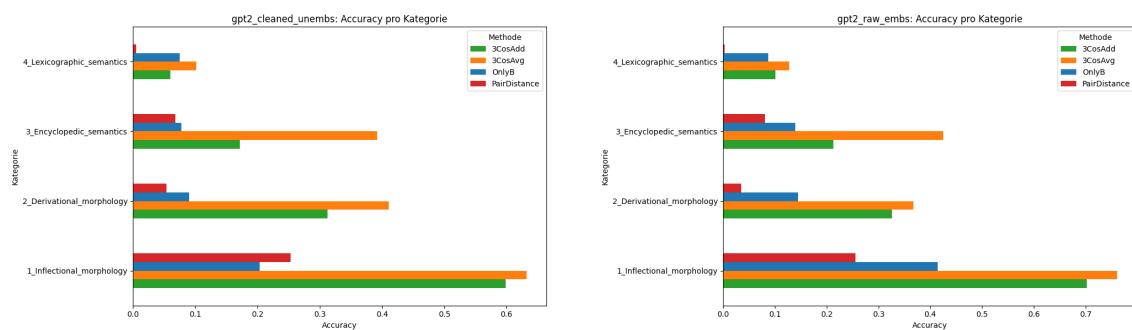


Abbildung 4.1.: Visualisierung der kategorialen Unterschiede in der Accuracy pro Kategorie für GPT-2 Raw Embeddings und GPT-2 Cleaned Unembeddings anhand der vier linearen Analogie-Verfahren

Aufgrund der erheblichen Varianz in der Bewertung sprachlicher Relationen, die durch die Wahl der Methode bedingt ist, wurde entschieden, die eingesetzten Verfahren vor dem eigentlichen Modellvergleich zu evaluieren. Ziel war es, eine Vergleichsbasis zu schaffen, die eine differenzierte und faire Beurteilung der Modellleistung über verschiedene sprachliche Phänomene hinweg ermöglicht.

4.1. Methoden im Vergleich

Die nachfolgende Evaluation der Analogie-Methoden berücksichtigt sowohl die mittlere Accuracy pro Modell als auch die Konsistenz der Ergebnisse innerhalb der verschiedenen Subkategorien. Für jede Modellvariante wurde dabei sowohl die *Top-1* als auch die *Top-5 Accuracy* berechnet. Während *Top-1 Accuracy* angibt, ob das richtige Wort an erster Stelle der Vorhersageliste steht, wird bei *Top-5 Accuracy* überprüft, ob es sich unter den fünf bestplatzierten Vorhersagen befindet.

Die mittlere Top-1 Accuracy pro Methode und Modell zeigt auf, wie stabil eine Methode modellübergreifend funktioniert. Die Genauigkeit pro Subkategorie liefert hingegen Hinweise darauf, ob eine Methode unterschiedliche linguistische Phänomene gleichermaßen verlässlich abbilden kann.

Die vergleichende Auswertung der mittleren Top-1 Accuracy pro Modell ist in Tabelle 4.1 dargestellt.

4. Analyse der Ergebnisse und Fazit

Modell	3CosAdd	3CosAvg	OnlyB	PairDistance
bert_embs	0.346	0.404	0.200	0.146
word2vec_embs	0.337	0.413	0.152	0.145
gpt2_cleaned_embs	0.319	0.344	0.200	0.086
gpt2_cleaned_unembs	0.286	0.384	0.112	0.095
gpt2_raw_embs	0.336	0.420	0.196	0.094
pythia_cleaned_embs	0.294	0.336	0.209	0.048
pythia_cleaned_unembs	0.355	0.410	0.192	0.116
pythia_raw_embs	0.296	0.347	0.205	0.050
pythia_raw_unembs	0.327	0.390	0.170	0.106

Tabelle 4.1.: Mittlere Top-1 Accuracy pro Modellvariante und Methode

Die Methode *PairDistance* zeigte stark unterdurchschnittliche Ergebnisse und konnte die Baseline-Methode *OnlyB* nicht übertreffen. Die Methoden *3CosAdd* und insbesondere *3CosAvg* erzielten dagegen deutlich bessere Resultate.

Auffällig ist, dass *3CosAvg* sowohl hinsichtlich der mittleren Accuracy über alle Modellvarianten als auch im Hinblick auf die sprachliche Breite – also die Robustheit über unterschiedliche Subkategorien hinweg – konsistent bessere Leistungen zeigt. Beim Vergleich der Top-1- und Top-5-Accuracy ist *3CosAvg* die einzige Methode, bei der eine deutliche Steigerung bei Erweiterung des Bewertungsbereichs erkennbar ist (siehe Abbildung 4.2).

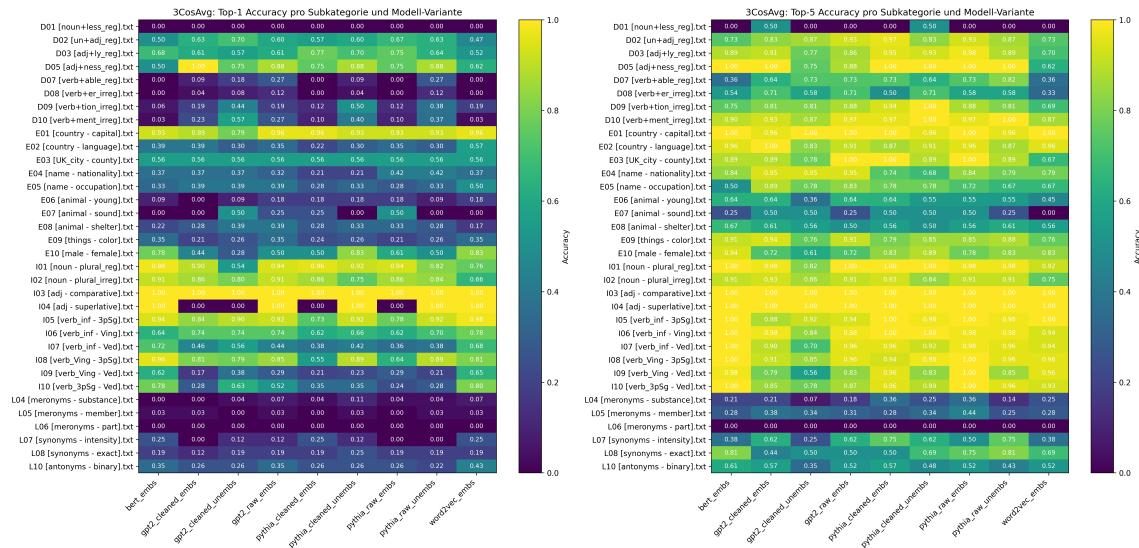


Abbildung 4.2.: 3CosAvg: Top-1- und Top-5-Accuracy pro Subkategorie und Modell-Variante

Bemerkenswert ist dabei, dass diese Leistungssteigerung nicht nur in morphologisch geprägten Subkategorien (z. B. Pluralformen oder Komparativbildung), sondern auch in semantisch komplexeren Kategorien wie *Encyclopedic Semantics* und *Lexicographic Semantics* sichtbar wird (siehe Abbildung 4.3.). Gerade in diesen semantischen Bereichen erzielt 3CosAvg bei Top-5-Accuracy teils deutlich höhere Werte als 3CosAdd – ein Hinweis auf die höhere Generalisierungsfähigkeit der Methode.

Die Heatmaps der anderen drei Methoden sind im Anhang dargestellt (vgl. Abbildungen A.4, A.5, A.6). Dabei wird ersichtlich, dass auch bei Top-5-Accuracy keine der Methoden an die Leistung von 3CosAvg heranreicht. 3CosAdd zeigt im Vergleich zur Baseline eine moderate Verbesserung, während OnlyB und PairDistance sowohl bei Top-1- als auch bei Top-5-Accuracy durch-

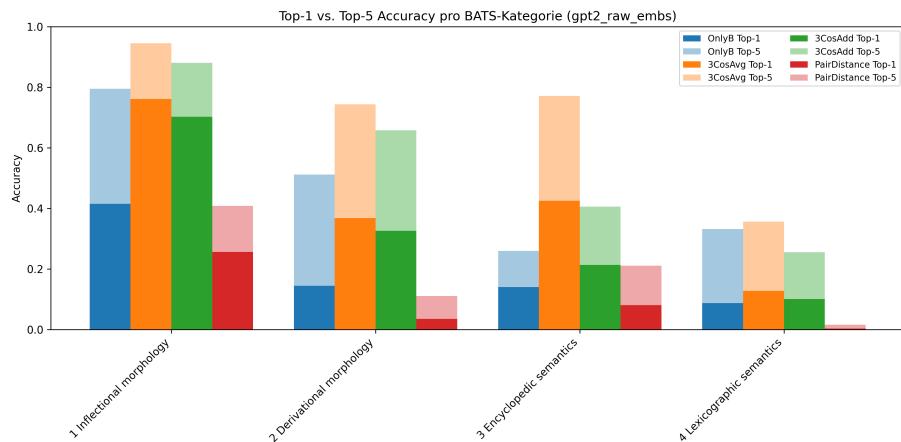


Abbildung 4.3.: Top-1 und Top-5 Accuracy pro Kategorie und Methode am Beispiel von GPT-2 Raw Embs

gängig schwache Resultate liefern.

Ein möglicher Grund für die bessere Performanz von *3CosAvg* liegt in seiner Robustheit gegenüber Ausreißern bei der Vektordifferenz einzelner Wortpaare. Durch die Mittelung der Offsets über alle Trainingspaare einer Subkategorie entsteht ein stabilerer semantischer Richtungsvektor, der die zugrunde liegende Relation verlässlicher abbildet. Aufgrund dieser Eigenschaften wurden in den folgenden Auswertungen alle Modellvarianten auf Basis von *3CosAvg* verglichen.

4.2. Modelle im Vergleich

Zur Bewertung der Modellleistung wurden zwei Metriken herangezogen: die *Top-1 Accuracy* und die *mittlere Kosinus-Ähnlichkeit* zur Zielvorhersage. In der vorliegenden Auswertung wurde die mittlere Kosinus-Ähnlichkeit ausschließlich für korrekte Vorhersagen berechnet, um gezielt die Präzision im Erfolgsfall zu analysieren. Diese Metrik ergänzt die Accuracy, indem sie zusätzlich Aufschluss über die Qualität der richtigen Vorhersagen gibt.

Die vergleichende Auswertung der mittleren Accuracy und der mittleren Kosinus-Ähnlichkeit pro Modell und BATS-Kategorie ist in Tabelle 4.2 dargestellt.

Modell	Derivational Morph.		Encyclopedic Sem.		Inflectional Morph.		Lexicographic Sem.	
	Accuracy	Cosine Similarity	Accuracy	Cosine Similarity	Accuracy	Cosine Similarity	Accuracy	Cosine Similarity
bert_embs	0.222	0.852	0.402	0.829	0.855	0.886	0.136	0.820
gpt2_cleaned_embs	0.350	0.813	0.353	0.819	0.606	0.856	0.070	0.819
gpt2_cleaned_unembs	0.411	0.754	0.393	0.768	0.633	0.802	0.102	0.696
gpt2_raw_embs	0.368	0.824	0.425	0.804	0.761	0.851	0.127	0.800
pythia_cleaned_embs	0.289	0.735	0.367	0.717	0.567	0.786	0.122	0.696
pythia_cleaned_unembs	0.402	0.753	0.395	0.742	0.714	0.813	0.129	0.712
pythia_raw_embs	0.299	0.740	0.436	0.722	0.573	0.784	0.081	0.691
pythia_raw_unembs	0.411	0.770	0.368	0.761	0.704	0.822	0.079	0.770
word2vec_embs	0.229	0.839	0.449	0.829	0.812	0.886	0.162	0.828

Tabelle 4.2.: Durchschnittliche Top-1 Accuracy und Kosinus-Ähnlichkeit pro Modellvariante und BATS-Kategorie (Methode: *3CosAvg*)

Die Tabelle zeigt, dass die Kategorie *Inflectional Morphology* über alle Modelle hinweg die höchsten Werte erzielt. Sowohl klassische Wortvektoren wie Word2Vec als auch Transformer-Modelle erreichen hier sehr gute Ergebnisse. Word2Vec, das in diesem Experiment als Referenzmodell fungiert, erzielt insgesamt eine starke Leistung, wird jedoch bei der Accuracy in dieser

Kategorie von BERT übertroffen (0,855). Beide Modelle zeigen hier zudem die höchsten Kosinus-Ähnlichkeiten aller Varianten, was auf eine besonders präzise Repräsentation morphologischer Muster hindeutet.

Diese Beobachtung spricht dafür, dass flexionsmorphologische Relationen – etwa Pluralbildung oder Tempusänderungen – relativ regelmäßig und formgebunden sind und sich daher besonders gut durch lineare Strukturen im Vektorraum erfassen lassen.

Im Kontrast dazu erweist sich die Kategorie *Lexicographic Semantics* als besonders herausfordernd: Die Top-1 Accuracies liegen hier durchweg unter 0,2, teils sogar unter 0,1. Auch die Kosinus-Ähnlichkeiten sind in dieser Kategorie am geringsten ausgeprägt. Dies deutet darauf hin, dass Bedeutungsrelationen wie Synonymie oder Antonymie von den getesteten Modellen nur unzureichend abgebildet werden können. Dieses Muster bestätigt frühere Befunde, wonach semantisch subtile Relationen eine besondere Herausforderung für vektorraumbasierte Modelle darstellen (Gladkova et al., 2016; Rogers et al., 2017). Bemerkenswert ist, dass Word2Vec auch in dieser schwierigen Kategorie mit einer Accuracy von 0,162 und einer Kosinus-Ähnlichkeit von 0,828 alle Transformer-Modelle übertrifft.

In der Kategorie *Encyclopedic Semantics* zeigen alle Modelle eher durchschnittliche Leistungen. Auch hier erzielt Word2Vec im Vergleich die höchsten Werte, wenngleich der Abstand zu den Transformer-Modellen geringer ausfällt.

Etwas differenzierter stellt sich das Bild in der Kategorie *Derivational Morphology* dar. Hier zeigen insbesondere einige Transformer-Modelle eine bessere Fähigkeit, Ableitungsrelationen wie *[adj+ness], [verb+er, irreg]* oder *[verb+tion, irreg]* abzubilden. Die besten mittleren Top-1 Accuracies erreichen dabei die *GPT-2 Cleaned Unembeddings* sowie die *Pythia Raw Unembeddings*. Hinsichtlich der Präzision, gemessen an der Kosinus-Ähnlichkeit, zeigt jedoch *BERT* die beste Leistung.

Insgesamt lässt sich feststellen, dass Word2Vec in vielen Kategorien entweder die höchste Leistung erzielt oder sehr nah an den besten Transformer-Varianten liegt. Dies deutet darauf hin, dass die höhere Dimension der Embeddings in Transformer-Modellen nicht zwangsläufig zu einer besseren Abbildung linguistischer Relationen führt. Die Ergebnisse unterstreichen vielmehr, dass keine Modellarchitektur universell überlegen ist, sondern ihre Eignung maßgeblich von der Art der sprachlichen Relation abhängt.

Embeddings vs. Unembeddings

Die Embedding- und Unembedding-Matrizen wurden im Rahmen desselben Modells evaluiert und nicht modellübergreifend verglichen. Abbildung 4.4 zeigt die durchschnittliche Top-1 Accuracy sowie die mittlere Kosinus-Ähnlichkeit für drei ausgewählte Modellvarianten: *GPT-2 Cleaned*, *Pythia Raw* und *Pythia Cleaned*.

Für alle drei Modelle zeigt sich ein konsistenter Leistungsvorteil der Unembedding-Matrizen gegenüber den jeweiligen Embedding-Matrizen. Besonders ausgeprägt ist dieser Unterschied bei den Pythia-Modellen. Die höheren Top-1-Accuracy-Werte deuten darauf hin, dass Unembeddings linguistische Regularitäten systematischer erfassen. Auch die Kosinus-Ähnlichkeiten fallen bei den Unembeddings tendenziell höher aus, wobei GPT-2 in dieser Hinsicht eine Ausnahme bildet. Der positive Effekt der Unembeddings bleibt dabei auch nach dem Preprocessing durch TransformerLens erhalten.

Diese Ergebnisse legen nahe, dass Unembedding-Matrizen besser geeignet sind, linguistische Analogien im Vektorraum abzubilden – unabhängig davon, ob das Modell einer Vorverarbeitung unterzogen wurde oder nicht.

Cleaned vs. Raw

Die im Rahmen dieser Arbeit eingesetzten Modelle GPT-2 und Pythia wurden sowohl in ihrer Originalform (*Raw*) als auch in vorverarbeiteter (*Cleaned*) Variante analysiert. Das Preprocessing

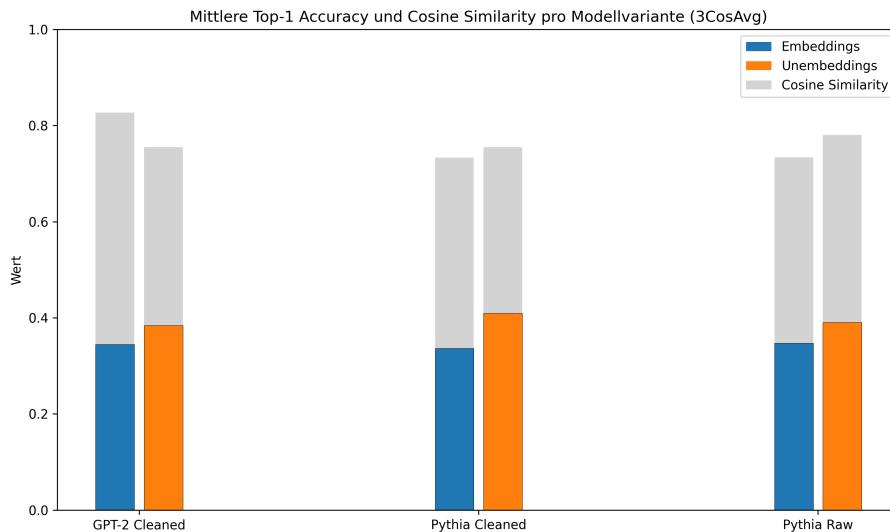


Abbildung 4.4.: Embeddings vs. Unembeddings: Durchschnittliche Top-1 Accuracy und Kosinus-Ähnlichkeit pro Modell

erfolgte durch die Bibliothek TransformerLens, die standardmäßig bestimmte Transformationen an den Embedding- und Unembedding-Matrizen durchführt. Die Preprocessing-Schritte sind im Unterkapitel 3.1 ausführlich beschrieben. Obwohl diese Operationen das Verhalten der Modelle bei Inferenz nicht verändern, beeinflussen sie die geometrische Struktur des Vektorraums – und damit die Ergebnisse der Analogie-Aufgaben. Tabelle 4.3 vergleicht die mittlere Accuracy und Kosinus-Ähnlichkeit zwischen *Raw* und *Cleaned* Matrizen in den vier BATS-Kategorien.

Modell	Matrix	Kategorie	Accuracy		Cosine Similarity	
			Raw	Cleaned	Raw	Cleaned
GPT-2	Embedding	Derivational Morph.	0.368	0.350	0.824	0.813
		Encyclopedic Sem.	0.425	0.353	0.804	0.819
		Inflectional Morph.	0.761	0.606	0.851	0.856
		Lexicographic Sem.	0.127	0.070	0.800	0.819
Pythia	Embedding	Derivational Morph.	0.299	0.289	0.740	0.735
		Encyclopedic Sem.	0.436	0.367	0.722	0.717
		Inflectional Morph.	0.573	0.567	0.784	0.786
		Lexicographic Sem.	0.081	0.122	0.691	0.696
Pythia	Unembedding	Derivational Morph.	0.411	0.402	0.770	0.753
		Encyclopedic Sem.	0.368	0.395	0.761	0.742
		Inflectional Morph.	0.704	0.714	0.822	0.813
		Lexicographic Sem.	0.079	0.129	0.770	0.712

Tabelle 4.3.: Einfluss des Preprocessings durch TransformerLens auf Accuracy und Cosine Similarity bei Embedding- und Unembedding-Matrizen von GPT-2 und Pythia.

Beim Vergleich der Embedding-Matrizen von GPT-2 zeigt sich, dass das Centering durch TransformerLens zu einer systematischen Verschlechterung der Accuracy in allen getesteten linguistischen Kategorien führt. Besonders ausgeprägt ist dies bei flexionsmorphologischen Aufgaben (Accuracy: 0.761 vs. 0.606) und bei lexikografischen Relationen (0.127 vs. 0.070). Die Kosinus-Ähnlichkeiten bleiben hingegen weitgehend stabil oder steigen minimal an.

Ein ähnliches Muster zeigt sich auch bei den Pythia-Embeddings: Auch hier schneiden die raw-Varianten meist leicht besser ab als die cleaned-Varianten – etwa in Encyclopedic Semantics (0.436 vs. 0.367). Allerdings sind die Unterschiede weniger stark ausgeprägt als bei GPT-2. Dies könnte darauf hindeuten, dass die Pythia-Embeddings robuster gegenüber dem Entfernen der Mittelwertkomponente sind oder von vornherein eine andere Vektorraumstruktur aufweisen.

Im Fall der Unembeddings fällt der Unterschied zwischen Cleaned und Raw anders aus. In Derivational Morphology erreicht Pythia Raw Unembeddings eine leicht höhere Accuracy als die cleaned-Variante (0.411 vs. 0.401), in drei anderen Kategorien jedoch ist die cleaned-Variante besser. Auffällig ist allerdings, dass die Kosinus-Ähnlichkeit in allen vier Kategorien bei der raw-Variante höher ist, was auf eine stärkere semantische Ausrichtung der rohen Unembedding-Vektoren hinweist.

Insgesamt zeigt sich, dass das Preprocessing durch TransformerLens nicht neutral im Hinblick auf Analogieaufgaben ist: Die durchgeführten Vorverarbeitungsoperationen verändern die Vektorstruktur in einer Weise, die lineare Relationen im semantischen Raum beeinträchtigen kann. Die ursprünglichen raw-Matrizen scheinen in Analogieaufgaben daher zuverlässiger und semantisch ausdrucksstärker zu sein als die vorverarbeiteten cleaned-Varianten. Besonders deutlich fällt dieser Effekt bei den Embeddings von GPT-2 aus, während die Pythia-Matrizen eine höhere Robustheit gegenüber dem Preprocessing zeigen.

4.3. Diskussion der Ergebnisse und methodische Einschränkungen

Die Reduktion der ursprünglichen Vokabulare der Modelle auf eine gemeinsame Schnittmenge von 14 181 Tokens stellt eine zentrale methodische Einschränkung dieses Experiments dar. Während diese Maßnahme eine faire Vergleichbarkeit der Modelle ermöglichen sollte, hatte sie modellabhängig sehr unterschiedliche Auswirkungen in Bezug auf Ergebnisse der Analogie-Tests.

Ein besonders relevanter Eingriff in diesem Zusammenhang war das systematische Lowercasing aller Tokens. Diese Entscheidung war notwendig, um die Überschneidung der Vokabulare zwischen den Modellen zu maximieren. Obwohl bei der Erstellung des BATS-Datensatzes Ambiguitäten in den morphologischen Kategorien durch den Ausschluss von Homonymen gezielt reduziert wurden, kann ein semantischer Informationsverlust durch das systematische Lowercasing in den semantischen Kategorien nicht ausgeschlossen werden. Gerade dort könnten Unterschiede zwischen Eigennamen und gewöhnlichen Nomen oder zwischen Homonymen erhalten geblieben sein und die Modellleistung beeinträchtigt haben.

Die Auswirkungen der Normalisierung waren allerdings nicht für alle Modelle gleich. Word2Vec wurde besonders stark getroffen: Aus einem ursprünglich dreimillionenstarken Vokabular blieb nur rund 0,5% erhalten. Dies könnte dazu führen, dass die verbleibenden Testpaare die Stärken des Modells verzerrt wiedergeben. BERT war hingegen als *uncased*-Modell bereits auf lowercased Eingaben trainiert und besitzt zudem ein kompakteres Vokabular, sodass die Effekte der Normalisierung hier deutlich geringer ausfielen.

Hinzu kommt, dass insbesondere die semantisch komplexeren Kategorien wie *Encyclopedic Semantics* und *Lexicographic Semantics* der Datenbasis BATS_shared zusätzlich stark vom Vokabularfiltering betroffen waren. Viele ursprüngliche Analogiepaare fielen weg, wodurch diese Subkategorien teils nur noch aus wenigen Items bestanden (siehe Tabelle A.2). Diese geringe Datenbasis mindert die Aussagekraft der Resultate erheblich.

In der Kategorie *Inflectional Morphology* hatten alle getesteten Modellvarianten überdurchschnittlich hohe Performanz. Sowohl klassische Wortvektoren wie Word2Vec als auch neuere Transformer-Modelle wie BERT und GPT-2 erreichen hier vergleichsweise hohe Accuracy-Werte. Gleichzeitig ist zu berücksichtigen, dass diese Ergebnisse auch durch die hohe Anzahl erhaltener Testpaare begünstigt sein könnten. Nach dem Filtern auf das gemeinsame Vokabular sind in *Inflectional Morphology* vergleichsweise viele valide Analogien erhalten geblieben, was eine stabilere

statistische Auswertung ermöglicht. In anderen Kategorien war dies nicht der Fall, was die Vergleichbarkeit zwischen den Kategorien einschränkt.

Die Analyse der Embedding- und Unembedding-Matrizen innerhalb einzelner Modelle zeigte einen klaren Leistungsvorteil der Unembeddings, insbesondere bei den Pythia-Varianten. Diese Resultate stützen die Annahme, dass Unembedding-Matrizen systematischere strukturelle Informationen über Wortrelationen enthalten – ein Effekt, der auch nach dem Preprocessing durch TransformerLens bestehen bleibt. Gleichwohl ist zu betonen, dass diese Befunde nicht modellübergreifend, sondern jeweils im Rahmen derselben Architektur erhoben wurden. Ein direkter Vergleich von Embedding- oder Unembedding-Matrizen über verschiedene Modelle hinweg wäre methodisch nicht zulässig, da sich die Modelle in ihrer Tokenisierung, Vektordimension und Trainingsarchitektur stark unterscheiden. Zudem basiert auch diese Bewertung auf der reduzierten Tokenbasis, die möglicherweise bevorzugt solche Tokens enthält, die in Unembeddings stabiler repräsentiert sind.

4.4. Fazit

Ziel der vorliegenden Arbeit war es, die geometrischen Strukturen der Embedding- und Unembedding-Matrizen klassischer und moderner Sprachmodelle vergleichend zu analysieren. Die zentralen Forschungsfragen lauteten dabei:

1. Weisen die Embedding- und Unembedding-Matrizen von Transformer-Modellen (*GPT-2, BERT, Pythia 410m*) ähnliche semantische und syntaktische Vektorstrukturen auf wie klassische Modelle (Word2Vec)?
2. Können Analogie-Metriken wie *3CosAdd*, *PairDistance* und *3CosAvg* auf diese Matrizen sinnvoll angewendet werden, um linguistische Relationen zu identifizieren?
3. Inwiefern unterscheiden sich Transformer-Modelle in ihrer Fähigkeit, bestimmte Relationen zu kodieren, und wie variieren die Ergebnisse je nach Modellarchitektur und Matrizen-Typ (Embedding vs. Unembedding)?

Zur ersten Forschungsfrage lässt sich festhalten, dass Transformer-Modelle syntaktisch reguläre Relationen wie Plural- oder Tempusbildung ähnlich zuverlässig abbilden wie Word2Vec – teils mit leichten Vorteilen für das klassische Modell. In semantisch komplexeren Kategorien wie *Lexico-graphic Semantics* oder *Encyclopedic Semantics* erzielen alle Modelle hingegen nur durchschnittliche oder schwache Ergebnisse, wobei die Resultate innerhalb der Subkategorien stark streuen. Dies erschwert belastbare Aussagen über die Qualität der Modelle. Zwar übertreffen Transformer-Modelle wie BERT oder GPT-2 Raw Word2Vec in einzelnen Subkategorien, der Abstand bleibt jedoch gering, sodass kein eindeutiger Vorteil einer bestimmten Modellarchitektur erkennbar ist.

Zur zweiten Forschungsfrage zeigt sich, dass klassische Analogie-Metriken grundsätzlich geeignet sind, um relationale Strukturen in den untersuchten Matrizen zu identifizieren. Ihre Effektivität unterscheidet sich jedoch deutlich: Während *PairDistance* durchweg schwache Resultate liefert, zeigt *3CosAvg* die stabilste Leistung – sowohl bei der mittleren Accuracy als auch in Bezug auf die Robustheit gegenüber unterschiedlichen linguistischen Kategorien. Besonders bei der Top-5-Accuracy erweist sich *3CosAvg* als generalisierungsfähiger als *3CosAdd*, auch in semantisch anspruchsvollen Aufgaben. Daher wurde *3CosAvg* als bevorzugte Methode für den Vergleich der Modellvarianten herangezogen.

Im Hinblick auf die dritte Forschungsfrage ergab die Analyse, dass sich Transformer-Modelle sowohl hinsichtlich ihrer Fähigkeit zur Kodierung sprachlicher Relationen als auch in der Struktur ihrer internen Vektorrepräsentationen unterscheiden. So zeigen Unembedding-Matrizen tendenziell bessere Leistungen als Embeddings, insbesondere bei derivations- und flexionsmorphologischen Relationen. Zudem erwies sich das durch TransformerLens durchgeführte Preprocessing – etwa das Entfernen der Mittelwertkomponente – als nicht neutral: Bei einigen Modellen führ-

te es zu messbaren Leistungseinbußen. Die Unterschiede zwischen Raw- und Cleaned-Varianten verdeutlichen, dass auch scheinbar technische Transformationen die semantische Geometrie eines Modells erheblich beeinflussen können.

Zusammenfassend lässt sich festhalten, dass moderne Transformer-Modelle durchaus vergleichbare geometrische Regularitäten wie klassische Wortvektormodelle aufweisen. Diese fallen jedoch architektur- und methodenabhängig aus. Die Wahl der Repräsentation (Embedding vs. Unembedding), der verwendeten Analogie-Methode sowie das Preprocessing wirken sich deutlich auf die beobachtbaren Strukturen aus.

Ausblick

Ein vielversprechender Ansatz für zukünftige Untersuchungen wäre die Analyse von Embedding- und Unembedding-Matrizen innerhalb analog strukturierter Modelle wie der Pythia-Reihe. Da alle Pythia-Modelle mit identischen Daten, Architekturen und Trainingsparametern trainiert wurden, wären sie weniger stark von Einschränkungen wie der Vokabularnormalisierung betroffen – zumal sich solche Eingriffe dort gleichmäßig auf alle Modelle auswirken würden. Ergänzend wäre die Entwicklung eines ausgewogeneren Analogie-Datensatzes sinnvoll, der auch nach Vokabularfiltern eine repräsentative Verteilung linguistischer Relationen gewährleistet. Dies würde fundiertere Vergleiche zwischen Modellen ermöglichen und die Aussagekraft künftiger Analysen deutlich erhöhen.

Literaturverzeichnis

- Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.
- Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. [Pythia: A suite for analyzing large language models across training and scaling](#). In *International Conference on Machine Learning*, pages 2397–2430.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. [Enriching word vectors with subword information](#). *Transactions of the Association for Computational Linguistics*, 5:135–146.
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 4171–4186.
- Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. 2016. [Word embeddings, analogies, and machine learning: Beyond king - man + woman = queen](#). In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3519–3530, Osaka, Japan. The COLING 2016 Organizing Committee.
- J. R. Firth. 1957. A synopsis of linguistic theory 1930-1955. In Philological Society, editor, *Studies in Linguistic Analysis*, pages 1–32. Blackwell, Oxford.
- Louis Fournier, Emmanuel Dupoux, and Ewan Dunbar. 2020. [Analogies minus analogy test: measuring regularities in word embeddings](#).
- Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. [The pile: An 800gb dataset of diverse text for language modeling](#).
- Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. 2016. [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*, pages 8–15, San Diego, California. Association for Computational Linguistics.
- Zellig S. Harris. 1954. Distributional structure. *Word*, 10(2–3):146–162.
- Martin Joos. 1950. *Readings in Linguistics: The Development of Descriptive Linguistics in America Since 1925*. American Council of Learned Societies, New York.
- Daniel Jurafsky and James H. Martin. 2025. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*, 3rd edition. Online manuscript released January 12, 2025.

- Eric Kafe. 2019. [Fitting semantic relations to word embeddings](#). In *Proceedings of the 10th Global Wordnet Conference*, pages 228–237, Wrocław, Poland. Global Wordnet Association.
- Omer Levy and Yoav Goldberg. 2014. [Linguistic regularities in sparse and explicit word representations](#). In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 171–180, Ann Arbor, Michigan. Association for Computational Linguistics.
- Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. [Improving distributional similarity with lessons learned from word embeddings](#). *Transactions of the Association for Computational Linguistics*, 3:211–225.
- Tal Linzen. 2016. [Issues in evaluating semantic spaces using word analogies](#).
- Yash Mahajan, Matthew Freestone, Naman Bansal, Sathyanarayanan Aakur, and Shubhra Kantri Karmaker Santu. 2025. [Revisiting word embeddings in the llm era](#).
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. [Efficient estimation of word representations in vector space](#).
- Tomás Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. [Distributed representations of words and phrases and their compositionality](#). *CoRR*, abs/1310.4546.
- Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. [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*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.
- Kiho Park, Yo Joong Choe, and Victor Veitch. 2024. [The linear representation hypothesis and the geometry of large language models](#).
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. [GloVe: Global vectors for word representation](#). In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar. Association for Computational Linguistics.
- Molly Petersen and Lonneke van der Plas. 2023. [Can language models learn analogical reasoning? investigating training objectives and comparisons to human performance](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16414–16425, Singapore. Association for Computational Linguistics.
- Jordan Peterson, Thomas L. Griffiths, Matthew L. Newman, and Mark Steyvers. 2020. [Parallelograms revisited: Exploring the limitations of vector space models for simple analogies](#). In *Proceedings of the 42nd Annual Conference of the Cognitive Science Society (CogSci)*, pages 1834–1840, Toronto, Canada. Cognitive Science Society.
- Ofir Press and Lior Wolf. 2017. [Using the output embedding to improve language models](#). In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. [Language models are unsupervised multitask learners](#). https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf. Technical report.
- Anna Rogers, Aleksandr Drozd, and Bofang Li. 2017. [The \(too many\) problems of analogical reasoning with word vectors](#). In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 135–148, Vancouver, Canada. Association for Computational Linguistics.

- David E. Rumelhart and Alan A. Abrahamson. 1973. [A model for analogical reasoning](#). *Cognitive Psychology*, 5(1):1–28.
- Natalie Schluter. 2018. [The word analogy testing caveat](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 242–246, New Orleans, Louisiana. Association for Computational Linguistics.
- Hinrich Schütze. 1992. [Dimensions of meaning](#). In *Proceedings of the 1992 ACM/IEEE Conference on Supercomputing*, pages 787–796.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems 30*.

Abbildungsverzeichnis

2.1. CBOW- und Skip-Gram-Architektur (Mikolov et al., 2013a)	4
2.2. Die Architektur eines (left-to-right) Transformers. Sie zeigt, wie jedes Eingabetoken kodiert, durch eine Abfolge gestapelter Transformer-Blöcke weiterverarbeitet und schließlich über einen Language-Modeling-Head zur Vorhersage des nächsten Tokens genutzt wird (Jurafsky and Martin, 2025, S.184).	11
4.1. Visualisierung der kategorialen Unterschiede in der Accuracy pro Kategorie für GPT-2 Raw Embeddings und GPT-2 Cleaned Unembeddings anhand der vier linearen Analogie-Verfahren	19
4.2. 3CosAvg: Top-1- und Top-5-Accuracy pro Subkategorie und Modell-Variante . .	20
4.3. Top-1 und Top-5 Accuracy pro Kategorie und Methode am Beispiel von GPT-2 Raw Embs	21
4.4. Embeddings vs. Unembeddings: Durchschnittliche Top-1 Accuracy und Kosinus-Ähnlichkeit pro Modell	23
A.1. BERT und Word2Vec: Accuracy pro Kategorie anhand der vier linearen Methoden	37
A.2. Pythia Cleaned: Accuracy pro Kategorie anhand der vier linearen Methoden . .	37
A.3. Pythia Raw: Accuracy pro Kategorie anhand der vier linearen Methoden	37
A.4. Accuracy-Heatmaps für 3CosAdd	38
A.5. Accuracy-Heatmaps für OnlyB	39
A.6. Accuracy-Heatmaps für PairDistance	40
A.7. BERT und Word2Vec: Top-1- und Top-5-Accuracy pro Kategorie	41
A.8. GPT-2 Cleaned Embs und Unembs: Top-1- und Top-5-Accuracy pro Kategorie .	41
A.9. Pythia Cleaned Embs und Unembs: Top-1- und Top-5-Accuracy pro Kategorie . .	41
A.10. Pythia Raw Embs und Unembs: Top-1- und Top-5-Accuracy pro Kategorie . . .	41

Tabellenverzeichnis

3.1. Vergleich zentraler Eigenschaften der in diesem Experiment untersuchten Modelle.	15
4.1. Mittlere Top-1 Accuracy pro Modellvariante und Methode	20
4.2. Durchschnittliche Top-1 Accuracy und Kosinus-Ähnlichkeit pro Modellvariante und BATS-Kategorie (Methode: 3CosAvg)	21
4.3. Einfluss des Preprocessings durch TransformerLens auf Accuracy und Cosine Similarity bei Embedding- und Unembedding-Matrizen von GPT-2 und Pythia.	23
A.1. Kategorien und Subkategorien im BATS-Datensatz mit Beispielen nach Gladkova et al. (2016).	35
A.2. Abdeckung der Subkategorien im BATS_shared-Datensatz: Anzahl der Vergleichspaare und resultierenden Analogiefragen.	36

A. Ergänzendes Material

A.1. Kategorien und Subkategorien der Datenbasis

Subcategory Analogy structure and examples		
Inflections	Nouns	I01: regular plurals (student:students) I02: plurals - orthographic changes (wife:wives)
	Adjectives	I03: comparative degree (strong:stronger) I04: superlative degree (strong:strongest)
	Verbs	I05: infinitive: 3Ps.Sg (follow:follows) I06: infinitive: participle (follow:following) I07: infinitive: past (follow:followed) I08: participle: 3Ps.Sg (following:follows) I09: participle: past (following:followed) I10: 3Ps.Sg : past (follows:followed)
		D01: noun+less (life:lifeless) D02: un+adj. (able:unable) D03: adj.+ly (usual:usually) D04: over+adj./Ved (used:overused) D05: adj.+ness (same:sameness) D06: re+verb (create:recreate) D07: verb+able (allow:allowable)
		D08: verb+er (provide:provider) D09: verb+ation (continue:continuation) D10: verb+ment (argue:argument)
Derivation	Hypernyms	L01: animals (cat:feline) L02: miscellaneous (plum:fruit, shirt:clothes)
	Hyponyms	L03: miscellaneous (bag:pouch, color:white)
	Meronyms	L04: substance (sea:water) L05: member (player:team)
		L06: part-whole (car:engine)
	Synonyms	L07: intensity (cry:scream) L08: exact (sofa:couch)
	Antonyms	L09: gradable (clean:dirty) L10: binary (up:down)
Lexicography	Geography	E01: capitals (Athens:Greece) E02: country:language (Bolivia:Spanish) E03: UK city:county York:Yorkshire
	People	E04: nationalities (Lincoln:American) E05: occupation (Lincoln:president)
	Animals	E06: the young (cat:kitten) E07: sounds (dog:bark) E08: shelter (fox:den)
		E09: thing:color (blood:red)
		E10: male:female (actor:actress)
	Other	

Tabelle A.1.: Kategorien und Subkategorien im BATS-Datensatz mit Beispielen nach Gladkova et al. (2016).

Kategorie	Subkategorie	Paare	Tests
Inflectional morphology	I01 [noun - plural_reg]	50	2450
	I02 [noun - plural_irreg]	44	1892
	I03 [adj - comparative]	4	12
	I04 [adj - superlative]	2	2
	I05 [verb_inf - 3pSg]	49	2352
	I06 [verb_inf - Ving]	50	2450
	I07 [verb_inf - Ved]	50	2450
	I08 [verb_Ving - 3pSg]	47	2162
	I09 [verb_Ving - Ved]	48	2256
	I10 [verb_3pSg - Ved]	46	2070
Derivational morphology	D01 [noun+less_reg]	2	2
	D02 [un+adj_reg]	30	870
	D03 [adj+ly_reg]	44	1892
	D05 [adj+ness_reg]	8	56
	D07 [verb+able_reg]	11	110
	D08 [verb+er_irreg]	24	552
	D09 [verb+tion_irreg]	16	240
	D10 [verb+ment_irreg]	30	870
Encyclopedic semantics	E01 [country - capital]	28	756
	E02 [country - language]	23	506
	E03 [UK_city - county]	9	72
	E04 [name - nationality]	19	342
	E05 [name - occupation]	18	306
	E06 [animal - young]	11	110
	E07 [animal - sound]	4	12
	E08 [animal - shelter]	18	306
	E09 [things - color]	34	1122
	E10 [male - female]	18	306
Lexicographic semantics	L04 [meronyms - substance]	28	756
	L05 [meronyms - member]	32	992
	L06 [meronyms - part]	4	12
	L07 [synonyms - intensity]	8	56
	L08 [synonyms - exact]	16	240
	L10 [antonyms - binary]	23	506

Tabelle A.2.: Abdeckung der Subkategorien im BATS_shared-Datensatz: Anzahl der Vergleichspaare und resultierenden Analogiefragen.

A.2. Kategorische Unterschiede in der Modell-Accuracy bei vier linearen Analogie-Verfahren

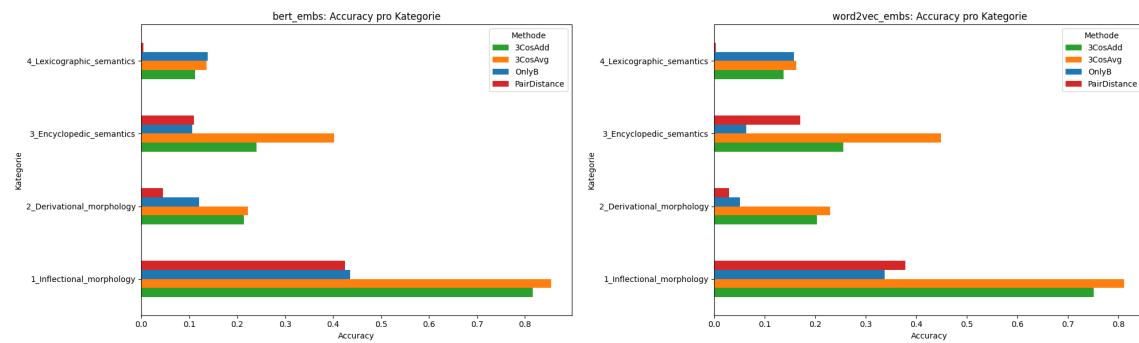


Abbildung A.1.: BERT und Word2Vec: Accuracy pro Kategorie anhand der vier linearen Methoden

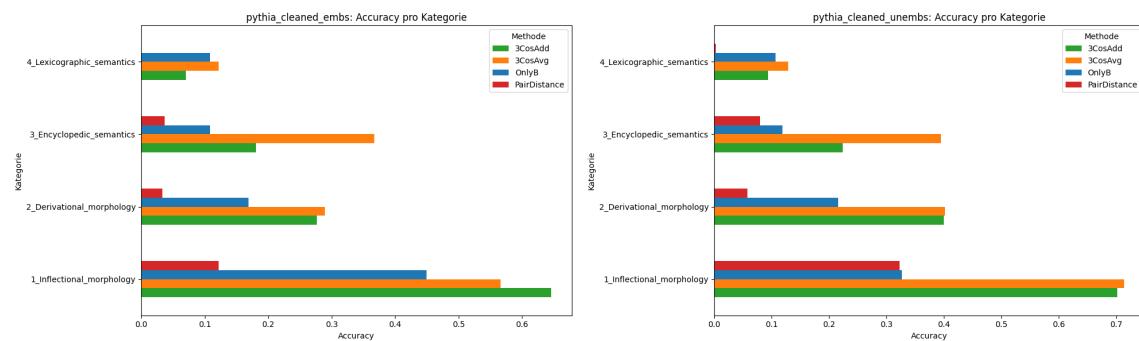


Abbildung A.2.: Pythia Cleaned: Accuracy pro Kategorie anhand der vier linearen Methoden

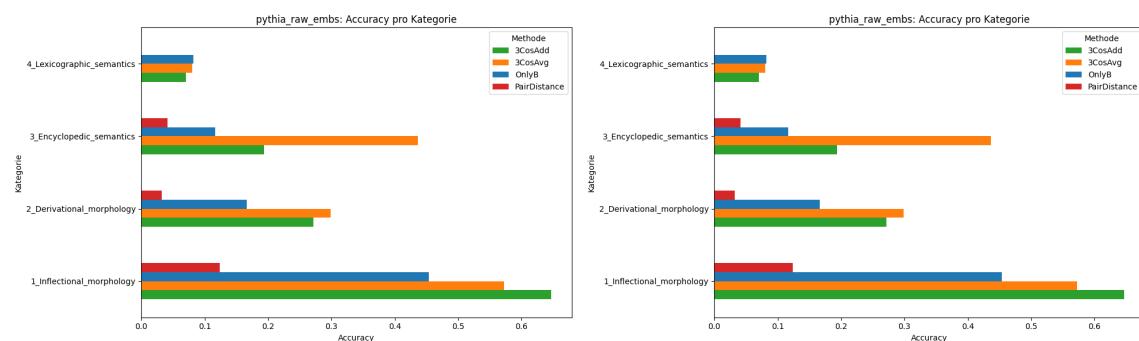


Abbildung A.3.: Pythia Raw: Accuracy pro Kategorie anhand der vier linearen Methoden

A.3. Top-1 und Top-5 Accuracy pro Methode, Subkategorie und Modellvariante

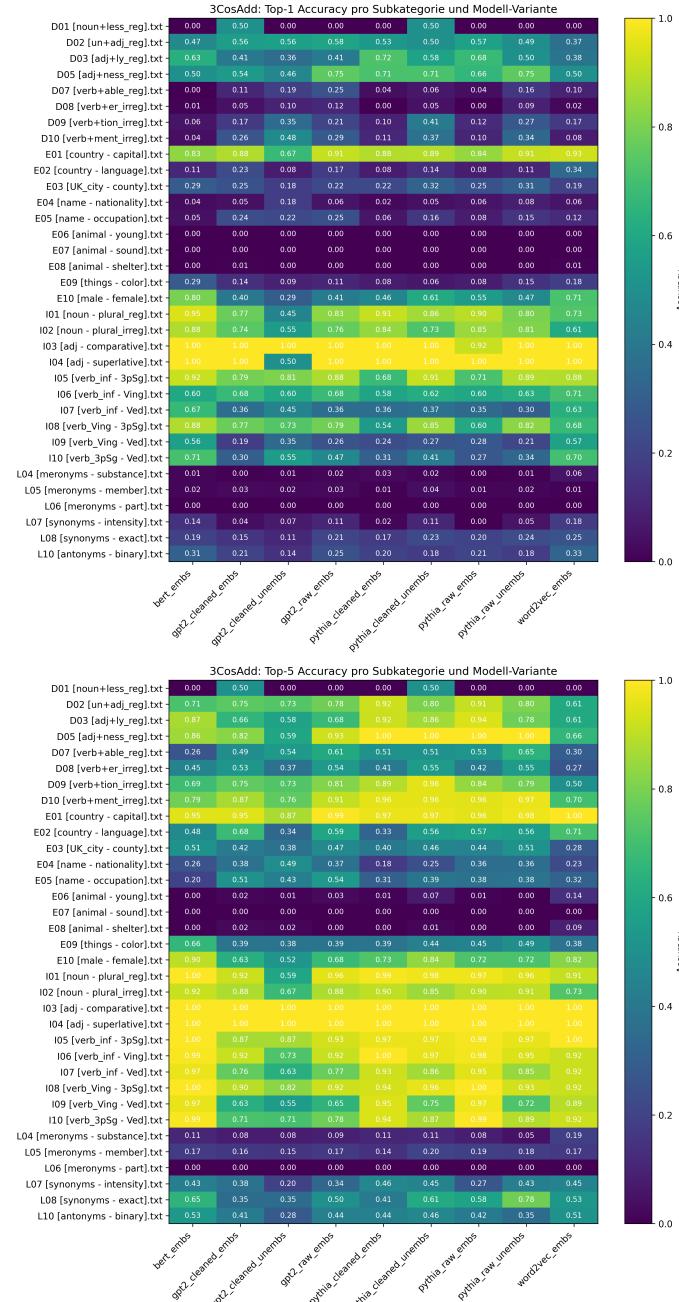


Abbildung A.4.: Accuracy-Heatmaps für 3CosAdd

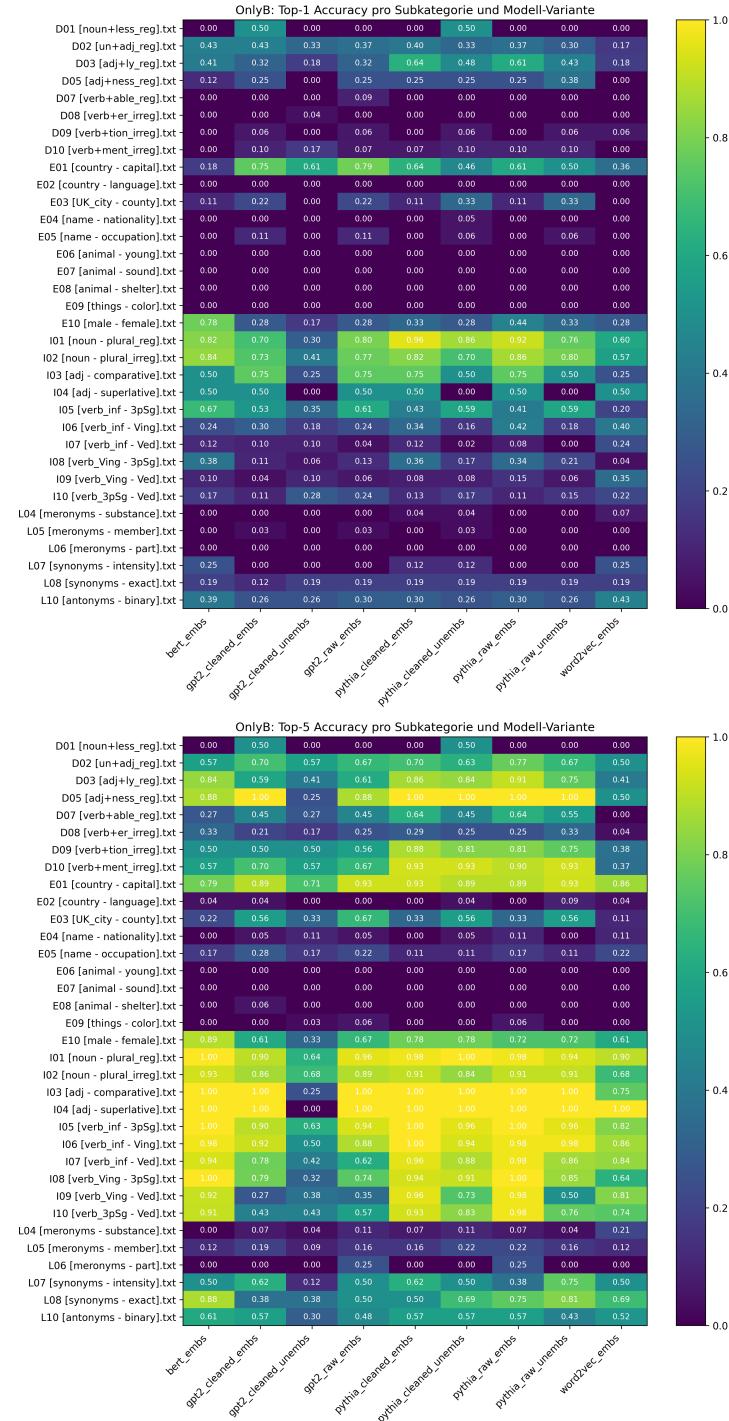


Abbildung A.5.: Accuracy-Heatmaps für OnlyB

A. Ergänzendes Material

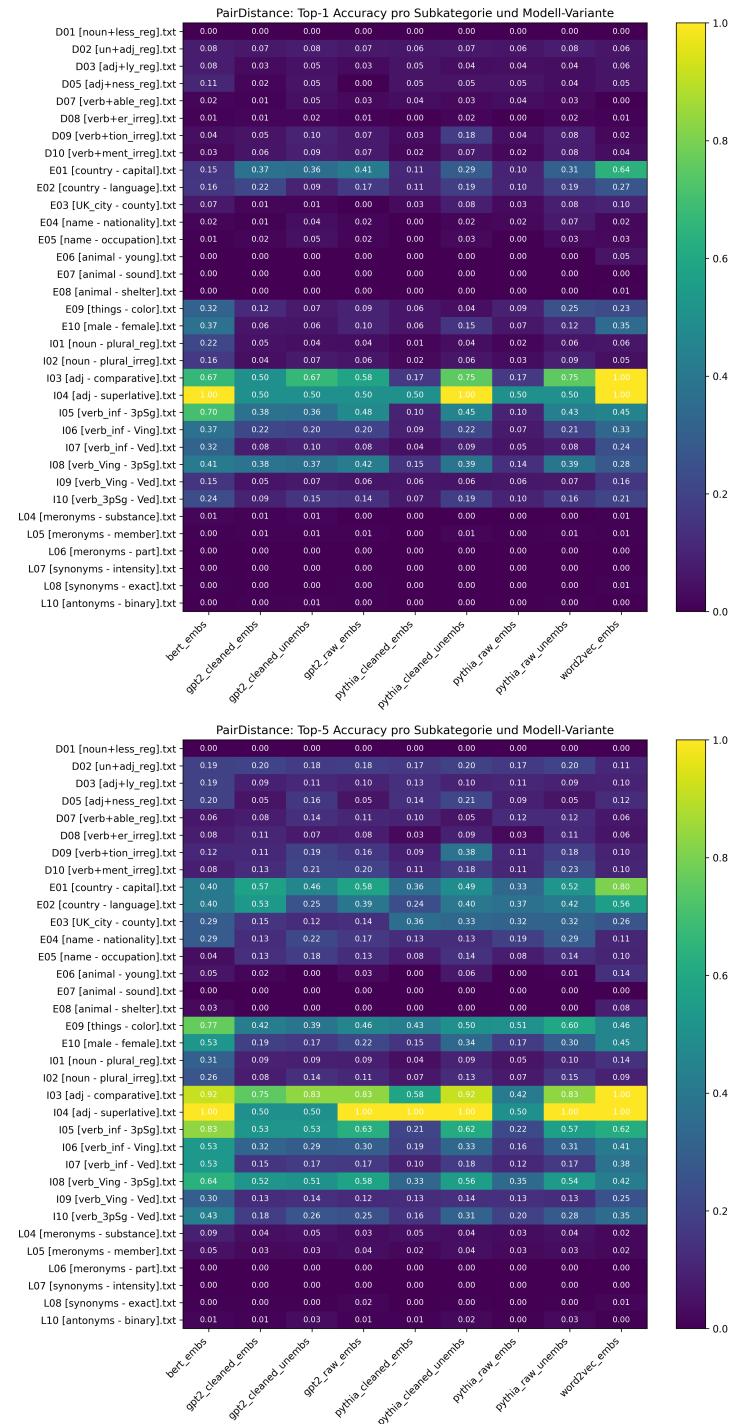


Abbildung A.6.: Accuracy-Heatmaps für PairDistance

A.4. Top-1 und Top-5 Accuracy pro Kategorie und Modellvariante

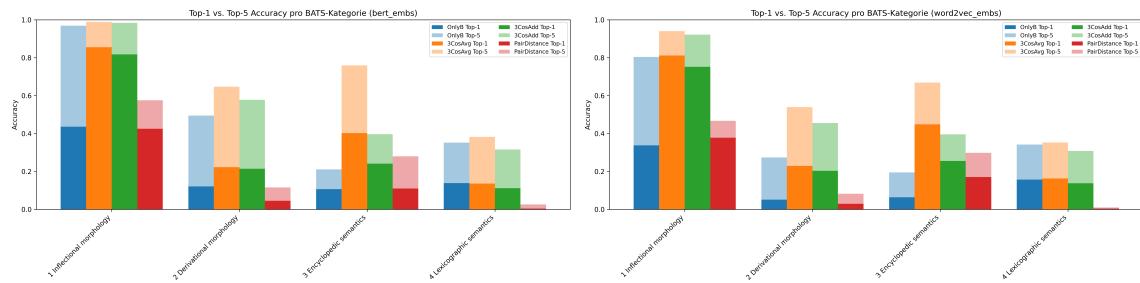


Abbildung A.7.: BERT und Word2Vec: Top-1- und Top-5-Accuracy pro Kategorie

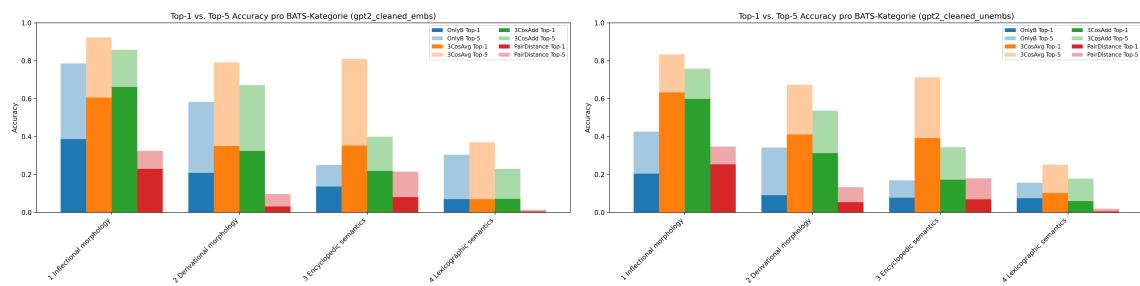


Abbildung A.8.: GPT-2 Cleaned Embs und Unembs: Top-1- und Top-5-Accuracy pro Kategorie

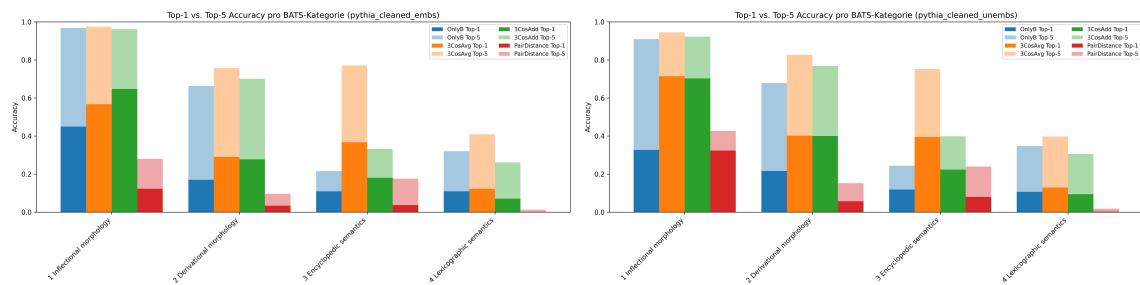


Abbildung A.9.: Pythia Cleaned Embs und Unembs: Top-1- und Top-5-Accuracy pro Kategorie

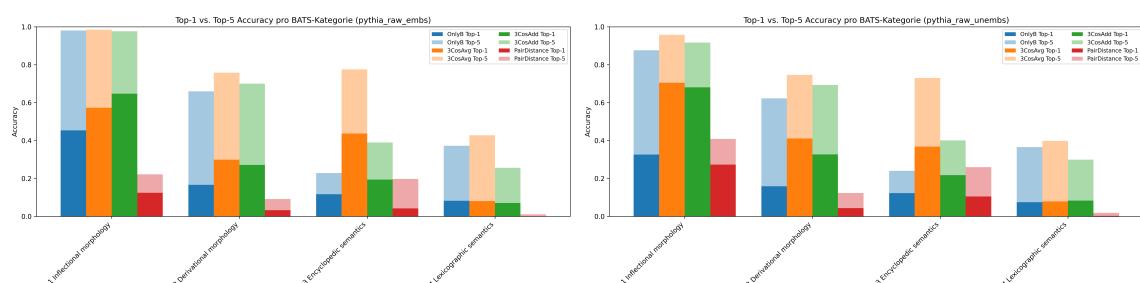


Abbildung A.10.: Pythia Raw Embs und Unembs: Top-1- und Top-5-Accuracy pro Kategorie

B. Eingereichte Software and Daten

Die im Rahmen dieser Arbeit entwickelte Software sowie alle relevanten Daten sind in einem GitHub-Repository¹ abgelegt.

Die Struktur des Repositories gliedert sich wie folgt:

- code / – Python-Skripte zur Extraktion der Embedding- und Unembedding-Matrizen, Durchführung der Analogie-Tests und Visualisierung der Ergebnisse
- data / – Gefilterter BATS-Datensatz, extrahierte Vektoren und JSON-Ergebnisse
- thesis.pdf – Finale Fassung dieser Bachelorarbeit
- README.md – Anleitung zur Replikation des Experiments

¹https://github.com/antoninagruzdeva/Bachelorarbeit_Vektor-Arithmetik.git

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig angefertigt, alle Zitate als solche kenntlich gemacht sowie alle benutzten Quellen und Hilfsmittel angegeben habe.

Als Hilfsmittel zur Korrektur von Rechtschreibung und Grammatik, zum Übersetzen einzelner Auszüge der zitierten wissenschaftlichen Artikel sowie zum Programmieren der Visualisierungsskripte wurde das KI-Tool ChatGPT eingesetzt.

München, den 11.06.2025

.....

Antonina Gruzdeva

