Philipp Schaer

C 9 Sprachmodelle und neuronale Netze im Information Retrieval

1 Einleitung

In den letzten Jahren haben Sprachmodelltechnologien unterschiedlichster Ausprägungen in der Informationswissenschaft Einzug gehalten. Diesen Sprachmodellen, die unter den Bezeichnungen GPT, ELMo oder BERT bekannt sind, ist gemein, dass sie dank sehr großer Webkorpora auf eine Datenbasis zurückgreifen, die bei vorherigen Sprachmodellansätzen undenkbar war. Gleichzeitig setzen diese Modelle auf neuere Entwicklungen des maschinellen Lernens, insbesondere auf künstliche neuronale Netze.

Diese Technologien haben auch im Information Retrieval (IR) Fuß gefasst und bereits kurz nach ihrer Einführung sprunghafte, substantielle Leistungssteigerungen erzielt. Neuronale Netze haben in Kombination mit großen vortrainierten Sprachmodellen und kontextualisierten Worteinbettungen dazu geführt, dass u. a. Lin et al. (2021) von nicht weniger als einem Paradigmenwechsel in der natürlichen Sprachverarbeitung (NLP) und dem IR sprechen. Wurde in vergangenen Jahren immer wieder eine stagnierende Retrievalleistung beklagt, die Leistungssteigerungen nur gegenüber „schwachen Baselines“ aufwies (Armstrong et al. 2009; Yang et al. 2019), so konnten mit diesen technischen und methodischen Innovationen beeindruckende Leistungssteigerungen in Aufgaben wie dem klassischen Ad-hoc-Retrieval, der maschinellen Übersetzung oder auch dem *Question Answering* erzielt werden (Trabelsi et al. 2021).

In diesem Kapitel soll ein kurzer Überblick über die Grundlagen der Sprachmodelle und der NN gegeben werden, um die prinzipiellen Bausteine zu verstehen, die hinter aktuellen Technologien wie ELMo oder BERT stecken, die die Welt des NLP und IR im Moment beherrschen.

2 Sprachmodelle und deren Anwendung

Sprachmodelle sind mathematische Modelle, die Wortfolgen Wahrscheinlichkeiten zuweisen und genutzt werden können, um bspw. vorherzusagen, welche Wörter auf eine gegebene Wortsequenz am wahrscheinlichsten folgen. Diese Modelle werden aus vorhandenen Texten abgeleitet und sind daher je nach Trainingskorpus unterschiedlich, z. B. in Bezug auf Sprache oder Vokabulargröße. Sprachmodelle fassen also a-priori-Kenntnisse über Sprache und deren Verwendung zusammen und können Auskunft darüber geben, wann welche Wörter in einer bestimmten Situation genutzt werden und wie wahrscheinlich diese Nutzung ist. Traditionell gibt es zwei Herangehensweisen an die Erstellung von Sprachmodellen: statistische und wissensbasierte Verfahren. Bei den statistischen Verfahren wird ein Textkorpus ausgezählt und so die Wahrscheinlichkeit des Auftretens bestimmter Wörter ermittelt. Bei den wissensbasierten Sprachmodellen wird auf Expertenwissen zurückgegriffen, z. B. in der Form grammatikalischen Wissens.

https://doi.org/10.1515/9783110769043-039

2.1 N-Gramme

Sprachmodelle der einfachsten Form sind sog. n-Gramme, also eine Folge von n Wörtern (Jurafsky & Martin 2021, Kapitel 3). N-Gramm-Modelle können verwendet werden, um die Wahrscheinlichkeit des letzten Wortes eines n-Gramms in Abhängigkeit von den vorangegangenen Wörtern zu schätzen und auch, um ganzen Wortketten oder -sequenzen Wahrscheinlichkeiten zuzuweisen. Beispielhaft am Satz „Ich studiere in Köln Informationswissenschaft“ lauten die entsprechenden n-Gramme:

Unigramme: <Ich>, <studiere>, <in>, <Köln>, <Informationswissenschaft>
Bigramme: <Ich studiere>, <studiere in>, <in Köln>, <Köln Informationswissenschaft>
Trigramme: <Ich studiere in>, <studiere in Köln>, <in Köln Informationswissenschaft>

Mit Hilfe großer Webkorpora, können sehr umfangreiche Sprachmodelle aufgebaut[1] und berechnet werden, welches das wahrscheinlichste Folgewort für eine gegebene Wortfolge ist. Vereinfachend kann man hierbei annehmen, dass die Wahrscheinlichkeit eines Wortes nicht von allen vorangegangenen, sondern näherungsweise nur von den letzten k-1 Wörtern abhängt. Bei einem Bigramm-Sprachmodell wird basierend auf dem vorherigen Wort w_{k-1} berechnet, wie hoch die Wahrscheinlichkeit eines Wortes w_k wäre, anstatt die gesamt vorherige Wortfolge zu betrachten:

$$P(w_k|w_1...w_{k-1}) \approx P(w_k|w_{k-1}) \quad (1)$$

Auf n-Grammen basierende Sprachmodelle haben die grundlegende Schwäche, dass nur der k Wörter umfassende Kontext erfasst wird und dieser in der Praxis meist nur wenige Wörter einbezieht. Gleichzeitig sind solche Sprachmodelle stark abhängig von den zugrundeliegenden Textkorpora, auf denen diese trainiert wurden. Passen die Trainingstexte z. B. hinsichtlich des verwendeten Vokabulars nicht zum Anwendungsfall, ist die zu erwartende Systemleistung gering.

2.2 *Bag of Words*

Im *Bag of Words*-Modell (BOW), das sowohl im NLP wie IR zur Anwendung kommt, werden Texte als sog. Multimenge verstanden. Im Gegensatz zur gewöhnlichen Menge aus der Mengenlehre können die Elemente einer Multimenge mehrfach vorkommen. Die Elemente sind unsortiert und jegliche Informationen über Grammatik oder Satzstellung werden ignoriert. Jedes Dokument kann so als Vektor dargestellt werden, der pro Term einen Eintrag enthält. Die einzelnen Dokumentvektoren können nun genutzt werden, um mit Hilfe geeigneter Maßzahlen Eigenschaften der Dokumente zu beschreiben, z. B. die Anzahl der jeweiligen Terme pro Dokument (die Termfrequenz TF, s. Kapitel C 2 Modelle im IR).

Termfrequenzen sind aber nicht die einzige und auch nicht zwangsläufig die sinnvollste Art der Repräsentation, da hierbei alltägliche, aber bedeutungsleere Terme überproportional häufig auftreten. Um dieses Problem zu umgehen, werden die Termhäufig-

1 Siehe z. B. das Web 1T 5-gram Korpus, https://catalog.ldc.upenn.edu/LDC2006T13.

keiten oft normalisiert, z. B. durch den Einsatz des TF-IDF-Gewichtungsverfahrens (s. Kapitel B 3 Automatisches Indexieren).

Die Einträge in der Term-Dokument-Matrix können nun genutzt werden, um z. B. Dokumente miteinander zu vergleichen (um ähnliche Dokumente zu identifizieren) oder um eine Anfrage, die als Pseudo-Dokument repräsentiert wird, mit den vorhandenen Dokumentvektoren abzugleichen.

Typischerweise wird im BOW nur mit einzelnen Termen gearbeitet, es ist aber prinzipiell auch möglich, das gleiche Modell mit n-Grammen zu befüllen. Die n-Gramme tragen dann eine gewisse Kontext-Information über die benachbarten Terme. Ein Problem des BOW und der Darstellung als Term-Dokument-Matrix ist, dass die besagte Matrix dünn besetzt ist und mit steigender Vokabulargröße |V| viele leere Einträge enthält.

Mit Blick auf das IR haben die bisher beschriebenen Ansätze das Problem, dass es zu einer fehlenden Überschneidung von Anfrage- und Dokumenttermen kommen kann und somit das Retrieval scheitert. Furnas et al. (1987, S. 964) nennen dies das „vocabulary problem“. Zur Lösung dieses Problems wird üblicherweise versucht, die Dokument- oder die Anfragerepräsentation zu erweitern, z. B. durch die Anwendung von Anfrageerweiterungen mit Hilfe kontrollierter Vokabulare und Thesauri wie WordNet (Miller 1995) oder korpusbasierter Statistiken wie z. B. Term-Autoren-Beziehungen (Schaer 2013). Diese Erweiterungen sind automatisch, z. B. in Form des Pseudo-Relevanz-Feedbacks (Cao et al. 2008), oder interaktiv, z. B. in Form von Vorschlagslisten (Hienert et al. 2011), möglich. Das zugrundeliegende Problem der terminologischen Nichtübereinstimmung wird dabei aber nicht grundlegend gelöst.

Mitra & Craswell (2018) beschreiben eine Reihe statistischer Funktionen, die im IR genutzt werden, um Verteilungsinformationen zu Termen bzw. Dokumenten als Grundlage für Anfragen an ein Retrievalsystem zu verwenden. Neben dem Vektorraummodell setzt auch das probabilistische Retrievalmodell BM25 auf TF-IDF auf (s. Kapitel C 2 Modelle im IR). Eine Alternative ist das Language-Model-Retrievalmodell (Zhai 2007). Jeder dieser Ansätze schätzt die Relevanz eines Dokuments zu einer Anfrage auf der Grundlage von Termhäufigkeiten und der Ähnlichkeit zu den Anfragetermen. Die Positionen der Terme im Dokument und deren semantische Beziehung werden allerdings ignoriert. Zwar gibt es Erweiterungen der Modelle, die mit der Nähe zwischen Termen (proximity) arbeiten können (Metzler 2011), jedoch führten diese Ansätze nicht zu durchschlagenden Erfolgen.

2.3 *Word Embeddings*

Worteinbettungen (*word embeddings*) sind kurze, bzw. dichte Wortrepräsentationen in Form von Vektoren, die im Vergleich zu den dünn besetzten Vektoren nicht für jeden Eintrag im Vokabular oder pro Dokument eine Dimension umfassen, sondern meist mit wenigen Hundert Dimensionen auskommen. Worteinbettungen repräsentieren dabei das ursprüngliche Wort in einem neuen Vektorraum, wobei aber die Eigenschaften des Wortes und seine Verbindungen zu anderen Wörtern bestmöglich bewahrt werden. Goodfellow et al. (2016) beschreiben, dass das Ziel einer Einbettung darin besteht, eine einfachere Darstellung zu erzeugen, wobei Vereinfachung eine Verringerung der Anzahl der Dimensionen bzw. eine knappere Darstellung und eine Entzerrung der Hauptkomponenten des Vektorraums oder eine Kombination dieser Ziele bedeuten kann.

Die Einträge dieser neuen Vektoren sind meist reelle Zahlen. Man spricht daher auch von *Continuous Bag of Words* (CBOW), das eines der Verfahren beschreibt, um Worteinbettungen zu generieren. Während die Interpretation der einzelnen Dimensionen selbst nicht mehr möglich ist, so zeigt sich doch in vielen NLP- und IR-Anwendungen eine deutliche Leistungssteigerung durch den Einsatz von Worteinbettungen (Ganguly et al. 2015). Obwohl noch nicht wirklich klar ist, warum diese Leistungssteigerungen messbar sind, so ist zu vermuten, dass dies mit den wesentlich geringeren Lernaufwänden und einem kleineren Parameterraum zusammenhängt, was folglich auch Probleme wie *Overfitting* vermeidet (Jurafsky & Martin 2021, Abschnitt 6.8). Vorteilhaft, z. B. für das Retrieval, ist die Nutzung der Vektorsemantik in Worteinbettungen, die bspw. Synonymen ähnliche Vektorrepräsentationen zuweist.

Eine der bekannteren Formen von Worteinbettungen ist unter dem Verfahren Word2Vec bekannt (Mikolov et al. 2013). Word2Vec basiert auf der Grundidee, dass wir nicht das gleichzeitige Auftreten eines Wortes w_1 in der Nähe eines anderen Wortes w_2 zählen (wie z. B. in den Term-Term-Matrizen), sondern dass ein binärer Klassifikator trainiert wird, um einzuschätzen, ob w_1 und w_2 gemeinsam auftreten. Der Klassifikator wird hierbei auf großen Textmengen trainiert und kommt dabei ohne manuelle Annotationen aus, da das tatsächliche gleichzeitige Auftreten von Wörtern im Text als korrekte Annotation gewertet wird. Word2Vec gestaltet diesen Lernprozess, der unter der Bezeichnung *Skip-gram* bekannt wurde, wie folgt: Zunächst werden ein Wort w_1 und ein benachbartes Wort w_2 als positive Beispiele ausgewählt. Gleichzeitig werden andere Wörter aus dem Vokabular ausgewählt und dienen als negative Beispiele. Mit Hilfe logistischer Regression wird zwischen dem positiven und den negativen Beispielen ein binärer Klassifikator trainiert, dessen gelernte Gewichtungsfaktoren dann als Worteinbettungen verwendet werden. Diese Wordeinbettungen sind statisch, d. h. die o. g. Methode lernt eine feste Einbettung für jedes Wort im Vokabular. Word2Vec setzt folglich auf dem alten Ansatz von Firth auf: „You shall know a word by the company it keeps!“ (Firth 1957, S. 11).

Neben Word2Vec ist auch das Verfahren GloVe verbreitet (Pennington et al. 2014). Während bei Word2Vec nur der lokale Kontext eines Wortes betrachtet wird, wird bei GloVe eine globale Textkorpusstatistik verwendet (daher auch der Name GloVe, der für *Global Vectors* steht). Da die Worteinbettungen vortrainiert werden können, können auch die Ergebnisse dieses Trainingsprozesses weitergegeben werden, sodass diese für viele Sprachen heruntergeladen und direkt nachgenutzt werden.[2]

2.4 Vektor-Semantik

Die Darstellung als Vektor erlaubt uns sowohl mit den hochdimensionalen, dünnbesetzten Vektoren, die auf beobachtbaren Eigenschaften basieren (wie z. B. Termhäufigkeiten oder -kookkurrenzen), als auch den dichten Vektoren der Worteinbettungen Distanzmaße zu bestimmen. Diese Distanzmaße können genutzt werden, um z. B. mit Hilfe einfacher Term-Statistik Retrieval durchzuführen. Der Retrievalprozess sieht nun einen Vergleich von Dokumentvektoren v_1 und Anfrage-Vektoren v_2 vor, bei dem bspw. auf die Kosinusähnlichkeit zurückgegriffen wird:

2 Die Open-Source-NLP-Bibliothek FastText beinhaltet z. B. 157 vortrainierte Modelle: https://fasttext.cc.

$$cosine(v_1, v_2) = \frac{v_1 \cdot v_2}{|v_1| \cdot |v_2|} \qquad (2)$$

Die Grundannahme hierbei ist, dass ähnliche Dokumente durch ähnliche Vektoren repräsentiert werden, da ähnliche Dokumente auch meist ähnliche Begriffe verwenden, wobei dieser Gedanke auf den Vergleich von Anfrage und Dokument analog übertragen wird.

Anstelle der zuvor beschriebenen Term-Dokument-Matrix lässt sich auch eine Term-Term-Matrix aufstellen. Hier wird für einen gegebenen Kontext das gemeinsame Auftreten zweier Terme hinterlegt. Ein Kontext kann z. B. das gemeinsame Auftreten innerhalb eines Dokumentes oder eines Satzes sein. Üblicherweise wird auch nur ein Fenster von n Wörtern im Text betrachtet, also n Wörter vor und nach dem jeweiligen Wort. Anstelle der Ähnlichkeit von Anfrage-Dokumentpaaren kann mit der o. g. Kosinusähnlichkeit auch die Ähnlichkeit von Termen bestimmt werden. Auch hier ist die Grundannahme, dass ähnlich Wörter ähnliche Term-Vektoren besitzen, da sie in einem ähnlichen Kontext auftreten.

Ein Beispiel hierzu ist in Abbildung 1 zusammengefasst. Für die Beispieldokumente aus Tabelle 1 werden sowohl eine Term-Dokument-, wie auch eine Term-Term-Darstellung gewählt. Die Term-Term-Darstellung basiert auf einer Kookkurrenz-Analyse, die auf der Auswertung des gleichzeitigen Auftretens von Termen im gleichen Dokument oder z. B. in der Form von n-Grammen basiert. Diese Art der Darstellung erlaubt es u. a., die Ähnlichkeit von Termen zu untersuchen, die nicht zwangsläufig im gleichen Dokument vorkommen. Die Vektor-Semantik erlaubt folgenden algebraischen Schluss: „Effzeh“ – „Köln“ + „Düsseldorf“ = „Fortuna“, da sich aus dieser einfachen Rechnung die größte Ähnlichkeit zwischen dem Ergebnisvektor und dem Vektor für den Term „Fortuna“ ergibt.

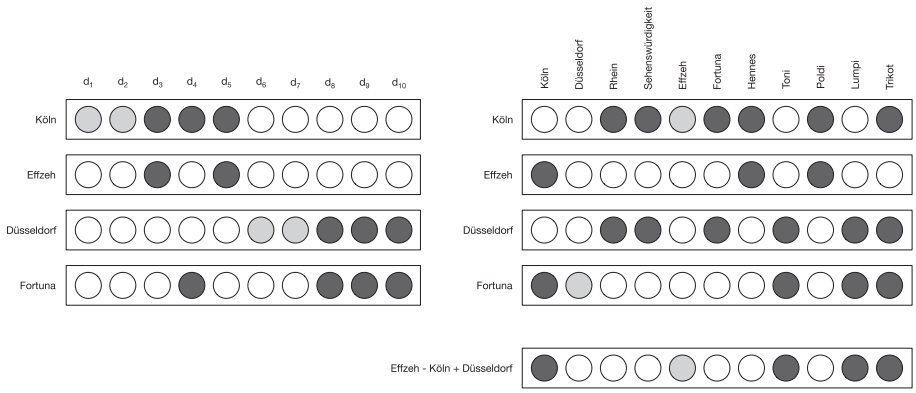

Abb. 1: Unterschiedliche Darstellung für die vier Term-Vektoren „Köln“, „Effzeh“, „Düsseldorf“ und „Fortuna“, einmal als Term-Dokument- (links) und als Term-Term-Darstellung (rechts): Hellgraue Kreise zeigen Werte an, die nicht Null sind und dunkelgraue Kreise solche, die zusätzlich auch in anderen Vektoren vorkommen

Tab. 1: Beispielkorpus bestehend aus kurzen Dokumenten

d_1	Köln Sehenswürdigkeiten	d_6	Düsseldorf Sehenswürdigkeiten
d_2	Köln Rhein	d_7	Düsseldorf Rhein
d_3	Effzeh Köln Hennes	d_8	Fortuna Düsseldorf Toni
d_4	Fortuna Köln Trikot	d_9	Fortuna Düsseldorf Trikot
d_5	Effzeh Köln Poldi	d_{10}	Fortuna Düsseldorf Lumpi

3 Neural IR – Deep Learning im Information Retrieval

Neben den angewandten Sprachmodellen in Form von Worteinbettungen sind in den letzten Jahren besonders Verfahren mit NN im IR populär geworden, die die Grundlage für das sog. Deep Learning bilden (Mitra & Craswell 2018). Neuronale IR-Modelle setzen auf eine Vektordarstellung der Dokumente und werden meist über sehr viele Parameter optimiert, wobei für das dort verwendete maschinelle Lernen und die hohe Anzahl an Parametern große Mengen von Trainingsdaten vorteilhaft und notwendig sind. Im Gegensatz zu maschinellen Lernverfahren in Form von *Learning to Rank* (s. Kapitel C 2 Modelle im IR), die auf einer großen Menge an händisch-definierten Eigenschaften (*features*) basieren, können aktuelle neuronale IR-Ansätze ohne solches Feature-Engineering auskommen und rein mit den Anfrage- und Dokumenttexten arbeiten. Zum Erlernen der passenden Vektordarstellungen benötigen diese Ansätze dafür aber sehr große Mengen an Trainingsdaten (Mitra et al. 2017). Hierfür können entweder überwachte oder unüberwachte Lernverfahren eingesetzt werden, also solche Verfahren, die mit entsprechenden ausgezeichneten Anfrage-/Dokumentpaaren arbeiten oder ohne diese auskommen und dann nur auf den Anfragen und/oder Dokumenten aufsetzen. Hier zeigen sich Gemeinsamkeiten zu *Latent Semantic Analysis* (Deerwester et al. 1990), in der Vektordarstellungen für Terme und Dokumente mit Hilfe der Singulärwertzerlegung bestimmt werden. Neuronale Ansätze erweitern diese Grundidee mit modernen Ansätzen und damals nicht verfügbaren Datenmengen und Rechenkapazitäten. Das Hauptziel der neuronalen Ansätze ist es, bessere Textdarstellungen (im Vergleich zu bekannten dünn besetzten Termvektoren) zu finden, die unempfindlicher sind gegen Störfaktoren wie unterschiedliche Textlängen, große Mengen von nicht-relevanten Textpassagen und den allgegenwärtigen Diskrepanzen in den Anfrage- und Dokumentvokabularen.

3.1 Neuronale Netze

Neuronale Netze für Informationssysteme basieren auf der Funktionsweise biologischer neuronaler Netze bzw. deren Bestandteilen, den Neuronen. Da hier das biologische Vorbild nachgeahmt wird, spricht man auch üblicherweise von künstlichen neuronalen Netzen (KNN). Die zum Netz gehörenden Neuronen lernen über eine Trainingsphase hinweg, indem Gewichte von Neuronen abgeändert werden oder Schwellenwerte zur Aktivierung eines Neurons angepasst werden (Goldberg 2017). Durch immer wieder erneutes Durchlaufen des Netzes mit Trainingsdaten können iterativ oder rekursiv die Parameter des KNN angepasst bzw. gelernt werden, damit sie bestmöglich den erwarteten Ausgabe-

werten entsprechen. Die Neuronen selbst haben meist nur sehr einfache Funktionen und die Mächtigkeit eines KNN ergibt sich aus der Verknüpfung und dem Zusammenspiel sehr vieler dieser kleinen Bauteile und deren Verknüpfung.

KNN können in unterschiedlichen Strukturen aufgebaut werden. Hierbei ist zum einen von Bedeutung, wie viele Neuronen sich auf wie vielen sog. Schichten befinden, und wie diese miteinander verbunden sind. Die Schichten (*layers*) sind dabei das häufigste Modell der Neuronenanordnung. Diese werden hintereinander angeordnet und sequentiell abgearbeitet. Es gibt einschichtige Netze (wie beispielhaft in Abbildung 3 zu sehen) und mehrschichtige Netze. Die jeweils trainierbaren Schichten, die vor der Ausgabeschicht liegen, werden als verdeckte Schichten bezeichnet. Mit genug Neuronen pro Schicht können selbst einfache, einschichtige KNNs sehr komplexe Aufgaben lösen. KNN, die nun eine bestmögliche Vektordarstellung für ein Retrievalproblem erarbeiten sollen, werden üblicherweise mit einem dünn besetzen Term-Eingabe-Vektor oder vorberechneten Worteinbettungen versorgt und lernen so Schritt für Schritt, welche neue Vektorrepräsentation für Anfragen und Dokumente am besten geeignet ist.

3.2 Deep Learning

Tiefe neuronale Netze (*deep neural networks* – DNN) bestechen durch ihre große Anzahl von Schichten (teilweise mehrere Tausend, s. Trabelsi et al. 2021) und einer entsprechend großen Anzahl an Parametern, die im Trainingsprozess gelernt werden können. Bei kleinen Datensätzen kann *Overfitting* hierbei ein Problem darstellen, weshalb eine Ausgewogenheit zwischen Trainingsdaten und zu trainierender Parameter beachtet werden sollte (Goodfellow et al. 2016).

Im IR bestehen Trainingsdatensätze üblicherweise aus Dokumenten, Anfragen und idealerweise Relevanzurteilen. Während die ersten beiden Entitäten meist in ausreichendem Umfang vorhanden sind, sind große Relevanzdatensätze (oder zumindest indirekte Relevanzsignale, wie Klickdaten) nur im industriellen Umfeld verfügbar. Wie Mitra und Craswell (2018) beschreiben, bestimmt die Menge an verfügbaren Dokumenten und Anfragen den Grad und die Art des Trainingprozesses des DNN. Liegen keine Relevanzurteile vor, werden die ungelabelten Dokumente (bzw. analog die Anfragen) verwendet, um passende Vektordarstellungen zu erlernen. Diese werden dann in bestehende Retrievalmodelle oder Ähnlichkeitsmetriken zum Abgleich von Anfragen und Dokumenten integriert. Sollte eine kleine Menge an gelabelten Daten zur Verfügung stehen, können diese genutzt werden, um ein Retrievalmodell mit wenigen Parametern zu trainieren, das wiederum Vektordarstellungen verwendet, die auf einem größeren, nicht gelabelten Korpus vortrainiert wurden. Einen sehr guten Überblick über aktuelle Deep Learning-Ansätze mit IR-Bezug findet sich bei Goldberg (2017).

3.3 Neuronale Retrieval-Modelle

Es gibt nun verschiedene Möglichkeiten, KNN für das Retrieval bzw. zum Berechnen der Ähnlichkeit von Anfrage und Dokument einzusetzen: repräsentationsorientierte Modelle, interaktionsorientierte Modelle und gemischte Modelle (Fan et al. 2021). Repräsentationsorientierte Modelle (Huang et al. 2013) konzentrieren sich auf das unabhängige Lernen dichter Vektordarstellungen von Anfragen und Dokumenten. Anschließend werden

Metriken wie die Kosinusähnlichkeiten verwendet, um die Ähnlichkeit zwischen Anfragen und Dokumenten zu berechnen. Bei den interaktionsorientierten Modellen (Guo et al. 2016) werden die sog. Interaktionen zwischen Anfragen und Dokumenten erfasst, die in einer Anfrage-Dokument-Matrix festgehalten werden. Jeder Eintrag der Matrix enthält den Ähnlichkeitswert von Worteinbettungen der jeweiligen Suchterme und den Worteinbettungen der Dokumentterme. Nachdem diese Matrix erstellt wurde, werden weitere Merkmale extrahiert, die zur Berechnung der Ähnlichkeit von Suchanfragen und Dokumenten verwendet werden können. Es gibt auch Mischformen beider Ansätze (MacAvaney et al. 2019).

4 Kontextualisierte Sprachmodelle im IR

Während mit Methoden wie Word2Vec oder GloVe bis dato erfolgreich semantische Beziehungen zwischen Wörtern dargestellt und z. B. Synonyme gut gefunden werden konnten, wurde in den folgenden Jahren deutlich, dass dies nicht ausreicht, um das Phänomen Sprache ausreichend abzubilden. Word2Vec lässt sich zwar auf Satz- oder Dokumentebene erweitern, jedoch gestaltet sich hier die Ähnlichkeitsbestimmung als schwierig. Die Unterschiedlichkeit der Sätze und Dokumente ist einfach zu groß. Durch die Möglichkeiten, die Deep Learning und große vortrainierte Sprachmodelle bieten, konnte man sich in den Folgejahren auf das Problem der Kontextualisierung (Smith 2020) fokussieren. Hierbei wird versucht, Wörter durch ihren jeweiligen semantischen und syntaktischen Kontext genauer zu erschließen (s. Abbildung 2). Der Fokus auf die Entwicklung von tiefen, kontextsensitiven Einbettungen führte zu Modellen wie GPT, ELMo oder BERT, die auf großen Mengen an ungelabelten Daten trainiert werden und bei verschiedenen NLP- und IR-Aufgaben sehr hohe Leistungen erzielen.

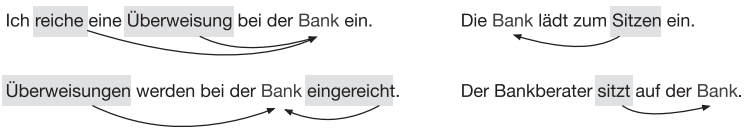

Abb. 2: Unterschiedliche Bedeutungen des Wortes Bank, die sich erst durch den Kontext erschließen

4.1 BERT

Bidirectional Encoder Representations from Transformers (BERT) ist ein Sprachmodell, das sowohl den linken als auch den rechten Kontext eines Wortes nutzt (Devlin et al. 2019). BERT setzt hierbei auf die sog. Transformer-Architektur und den Attention-Mechanismus (Vaswani et al. 2017). BERT wird typischerweise für zwei Aufgaben trainiert: (1) *Masked Language Model* (MLM), bei dem einzelne Wörter im Text ausgeblendet bzw. „maskiert" werden und vom System korrekt rekonstruiert werden sollen, sowie (2) die *Next Sentence Prediction* (NSP), bei der das System Satzpaare als Eingabe erhält und lernt vorherzusagen, ob der zweite Satz im Paar der nachfolgende Satz im Originaldokument ist.

Zum Training des MLM werden in der ursprünglichen Implementierung von BERT[3] 15 % aller Wörter maskiert und die entsprechenden Textsequenzen durch das Encoder-Modul eines Transformers geschickt. So wird die Vorhersage der maskierten Wörter gelernt, und nicht nur das folgende Wort:

Eingabe: Die Studierenden saßen in der [MASKIERUNG-1] und grübelten über die [MASKIERUNG-2] zu BERT.
Label: [MASKIERUNG-1] = Klausur; [MASKIERUNG-2] = Aufgabe

Analog wird für das Training der NSP eine große Menge an Sätzen aus beliebigen Textmengen extrahiert und es werden jeweils Paare A und B als „IsNextSentence“ oder „NotNextSentence“ ausgezeichnet.

Satz A: Die Studierenden saßen in der Klausur.
Satz B: Sie grübelten über die Aufgabe zu BERT.
Label: IsNextSentence

Satz A: Die Studierenden saßen in der Klausur.
Satz B: Ohne Modeste ist der Effzeh einfach nicht der Gleiche.
Label: NotNextSentence

In der Standardimplementierung bestehen Transformermodelle aus zwei separaten Mechanismen, einem Encoder, der die Texteingabe liest, und einen Decoder, der eine Vorhersage für die Aufgabe erstellt. BERT setzt hierbei jedoch nur den Encoder-Mechanismus um. Im Gegensatz zu Modellen, die den eingegebenen Text nur sequentiell von links nach rechts (oder von rechts nach links) lesen, kann ein Transformer mit seinem Encoder einen ganzen Satz (oder eine andere Textsequenz) auf einmal lesen. Dies ermöglicht es den Transformern und somit auch BERT, den Kontext eines Wortes auf der Grundlage seiner gesamten Umgebung und somit den Kontext zu lernen. Im Gegensatz zu ELMo oder GPT wird so eine noch realistischere Sprachverarbeitung ermöglicht, die den Kontext eines Wortes besser einbinden kann.

BERT wird in vortrainierter Form zum Download und zur Nutzung in Standardbibliotheken wie PyTorch oder Tensorflow angeboten. Eine große Stärke liegt in der Möglichkeit, BERT für eigene Datensätze und Aufgaben nachzutrainieren (Devlin et al. 2019, S. 4171, sprechen hier von „fine-tuning“).

4.2 Dokument-(Re-)Ranking mit BERT

Kontextualisierte Sprachmodelle wie BERT lassen sich auf unterschiedlichen Wegen für das Dokumentenretrieval einsetzen. Der häufigste Ansatz ist die sog. MultiStage-Architektur (MSA), die darauf basiert, zunächst mit Hilfe traditioneller Verfahren wie BM25 eine Top-k-Liste von Kandidaten zu ermitteln, die danach von einem BERTReranker weiterverarbeitet werden (Lin et al. 2021). In Abbildung 3 ist dieser Ablauf schematisch dargestellt. Die erste Stufe der MSA liefert neben dem Kandidatendokument auch einen Relevanz-Score pro Dokument, der später weiterverwendet werden kann. Im zweiten

3 https://github.com/google-research/bert.

Schritt wird mit Hilfe von BERT und etwaiger Feinabstimmungen eine erneute Berechnung der Relevanz-Scores durchgeführt und die aus Schritt 1 übernommenen Kandidatendokumente neu gerankt. BERT wird also als sog. Reranker eingesetzt. Bekannt wurde dieser Ansatz durch Nogueira et al. (2019), die ihren Reranker MonoBERT nannten. Die Leistungsfähigkeit in Bezug auf die Precision dieser MSA-Ansätze basiert zu großen Teilen auf den kontextsensitiven Fähigkeiten BERTs, wird in der Praxis aber kaum durch die prinzipiell Recall-kritische termbasierte Kandidatensuche über BM25 in Schritt 1 ausgebremst. Tatsächlich besteht hier aber immer noch das Problem, dass durch BERT natürlich nur Kandidaten neu gerankt werden können, die auch zuvor gefunden wurden. Da hier wieder die ursprünglichen *vocabulary mismatches* ins Spiel kommen, wird der Schritt 1 auch gerne mit den in Abschnitt 2.2 genannten Lösungsansätzen wie z. B. Anfrageerweiterungen angereichert.

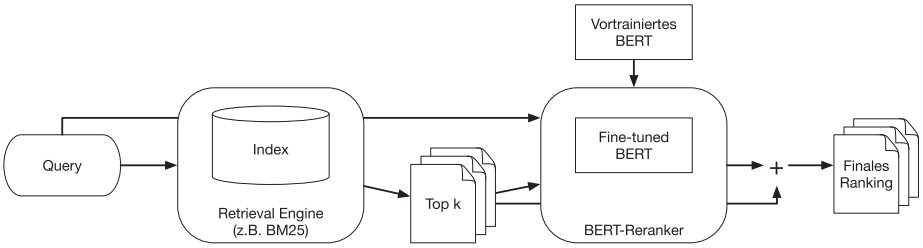

Abb. 3: Multi-Stage-Architektur für das Reranking mit Hilfe von BERT

5 Zusammenfassung

Zusammenfassend lässt sich sagen, dass traditionelle Retrievalmodelle, bei allem Aufwand dies zu kompensieren, auf einer möglichst *exakten Überschneidung* von Anfrage und Dokumenttermen basieren, wohingegen neuronale Retrievalansätze auf eine *semantische Überschneidung* setzen. Neuronale Retrievalsysteme definieren kein festes Regelwerk, sondern nutzen vortrainierte Sprachmodelle, um dichte Vektorrepräsentationen der Texte mit wenigen Dimensionen zu erlernen. Diese Ansätze sind robuster gegenüber Rauschen, lassen sich einfach erweitern und skalieren in der Regel gut. Jedoch sind die Systeme und die Entscheidungswege selbst schlechter nachvollziehbar und die notwendige Menge an Trainingsdaten und die damit verbundene Rechenleistung ist nicht zu unterschätzen (Fan et al. 2021). Mit Hilfe vortrainierter, kontextualisierter Sprachmodelle wie BERT lassen sich leistungsfähige Multi-Stage-Architekturen für das Information Retrieval aufsetzen.

6 Literaturverzeichnis

Armstrong, T. G., Moffat, A., Webber, W. & Zobel, J. (2009). Improvements that don't add up: Ad-hoc retrieval results since 1998. *Proceeding of the 18th ACM conference on information and knowledge management*, 601–610. https://doi.org/10.1145/1645953.1646031.

Cao, G., Nie, J.-Y., Gao, J. & Robertson, S. (2008). Selecting good expansion terms for pseudo-relevance feedback. *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval – SIGIR '08*, 243–250. https://doi.org/10.1145/1390334.1390377.

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

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In J. Burstein, C. Doran & T. Solorio (Eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2–7, 2019, Volume 1 (Long and Short Papers)* (S. 4171–4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/n19-1423.

Fan, Y., Xie, X., Cai, Y., Chen, J., Ma, X., Li, X., Zhang, R., Guo, J. & Liu, Y. (2021). Pre-training Methods in Information Retrieval. *arXiv:2111.13853 [cs]*. http://arxiv.org/abs/2111.13853.

Firth, J. R. (1957). A synopsis of linguistic theory 1930–55. *Studies in Linguistic Analysis (special volume of the Philological Society)*, *1952–59*, 1–32.

Furnas, G. W., Landauer, T. K., Gomez, L. M. & Dumais, S. T. (1987). The Vocabulary Problem in Human-System Communication. *Commun. ACM*, *30*(11), 964–971.

Ganguly, D., Roy, D., Mitra, M. & Jones, G. J. F. (2015). Word Embedding based Generalized Language Model for Information Retrieval. *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 795–798. https://doi.org/10.1145/2766462.2767780.

Goldberg, Y. (2017). *Neural network methods for natural language processing*. Morgan & Claypool Publishers. https://doi.org/10.2200/S00762ED1V01Y201703HLT037.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning*. MIT Press.

Guo, J., Fan, Y., Ai, Q. & Croft, W. B. (2016). A Deep Relevance Matching Model for Ad-hoc Retrieval. *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 55–64. https://doi.org/10.1145/2983323.2983769.

Hienert, D., Schaer, P., Schaible, J. & Mayr, P. (2011). A Novel Combined Term Suggestion Service for Domain-Specific Digital Libraries. In S. Gradmann, F. Borri, C. Meghini & H. Schuldt (Eds.), *TPDL* (Vol. 6966, S. 192–203). Springer.

Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A. & Heck, L. (2013). Learning deep structured semantic models for web search using clickthrough data. *Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management – CIKM '13*, 2333–2338. https://doi.org/10.1145/2505515.2505665.

Jurafsky, D. & Martin, J. H. (2021). *Speech and Language Processing (3rd ed. Draft)*. https://web.stanford.edu/~jurafsky/slp3/ed3book_sep212021.pdf.

Lin, J., Nogueira, R. & Yates, A. (2021). Pretrained Transformers for Text Ranking: BERT and Beyond. *arXiv:2010.06467 [cs]*. http://arxiv.org/abs/2010.06467.

MacAvaney, S., Yates, A., Cohan, A. & Goharian, N. (2019). CEDR: Contextualized Embeddings for Document Ranking. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1101–1104. https://doi.org/10.1145/3331184.3331317.

Metzler, D. (2011). *A Feature-Centric View of Information Retrieval* (Bd. 27). Springer.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In C. J. C. Burges, L. Bottou, Z. Ghahramani & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5–8, 2013, Lake Tahoe, Nevada, United States* (S. 3111–3119). Neural Information Processing Systems Foundation, Inc. (NIPS). https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html.

Miller, G. A. (1995). WordNet: A lexical database for English. *Communications of the ACM*, 38(11), 39–41. https://doi.org/10.1145/219717.219748.

Mitra, B. & Craswell, N. (2018). An Introduction to Neural Information Retrieval. *Foundations and Trends® in Information Retrieval*, 13(1), 1–126. https://doi.org/10.1561/1500000061.

Mitra, B., Diaz, F. & Craswell, N. (2017). Learning to Match using Local and Distributed Representations of Text for Web Search. *Proceedings of the 26th International Conference on World Wide Web*, 1291–1299. https://doi.org/10.1145/3038912.3052579.

Nogueira, R., Yang, W., Cho, K. & Lin, J. (2019). Multi-Stage Document Ranking with BERT. *arXiv:1910.14424 [cs]*. http://arxiv.org/abs/1910.14424.

Pennington, J., Socher, R. & Manning, C. D. (2014). Glove: Global Vectors for Word Representation. In A. Moschitti, B. Pang & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25–29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL* (pp. 1532–1543). ACL. https://doi.org/10.3115/v1/d14-1162.

Schaer, P. (2013). Applied Informetrics for Digital Libraries: An Overview of Foundations, Problems and Current Approaches. *Historical Social Research*, 38(3), 267–281.

Smith, N. A. (2020). Contextual word representations: Putting words into computers. *Communications of the ACM*, 63(6), 66–74. https://doi.org/10.1145/3347145.

Trabelsi, M., Chen, Z., Davison, B. D. & Heflin, J. (2021). Neural ranking models for document retrieval. *Information Retrieval Journal*, 24(6), 400–444. https://doi.org/10.1007/s10791-021-09398-0.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. (2017). Attention is All you Need. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA, USA* (S. 5998–6008). Neural Information Processing Systems Foundation, Inc. (NIPS). https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Yang, W., Lu, K., Yang, P. & Lin, J. (2019). Critically Examining the „Neural Hype“: Weak Baselines and the Additivity of Effectiveness Gains from Neural Ranking Models. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1129–1132. https://doi.org/10.1145/3331184.3331340.

Zhai, C. (2007). Statistical Language Models for Information Retrieval A Critical Review. *Foundations and Trends in Information Retrieval, 2*(3), 137–213. https://doi.org/10.1561/1500000008.