Martin-Luther-Universität Halle-Wittenberg
Institut für Informatik
Studiengang Informatik, M.Sc.

Simulation von Suchanfragen durch Anchortexte

Masterarbeit

Max Steffen Henze
geb. am: 14.12.1997 in Halle (Saale)

Matrikelnummer 216227907

1. Gutachter: Prof. Dr. Matthias Hagen
2. Gutachter: Sebastian Günther

Datum der Abgabe: 30. Mai 2023

Erklärung

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Halle (Saale), 30. Mai 2023

. . . . . .
Max Steffen Henze

Danksagung

Ich danke in erster Linie meinen beiden Mentoren Sebastian Günther und Maik Fröbe für die enge Betreuung dieser Arbeit und ihr hilfreiches Feedback. Des Weiteren möchte ich Isabelle Puchta, Jan Heinrich Reimer und Andreas Zimmermann für das Korrekturlesen dieser Arbeit danken. Ein ganz besonderer Dank gilt meinen Eltern: Nicole und Steffen Henze, ohne deren Unterstützung ich es nicht so weit geschafft hätte. Zu guter Letzt möchte ich Isabelle Puchta für ihre jahrelange Begleitung an meiner Seite sowie Unterstützung in sämtlichen Lebenslagen danken.

Zusammenfassung

An Suchmaschinen gestellte Queries sind der tägliche Einstiegspunkt ins Web für viele Menschen. Im Gegenzug für das Vorschlagen relevanter Web-Dokumente erhalten Suchmaschinenbetreiber Query-Daten zur Verbesserung der unterliegenden Retrieval-Methoden und -Algorithmen. Anbieter kleinerer Suchmaschinen sehen sich dem Problem gegenübergestellt, dass eine geringere Nutzeranzahl keine schnelle und umfängliche Datensammlung ermöglicht. Modelle wie Doc2Query können hierbei durch die Generierung von Queries mit Hilfe von Texten aus Web-Dokumenten Abhilfe schaffen. Somit ergibt sich die Frage, ob weitere frei zugängliche Daten von Web-Dokumenten in diesem Kontext genutzt werden können.

Anchortexte, die anklickbaren Texte von Hyperlinks, lassen sich in großen Mengen sammeln, indem Verlinkungen aus einem Korpus mit Web-Dokumenten extrahiert werden. Diese Arbeit untersucht daher Anchortexte zur Generierung von künstlichen Queries durch die Verwendung von Machine-Learning-Modellen in einer Sequence-to-Sequence-Aufgabe. Die Qualität der generierten Queries wird überprüft durch die Betrachtung von syntaktischen und semantischen Ähnlichkeiten zu echten Queries. Darüber hinaus werden inhaltliche Bezüge zu relevanten Web-Dokumenten in Retrieval-Szenarien untersucht. Der Vergleich mit Doc2Query zeigt, dass die auf dem MS MARCO Datensatz trainierten Machine-Learning-Modelle doppelt so ähnliche Queries erzeugen in Bezug auf Metriken wie ROUGE, BLEU und BERTScore. Zusätzlich bleibt die Retrieval-Effektivität der generierten Queries im Vergleich zu Doc2Query nahezu identisch.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Das Finden von Informationen und Web-Dokumenten im Web gleicht, durch die rasante Zunahme an Web-Dokumenten in den letzten Jahrzehnten [CO98, Odl03, SS11], der Suche nach der Nadel im Heuhaufen. 85% der Web-Nutzer wendet sich daher an kommerzielle Suchmaschinen, um sich bei der Informationsbeschaffung unterstützen zu lassen [KT00]. Web-Suchmaschinen verfolgen das Ziel, für Queries, von Nutzern gestellte Suchanfragen, die relevantesten Web-Dokumente zu finden. Um innerhalb dieses Anwendungsszenarios immer akkuratere Vorschläge aus der Vielzahl von Web-Dokumenten zu erzeugen, stellen Queries einen wichtigen Faktor für Verbesserungen von Methoden und Algorithmen dar. Beispielsweise existiert zwischen Web-Dokumenten und Queries ein unterschiedliches Vokabular, welches ein Hindernis für Retrieval-Techniken die auf Wortüberdeckungen basieren darstellt [EM03, FGP+22]. Dieses sogenannte *Vocabulary-Mismatch-Problem* [FLGD87] kann durch den Einsatz von Queries mittels Document-Expansion oder Query-Reformulation reduziert werden [CWNM02, HTO+16, SMAS17, NYLC19]. Darüber hinaus werden Queries als Trainingsdaten im Machine-Learning für beispielsweise Dense-Retrieval [KOM+20] oder Learning-to-Rank [NRS+16, NYCL19, NC19, HWBN20, NJL20] verwendet. Die Sammlung einer Vielzahl von Queries für solche Zwecke gelingt Suchmaschinen wie Google[1], Bing[2] oder Baidu[3] angesichts einer riesigen Anwenderzahl [TZ11, Lew23]. Durch Tracking des Nutzerverhaltens und der Speicherung in sogenannten Session-Logs können große Web-Suchmaschinen große Datenmengen erstellen, auf denen sie ihre unterliegenden Algorithmen und Retrieval-Modelle verbessern können [BHM04, CB08]. Die Annahme, dass kleineren Suchmaschinen solche Möglichkeiten verwehrt bleiben liegt somit nahe, da ein kleinerer Nutzerstamm vermutlich weniger

[1] `https://google.com/`
[2] `https://bing.com/`
[3] `https://baidu.com/`

leicht für ein äquivalentes Datenaufgebot sorgen kann. Zusätzlich stellt die Verwendung von fremden Query-Datensätzen keine valide Alternative dar. Query-Logs kommerzieller Suchmaschinen sind aus kommerziellen Gründen oftmals unzugänglich und frei zugängliche Datensätze sind, aufgrund der Voreinnahme von Betrachtungsweisen während der Erstellung, oftmals in ihrem Umfang oder Verwendungszweck eingeschränkt [PRB+20, GM22]. Allgemein stellt die Verwendung von Daten einer anderen Domäne eine Problematik in sich dar. Zhang et al. [ZYL20] konnten zeigen, dass die Nutzung weniger domänenspezifischer Daten für das Fine-Tuning von Retrieval-Modellen schlechter sein kann als die Verwendung eines Modells ohne Fine-Tuning.

Um diesen Limitationen entgegenzuwirken, stellt die Generierung von künstlichen Queries somit eine attraktive Variante zum Ausgleich fehlender Query-Daten für die Verbesserung von Retrieval-Mechanismen dar. Verfahren wie *Doc2Query* [NYLC19, FNLE19] sind diesbezüglich bereits in der Lage, Abhilfe zu schaffen. Gegeben eines Web-Dokumententextes kann Doc2Query passende Queries generieren, welche in hoher Relevanz zum Ausgangsdokument stehen. Dies führt zu der Frage, ob die Betrachtung anderer Ausgangsdaten ebenfalls zur Generierung von qualitativ hochwertigen Queries führen kann. Allen voran bieten Anchortexte dabei einen wesentlichen Mehrwert durch ihre Verfügbarkeit und Informationsdichte [BP98, EM03, FGP+22]. Bei der Verlinkung eines Web-Dokumentes zu einem anderen formuliert ein Autor einen kurzen Textabschnitt, auf dem er dann einen Verweis für die Zielseite hinterlegen kann. Dieser Textabschnitt sollte möglichst gut zur Zielseite passen, um dem Leser einen Kurzeindruck zu verschaffen, was ihn nach Anklicken des Verweises erwartet. Anchortexte enthalten somit viele Informationen bei durchschnittlich kurzer Länge und werden darüber hinaus durch einen ähnlichen Gedankenprozess wie Queries erstellt [EM03]. Die Anreicherung des Datensatzes MS MARCO [NRS+16] mit einer Vielzahl von Anchortexten [FGP+22], bildet die Grundlage zur Untersuchung dieser Überlegungen in dieser Arbeit. Verschiedene Sequence-to-Sequence Transformer-Modelle werden mit unterschiedlichen Varianten des MS MARCO Datensatzes auf bis zu 1.000.000 Query-Dokument-Paaren trainiert. Die syntaktische und semantische Ähnlichkeit von generierten und echten Queries wird durch die Metriken ROUGE [Lin04], BLEU [PRWZ02, Pos18] und BERTScore[ZKW+19] überprüft. Um zusätzlich einen inhaltlichen Bezug von Query und Ausgangsdokument sicherzustellen, wird die Effektivität der generierten Queries in Retrieval-Szenarien unterschiedlicher TREC-Tracks [CCS09, CMY+20] ermittelt.

Die auf Dokumenten- und Anchortexten trainierten Modelle erzeugen Queries, denen durch ROUGE, BLEU und BERTScore eine doppelt so hohe Ähn-

lichkeit zu echten Queries aus MS MARCO zugeordnet wird, wie mit Doc2Query generierte Queries. Das effektivste, auf Grundlage von T5-`base` [RSR+20] trainierte, Modell erzielt dabei einen ROUGE-L F1-Score von 0,49 und einen BLEU-Score von 0,21. Die Reduktion der Trainingsdaten auf 1% der Query-Dokument-Paare führt zu minimalen Verschlechterungen des BLEU-Scores auf 0,17. Die Modelle sind in der Lage, die Retrieval-Effektivität bezüglich der untersuchten TREC-Tracks auf MS MARCO im Vergleich zu Doc2Query zu halten und zeigen sogar eine eindeutige Verbesserung auf dem ClueWeb09[4]-Datensatz.

Die anschließenden Kapitel dieser Arbeit strukturieren sich wie folgt: Kapitel 2 geht näher auf verwandte Arbeiten ein, deren Betrachtungen als Grundlage dieser Arbeit dienen. Daraufhin folgen in Kapitel 3 Überlegungen für die Umsetzung der betrachteten Aspekte dieser Arbeit. Kapitel 4 umfasst die nähere Beschreibung der Experimente sowie die Auswertung der erzielten Ergebnisse. Abschließend wird in Kapitel 5 ein Fazit zu den Ergebnissen gezogen und auf mögliche zukünftige Betrachtungen eingegangen.

[4] `https://lemurproject.org/clueweb09/`

Kapitel 2

Grundlagen und Verwandte Arbeiten

Diese Arbeit befasst sich mit der Untersuchung von Anchortexten zur Generierung von künstlichen Queries mit Hilfe von Transformer-Modellen. In den nachfolgenden Paragraphen wird näher auf verwandte Arbeiten der Abschnitte: (1) Anchortexte, (2) Transformer-Modelle, und (3) Generierung von Pseudo-Test-Collections eingegangen.

Anchortexte Innerhalb des Webs werden Web-Dokumente durch Hyperlinks („Links“) miteinander verbunden. Um aber den Lesefluss eines Besuchers nicht zu unterbrechen, werden solche Links durch sogenannte Anchortexte maskiert. Zum Beispiel signalisiert der Hyperlink:

```
<a href="https://www.uni-halle.de/">
  Webseite der Martin-Luther-Universität Halle-Wittenberg
</a>
```

Nutzern, dass sie durch Anklicken auf das Web-Dokument der Martin-Luther-Universität Halle-Wittenberg geführt werden. Die Abbildung 2.1 zeigt eine beispielhafte Einbindung von Anchortexten innerhalb eines Web-Dokumentes als Navigationselement. Die Einbindung innerhalb eines Fließtextes ist ebenfalls möglich. Allgmein werden Anchortexte durch das HTML-Tag: `<a>` gekennzeichnet und führen über das `href`-Attribut zur verlinkten Seite. Anchortexte sind ein wichtiges Element innerhalb der Web-Gestaltung. Sie dienen als prägnante und beschreibende Zusammenfassung der hinterlegten Seite [EM03]. Dies gibt ihnen, wegen ihrer oftmals geringen Länge, eine sehr hohe Informationsdichte. Brin und Page [BP98] argumentieren sogar, dass Anchortexte ein Web-Dokument besser beschreiben als der Dokumententext selbst. Im Gegensatz

Abbildung 2.1: Einbindung von Anchortexten (Studium, Forschung etc.) als Navigationselement des Web-Dokuments der Martin-Luther-Universität Halle-Wittenberg.[1]

zu Titeln, welche meist von einem Autor verfasst werden, bieten Anchortexte verschiedener Web-Dokumente den Vorteil, dass sie von unterschiedlichen Autoren stammen. Somit ergeben sie in Summe eine diversere Zusammenfassung des Dokumenteninhalts, als ein zugehöriger Titel [EM03]. Diese Eigenschaften machen Anchortexte unter anderem sehr wertvoll für die Verbesserung von Retrieval-Ansätzen.

Im Rahmen der *Text REtrieval Conference* (TREC) wurde bereits seit 1999 an der Verwendung von Anchortexten geforscht. Dabei konnten aber im TREC 8 und TREC 9 Web Track keine Verbesserungen für Ad-Hoc-Retrieval durch die Hinzunahme von Link-Informationen erzielt werden [HVCB99, Haw00, KW00]. Ad-Hoc-Retrieval ist eine Retrieval-Aufgabe, bei der, gegeben einer Query, ein Informationssystem eine Menge von relevanten Dokumenten aus einem gegebenen Korpus zurückgeben soll. Die erzielten Ergebnisse waren ein Widerspruch zu den Erfahrungen kommerzieller Suchmaschinen, da vor allem diese eine erfolgreiche Verwendung von Link-Informationen angaben [BP98]. Ad-Hoc-Retrieval galt bis dato als realistischste Art und Weise, wie Nutzer ihren Informationsbedürfnissen im Web nachkommen. Hawking und Craswell [HC05] realisierten aber, dass die Diskrepanz der Ergebnisse ihre Ursache im Suchprozess von Internetnutzern hatte. Sie schlussfolgerten, dass dieser Suchprozess verschieden zum bisher untersuchten Ad-Hoc-Retrieval ist. Sie begründeten dies damit, dass Nutzer die Hauptseite eines bekannten, thematisch passenden Web-Dokuments gegenüber einem alleinstehenden Text bevorzugen, egal wie relevant der Text ist [HC05]. Sie zeigten daher, dass Anchortexte für eine, zu dieser Aussage passende, Entry-Page- beziehungsweise Homepage-Finding-Aufgabe doppelt so effektiv sind, wie rein inhaltsbasierte Ansätze [CHR01]. Beide Aufgaben zeichnen sich durch ihre Untersuchungen von navigierenden Query-Eigenschaften aus. Die folgenden TREC Web Tracks fokussierten sich somit auf navigationsbasierte Aufgaben wie Site-

[1] `https://uni-halle.de`

Finding und konnten ebenfalls Verbesserungen durch Anchortexte nachweisen [WKH01, HC02, CH02]. Gurrin und Smeaton [GS04] zeigten, dass die im TREC 8 und TREC 9 Web Track verwendeten Datensätze WT2g und WT10g [BCH03] über keine genügende Link-Dichte verfügten. Eine realistische Betrachtung sei somit nicht möglich gewesen. Um dies zu verifizieren separierten sie Links kategorisch in interne und externe Links. Interne Links sind Verlinkungen innerhalb einer Domäne:

`https://uni-halle.de/` → `https://uni-halle.de/studium/`

und externe Links meinen die Verlinkungen zwischen zwei Domänen:

`https://uni-halle.de/` → `https://trec.nist.gov/`

Sie schlussfolgerten, dass für eine realitätsnahe Linkdichte jedes Dokument fünf eingehende externe Links besitzen sollte und ein Datensatz ausreichend groß sein muss. Erst nach der Erfüllung dieser Anforderungen durch den ClueWeb09[2]-Datensatz konnten Koolen und Kamps [KK10] im Rahmen des TREC 2009 Web Track [CCS09] zeigen, dass für Ad-Hoc-Retrieval Anchortexte, in Bezug auf Early-Precision, performanter sind als Volltext-Retrieval. Eine Kombination beider Elemente führte zu einer allgemeinen Verbesserung der Precision [KK10]. Seitdem unterliegen Anchortexte verschiedenen Betrachtungen. Dai und Davison [DD10] untersuchten die Erzeugungsrate von Anchortexten als Maß der Wichtigkeit von Web-Dokumenten. Sie wiesen durch Nutzung dieser Informationen eine Verbesserung der Ranking-Effektivität auf dem Stanford-WebBase Web-Crawl [CGH+06] nach. Yi und Allan [YA10] verwendeten Anchortexte für die Erzeugung eines Sprachmodells, welches fehlende Anchortexte aus Dokumententexten generieren kann. Der Einsatz im TREC Terabyte Track [BCS06] zeigte Verbesserungen gegenüber Methoden, die nur Anchortexte des Web-Graphen verwenden. Ma et al. [MDX+21] untersuchten Anchortexte für das Pre-Training von Sprachmodellen im Kontext des Ad-Hoc-Retrieval. Sie stellten durch die Informationsdichte der Anchortexte eine Verbesserung der trainierten Modelle im Vergleich zu Referenzmodellen wie BERT [DCLT18] fest. Durch die Verwendung von Anchortexten zum Training eines Dense-Retrieval-Modells zeigten Xie et al. [XLX23] die Überlegenheit zu einem konkurrierenden Unsupervised-Dense-Retrieval-Modell.

Die fehlende Betrachtung von Anchortexten als Grundlage zum Training eines Sequence-to-Sequence-Sprachmodells dient unter anderem als Motivation dieser Arbeit. Durch die Umwandlung hin zu Queries können Retrieval-Mechanismen sowohl direkt, durch Methoden wie Document-Expansion, als

[2] `http://lemurproject.org/clueweb09/`

auch indirekt, mit der Verwendung generierter Queries als Testdatensätze, verbessert werden.

Transformer Basierend auf der von Rosenblatt entworfenen Perceptron-Architektur [Ros58] entwickelte sich mit *Recurrent Neural Networks* (RNNs) [RHW86] 1986 ein Modell mit hoher Effektivität für Sequence-to-Sequence-Aufgaben [GFGS06, SVL14, VRD+15, SSRB17, PRS+17]. RNNs erlauben den umliegenden Kontext von Termen zeitweise zu speichern und zu einem späteren Zeitpunkt wiederzuverwenden. Somit konnten komplexere Satzstrukturen erkannt und von Sprachmodellen erlernt werden. Doch lösen RNNs nicht die Problematik des Vergessens bei langen Eingaben. Früh auftretende Merkmale bleiben nicht lange genug im Speicher des RNNs. Die Weiterentwicklung von RNNs mittels *Long-Short-Term-Memory*-Zellen (LSTMs) [HS97, SVL14] begann dieses Problem zu lösen. Über verschiedene Gatter sind LSTMs in der Lage, wichtige Informationen lange zu speichern. Ebenfalls können sie unwichtige Bestandteile einer Eingabe identifizieren und diese nach dem Einlesen gezielt wieder vergessen.

Die Weiterentwicklung von LSTMs führte zu den *Gated Recurrent Units* (GRUs) [CvMG+14, CvMBB14, CGCB14, BCB15, YKYS17]. Hierbei handelt es sich um eine Vereinfachung der LSTMs, die mit weniger Parametern beim Training auskommen. Obwohl LSTMs und GRUs in der Lage sind, längere Eingabesequenzen als RNNs zu verarbeiten, scheitern dennoch alle drei Modellarchitekturen daran, ab einer gewissen Länge kontextuelle Bezüge aufrechtzuerhalten. Dies geschieht insbesondere durch ihre sequentielle Verarbeitungsweise, bei der die Hinzunahme neuer Eingabebestandteile zum Vergessen vergangener Aspekte führt. Durch die eigenständige Fokussierung von Modellen auf wichtige Bestandteile der Eingabe, dem sogenannten *Attention-Mechanismus* [BCB15], konnte diese Problematik überwunden werden. Zum Einsatz kommt der Attention-Mechanismus aktuell als Bestandteil von Transformer-Modellen [VSP+17] für die Umsetzung vieler effektiver Modelle in den unterschiedlichsten Einsatzgebieten [DCLT18, LOG+19, RWC+19, LLG+20, RSR+20, BMR+20]. Die Zweiteilung in *Pre-Training* und *Fine-Tuning* [MBXS17, HR18, PNI+18, DCLT18, RNS+18] erlaubt es Modelle auf generischen Korpora vorzutrainieren und diese dann später für spezifische Aufgaben anzupassen, um eine starke Leistung zu erzielen [WDS+20].

Im Zuge dieser Arbeit werden die Transformer-Modelle BERT [DCLT18], BART [LLG+20] und T5 [RSR+20] für Sequence-to-Sequence-Aufgaben wie Translation oder Summarization eingesetzt, um künstliche Queries aus Anchortexten zu generieren.

Generierung von Pseudo-Test-Collections Wie durch Wang et al. [WTRG21] untersucht, brauchen bestimmte Ansätze wie beispielsweise Dense-Retrieval eine große Menge von Trainingsdaten. Zusätzlich kann ein Ausweichen auf alternative Datensätze, welche nicht aus derselben Domäne stammen, zu Leistungseinbußen führen [TRR+21]. Selbiges gaben Zhang et al. [ZYL20] an und zeigten, dass Zero-Shot-Ranking mit Transformer-Modellen ohne Fine-Tuning effektiver ist als mit Fine-Tuning auf nur wenigen domänenspezifischen Daten. Die Problematik mangelnder Daten ist insbesondere im Kontext von Query-Logs vorzufinden. Datensätze von Suchmaschinen sind oftmals proprietär oder in ihrer Größe eingeschränkt. Öffentliche Query-Logs wie MS MARCO [NRS+16] werden nur in großen Zeitabständen und unter großem Aufwand veröffentlicht und verzögern somit Ergebnisse in der Forschung. Doch die Notwendigkeit von großen Query-Datensätzen ist in vielen Aufgabenbereichen des Information Retrieval gegeben. Diverse Forschungen suchen daher nach möglichen Strukturen, welche als Pseudo-Queries verwendet werden können. Broder [Bro02] definierte bereits 2002 eine Unterscheidung von Queries nach Nutzerbedürfnis in *Informational*, *Transactional* und *Navigational*. Darauf aufbauend benutzten beispielsweise Hawking et al. [HCCU04] Site-Maps als Ersatz für Navigational-Queries. Bei Site-Maps handelt es sich um ein Verzeichnis mit internen Links eines Web-Dokumentes. Asadi et al. [AMEL11] verwendeten gefilterte Anchortexte aus dem Web-Graphen des ClueWeb09-Datensatzes und übertrafen BM25, mit einem darauf trainierten Learning-To-Rank-Modell. In zwei Arbeiten von Berendsen et al. [BTdRM12, BTWdR13] wurden Annotationen von wissenschaftlichen Arbeiten beziehungsweise Tweets und Hashtags als Query-Ersatz verwendet. Beide Pseudo-Datensätze haben für das Training von Learning-to-Rank-Modellen eine ähnliche, wenn auch leicht schlechtere, Effektivität wie konventionelle Datensätzen. Als eines der bekanntesten Modelle ist Doc2Query von Noguiera et al. [NYLC19, FNLE19] in der Lage, durch ein T5-Modell, aus Texten von Web-Dokumenten wohlgeformte Queries zu erzeugen. Das Anhängen dieser Queries an den ursprünglichen Dokumententext sorgt für eine Reduzierung des Vocabulary-Mismatch-Problems [FLGD87] und erhöht die Retrieval-Effektivität in Hinblick auf MAP, MRR@10 und Recall@1000 gegenüber BM25. Die darauf aufbauende Erweiterung Doc2Query−−, von Gospodinov et al. [GMM23], identifiziert weniger relevante Queries, die durch Halluzination von Doc2Query erzeugt wurden. Die Retrieval-Effektivität nimmt nach Entfernung von weniger relevanten Queries zu.

Eine weitere Form der Generierung von Pseudo-Queries ist die Umformulierung existierender Queries. Ziel ist dabei oftmals die Reduktion des zuvor erwähnten Vocabulary-Mismatch zwischen Query und Dokumenttexten. Kraft und Zien [KZ04] nutzten Anchortexte als Basis für Umformulierung mit In-

tention der Annäherung an Queries. Dang und Croft [DC10] untersuchten die Verwendung eines Anchortext-Logs zur Anreicherung von Queries durch die Substitution einzelner Terme oder Erweiterung bestehender Queries. Sie fanden dabei, dass Substitution insbesondere bei längeren Queries und Erweiterung bei kürzeren Queries effektiv ist. Sowohl Kraft und Zien als auch Dang und Croft stützen sich auf die Ergebnisse von Eiron und McCurley [EM03], dass Anchortexte und Queries eine starke Ähnlichkeit aufweisen. Diese Eigenschaft konnte von Fröbe et al. [FGP+22] für moderne Web-ähnliche Datensätze wie MS MARCO nicht nachgewiesen werden. Craswell et al. [CBFN13] untersuchten ebenfalls die Verwendung von Anchortexten für Query-Reformulation, aber auf den Web-Datensätzen Robust04 [3], GOV2[4] und ClueWeb09. Sie fanden dabei eine Verbesserung der Early-Precision und Antwortzeit innerhalb der Retrieval-Experimente.

Die Generierung von Pseudo-Queries allein ist nicht ausreichend für eine umfängliche Test-Collection. Relevanzbewertungen müssen erstellt werden, um inhaltliche Beziehungen zwischen Pseudo-Queries und Web-Dokumenten erkennbar zu machen [STS11]. Die Erstellung dieser Relevanz-Bewertungen erweist sich als zeit- und kostenintensiv [MRR13]. Oftmals braucht es Experten zur Bewertung von thematischen Zuordnungen. In Szenarien, in denen Pseudo-Queries nicht durch Umformulierung oder ähnliche Techniken generiert werden und schon bestehende Relevanzbewertungen verwendet werden können, kommen andere Methoden zum Einsatz, die beispielsweise anhand von Session-Logs solche Relevanzbewertungen ableiten. [Joa02, JGP+05, BCYS07, CJ07].

[3] `https://trec.nist.gov/data/robust.html`

[4] `http://ir.dcs.gla.ac.uk/test_collections/gov2-summary.htm`

Kapitel 3

Simulation von Queries durch Anchortexte

In den nachfolgenden Abschnitten wird näher darauf eingegangen, warum Transformer-Modelle und Anchortexte eine geeignete Basis zur Generierung von künstlichen Queries darstellen. Ebenfalls wird analysiert, welche Methoden der Umformung im Sinne einer Sequence-to-Sequence-Aufgabe geeignet sind.

3.1 Modellgrundlage

Die Umwandlung von Anchortexten zu Queries mittels Machine-Learning fällt in das Thema der sogenannten *Sequence-to-Sequence-Generierung*. Durch das Erlernen von Textstrukturen, die Fokussierung auf kontextuelle Bezüge sowie die Zunahme an Parameterzahl und Komplexität entstanden in den letzten Jahren Modelle mit immer besserer Effektivität innerhalb des Natural Language Processing (vgl. Abschnitt 2). Die Anzahl an Parametern und verwendeten Daten dieser Sprachmodelle wächst seitdem exponentiell (vgl. Abbildung 3.1). Machine-Learning-Modelle, mit einem Fokus auf Sprache und Generierung von Texten nach einer erlernten Wahrscheinlichkeitsverteilung, werden als *Large-Language-Models* (LLMs) bezeichnet. Sie werden in vielen Aufgabenbereichen sehr erfolgreich eingesetzt. Einer dieser Bereiche ist Sequence-to-Sequence, bei dem Eingabe- in Ausgabetexte umgewandelt werden müssen. Hierbei sind es Transformer-Modelle [VSP+17], die durch ihre Effektivität auf vielen Teilgebieten [DCLT18, RSR+20, LLG+20, WDS+20] überzeugen und deshalb auch Anwendung in dieser Arbeit finden.

Ein besonderes Merkmal von Transformer-Modellen ist der sogenannte *Self-Attention-Mechanismus* [VSP+17]. Durch die Bezugnahme jedes Wortes zu

jedem anderen Wort eines Eingabetextes kann Kontext auch über längere Sequenzen erhalten und wichtige Aspekte herausgestellt werden. Dadurch kann beispielsweise in den Eingaben: „Mein Schuh ist kaputt. [...] Der Absatz hat mir sowieso nicht gefallen." und „Meine Textdatei ist beschädigt. [...] Der Absatz hat mir sowieso nicht gefallen." dem Wort „Absatz" eine unterschiedliche Bedeutung durch gegebenen Kontext zugewiesen werden. Ebenso wird durch Self-Attention deutlich, dass „Absatz" und „Schuh" beziehungsweise „Textdatei" in Bezug zueinander stehen und durch das Auftauchen am Anfang und Ende des Textes scheinbar wichtige Merkmale sind.

Ein weiterer wichtiger Vorteil ist die Parallelverarbeitung von Transformer-Modellen. Frühere Modelle wie RNNs [RHW86] oder LSTMs [HS97] (vgl. Abschnitt 2) basieren auf einer sequentiellen Verarbeitung, bei der eine Eingabe von links nach rechts eingelesen wird. Durch Self-Attention in mehreren Layern sind Transformer-Modelle in der Lage, die gesamte Eingabe auf einmal zu verarbeiten. Vaswani et al. [VSP+17] verdeutlichen dies durch die Komplexität der nötigen sequentiellen Operationen, welche bei Transformern konstante ($\mathcal{O}(1)$) und bei RNNs/LSTMs lineare Laufzeitkomplexität ($\mathcal{O}(n)$ bei Eingabelänge n) benötigen.

Allgemein hat sich bei der Verwendung von LLMs und somit auch Transformer-Modellen eine etablierte Zweiteilung ergeben. Durch sogenanntes *Pre-Training* können große Modelle auf ungelabelten Daten vortrainiert werden. Dabei erlernen sie beispielweise Syntax und Semantik, welche, unabhängig eines spezifischen Einsatzziels, wichtig für den Umgang mit Sprache sind. Für die Verwendung auf sogenannten *Downstream-Tasks* werden dann mittels *Fine-Tuning* die Modelle auf ihre Endaufgabe abgestimmt. Dabei kann sich das Ziel dieser Aufgabe sehr stark unterscheiden. Beispielsweise versucht *Question-Answering* Antworten auf gestellte Fragen zu finden. Das Teilgebiet der *Translation* hingegen versucht, Texte einer Ausgangs- hin zu einer Zielsprache zu übersetzen und dabei möglichst syntaktische und semantische Merkmale zu erhalten.

3.2 Datengrundlage

Um die Generierung von realistischen Queries durch Transformer-Modelle zu gewährleisten, muss die verwendete Datengrundlage Queries mit Charakteristiken enthalten, die denen von natürlich vorkommenden Queries entsprechen. Die Replikation der Merkmale solcher Queries, die von Nutzern an eine Such-

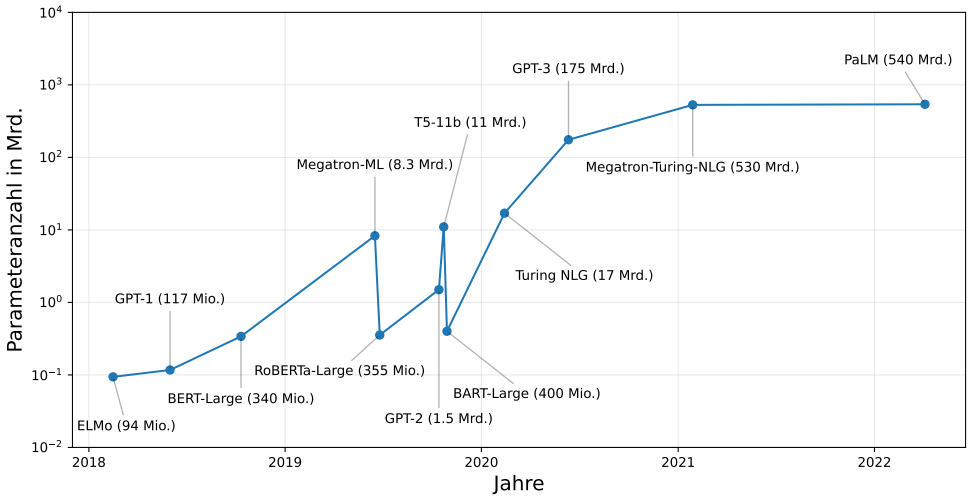

Abbildung 3.1: Überblick über die Parameterentwicklung von Large-Language-Models (LLMs) in den letzten Jahren.

maschine gestellt werden, gilt in dieser Arbeit als höchstes Ziel. Verwendete LLMs sollen in der Lage sein, diese charakteristischen Merkmale im Fine-Tuning zu erlernen und für die Generierung auf unbekannten Daten wiederzuverwenden. Allgemein lassen sich die Daten in drei Unterkategorien einordnen: (1) Queries, (2) Anchortexte, und (3) Web-Dokumente.

Queries Im Kontext des Machine-Learnings und spezieller der Sequence-to-Sequence-Generierung können Queries als Label angesehen werden und dienen LLMs als syntaktisches und semantisches Ziel, nach dem diese ihre Parameter anpassen sollen.
Bereits erschienene Veröffentlichungen konnten markante Merkmale bei Queries feststellen. Sie sind demnach oftmals, aus wenigen Wörtern bestehende, Nominalphrasen [EM03, FGP+22]. Bei einer Query-Log-Analyse von Yahoo-Queries[1] fanden Barr et al. [BJR08], dass ein Großteil (83%) von Queries ausschließlich Kleinschreibung verwendet und sich Queries fundamental von natürlicher Sprache, durch das Weglassen von Verben und dem vermehrten Gebrauch von Nomen, unterscheiden. Broder [Bro02] und Jansen et al. [JBS08] zeigten, dass sich Queries in ihrer inhaltlichen Orientierung in drei Kategorien einteilen lassen: (1) *Informational*, (2) *Navigational*, und (3) *Transactional*. Diese lassen sich unter anderem durch unterschiedliche Formulierungen identifizieren. Informational-Queries versuchen, einen Wissensdrang zu befriedigen und sind eher wohlgeformt und enthalten fragende Phrasen wie: `wie kann`

[1] `https://yahoo.com`

`ich...` oder `was ist...`. Navigational-Queries enthalten Firmen- oder Webseitennamen und dienen zum Auffinden von Domänen im Internet, beispielweise: `facebook login`. Transactional-Queries können zusammengefasst werden unter der Suche nach beschaffbaren Elementen wie beispielsweise `firefox download`, aber auch nach dem Drang der kommerziellen Nutzung wie etwa: `blaue jeans zalando`. Allgemein ergab die Untersuchung des Query-Logs der Dogpile[2] Suchmaschine, dass 81% aller Queries Informational, 10% Navigational und 9% Transactional sind.

In Hinblick auf Retrieval leiden Queries unter dem *Vocabulary-Mismatch-Problem* [FLGD87, NYLC19]. Nutzer verwenden dabei Wörter zur Bildung ihrer Queries, welche nicht zum relevantesten Web-Dokument führen, da diese beispielsweise Synonyme oder doppeldeutige Wörter enthält. Ein Beispiel dafür sind viele Begriffe innerhalb der Informationstechnologie. So meint „Python“ gleichzeitig eine Programmiersprache und eine Schlangenart oder „Transformer“ eine Machine-Learning-Architektur und eine populäre Filmreihe. Relevante Dokumente können in einem Single-Shot-Szenario nicht akkurat zu einer Query, die nur aus solch einem mehrdeutigen Begriff besteht, zugeordnet werden.

Anchortexte Die Verwendung von Anchortexten als primäres Element zur Erzeugung von Queries wird durch die folgenden vier Überlegungen motiviert: (1) Anchortexte stammen aus verschiedenen Quellen, (2) sie besitzen einen diversen Wortschatz, um einen Sachverhalt unterschiedlich auszudrücken, (3) sie agieren als Relevanzbewertung für Web-Dokumente, und (4) sie werden bewusst von Menschen mit der Intention einer prägnanten Beschreibung eines Web-Dokuments gewählt.
Allgemein lässt sich die Erstellung von diversen Anchortexten durch unterschiedliche Autoren erklären, welche sich an einem abweichenden Vokabular bedienen. Hinzu kommt die Tatsache, dass bestimmte Themen sich aus diversen Richtungen betrachten lassen, welche die Verwendung verschiedener Wörter benötigen. Zum Beispiel enthalten die Anchortexte für einen Baumarkt: `weiße Farbe 50L` und `Pflanzendünger mit extra Nährstoffen`, solche Betrachtungsunterschiede. Ersterer verbindet Farbe und somit Malerarbeiten mit einem Baumarkt und letzterer Pflanzen und deren Zucht. Eiron und McCurley [EM03] propagieren diesbezüglich die Idee eines sogenannten *Concept-Space*. Dabei handelt es sich, im Kontext eines spezifischen Themas, um eine Menge an Wörtern, die mit diesem in Verbindung gebracht werden. Neben Unterschieden in der Formulierung bedienen sich Ersteller von Anchortexten

[2] `https://dogpile.com/`

oder Queries an Wörtern aus diesem Concept-Space. Das Zusammentragen von Anchortexten aus verschiedenen Quellen kann demnach als Annäherung an diesen Concept-Space betrachtet werden. Es ergibt sich der Vorteil, dass Anchortexte durch verschiedene Autoren in ihrer Summe eine potentiell höhere Informationsdichte enthalten können als ein einzelner Text eines einzelnen Autors.

Da beliebte Web-Dokumente im Internet als Anlaufstelle vieler Nutzer dienen, aquirieren diese ebenfalls eine Vielzahl an Anchortexten, die auf diese Web-Dokumente verweisen. Somit können Anchortexte ebenfalls als Bewertungsmaßstab, ähnlich zum PageRank [PBMW99], dienen. Die Vermutung liegt nahe, dass Autoren bemüht sind ihre Anchortexte so zu wählen, dass beim Leser ein ausreichender Eindruck der verlinkten Seite vermittelt wird. Je relevanter eine Seite zum thematisierten Inhalt ist, desto beschreibender sollte demnach der gewählte Anchortext zu dieser Seite sein. In der Praxis fällt aber auf, dass oftmals viele generische Anchortexte wie beispielsweise `Click Here` verwendet werden. Verfügbare Datensätze wenden daher teilweise Vorfilterungsschritte an, um solche, als Stopwörter klassifizierten Anchortexte, zu entfernen [FGP+22].

Ein weiterer zu betrachtender Aspekt ist die Ähnlichkeit von Anchortexten zu Queries. In der zuvor erwähnten Studie analysierten Eiron und McCurley [EM03] einen IBM-Intranet-Datensatz und konnten feststellen, dass Anchortexte Ähnlichkeiten zu Queries in Bezug auf Länge und Wortverteilung aufwiesen. Eine Folgestudie von Fröbe et al. [FGP+22] zeigt aber, dass diese Charakteristiken in heutigen Web-artigen Datensätzen wie MS MARCO [NRS+16] nur noch abgeschwächt vorzufinden sind. Dies lässt sich unter anderem dadurch erklären, dass der IBM-Datensatz einem Intranet entspringt. Die Notwendigkeit aussagekräftiger und wohlgeformter Anchortexte ist durch den ähnlichen Kenntnisstand der Mitarbeiter hier nicht gegeben. Zielseiten hinterlegter Anchortexte, wie beispielweise `Homepage` oder `Mail`, sind dem Großteil bekannt. Der disjunktive Kenntnisstand von verschiedenen Nutzern des Webs hingegen erfordert Anchortexte mit einer höheren Informationsdichte.

Web-Dokumente Neben Anchortexten werden zusätzlich die Informationen von Web-Dokumenten als Eingabe verwendet. Diese umfassen den Dokumenteninhalt und -titel. Modelle wie Doc2Query [NYLC19, FNLE19] verwenden bereits Dokumenteninhalte, um künstliche Queries zu erzeugen. Dies lässt darauf schließen, dass LLMs in der Lage sind, selbst aus langen Eingaben wichtige Aspekte zu filtern und in Query-artige Strukturen umzuwandeln. Durch die Kombination von Anchortexten soll somit eine größere Wortmenge gebil-

det werden, auf denen LLMs mittels Self-Attention wichtige Bezüge herstellen sollen.
Bezüglich Dokumententiteln konnten Jin et al. [JHZ02] zeigen, dass sie eine starke Ähnlichkeit zu Queries aufweisen und dass diese einem ähnlichen Gedankenprozess bei der Erstellung entspringen. Anchortexte weisen geringfügig stärkere Ähnlichkeiten zu Queries als zu Dokumententiteln auf. [FGP+22]. Dadurch dass Titel in der Regel wohlgeformte Textstücke sind und zusammenfassende Charakteristiken aufweisen, bilden sie eine wertvolle Ergänzung der Eingabe.

3.3 Sequence-to-Sequence-Ansätze

Um Eingabe- in Ausgabesequenzen zu transformieren, können innerhalb der Sequence-to-Sequence-Generierung verschiedene Methoden verwendet werden. Am vielversprechensten für textuelle Umwandlungen sind dabei: (1) *Summarization*, und (2) *Translation*, welche ebenfalls von populären LLMs wie T5 und BART (vgl. Abschnitt 4.2) benutzt werden [DCLT18, LLG+20, RSR+20].

Summarization Bei der Summarization von Texten sollen Ausgangstexte so reduziert werden, dass die Zieltexte in ihrer Länge deutlich kürzer sind und dabei die wichtigsten Aspekte enthalten. Innerhalb der Summarization gibt es deshalb zwei verschiedene Möglichkeiten der Durchführung. *Extractive Summarization* versuchen Elemente des ursprünglichen Textes wiederzuverwenden und fokussieren sich dabei auf das Weglassen unwichtiger Informationen. *Abstractive Summarization* erzeugen einen neuartigen Zieltext, der Kernaspekte durch eine andere, aber kürzere Ausdrucksweise vermittelt. LLMs überzeugen insbesondere bei Abstractive Summarization [HBCC19, LLG+20]. Dies ist vielversprechend für die Transformation von Anchortexten hin zu Queries, da diese in ihrer Grundstruktur nur schwache Ähnlichkeiten zu Queries aufweisen (vgl. Abschnitt 3.2). Durch das Erlernen komplexer Zusammenhänge innerhalb der Anchortexte könnten LLMs somit die wichtigsten Inhalte identifizieren und diese anschließend in eine Query-artige Struktur umformulieren.

Translation Neben der Summarization von Texten bietet sich als weniger offensichtliche Methode die Translation an. Hierbei ist, im gängigen Sinne, die Übersetzung eines Textes von einer Sprache in eine andere gemeint, wobei sich Ausgangs- und Zielsprache durch bestimmte syntaktische Merkmale auszeichen. Diese gilt es bei der Translation einzuhalten, ohne den Inhalt zu verändern. Im Kontext dieser Arbeit können Queries und Anchortexte als solche Sprachen interpretiert werden. Beide besitzen markante Charakteristiken

(vgl. Abschnitt 3.2), die LLMs bei der Translation beachten müssen. Zu erkennen ist dennoch, dass eine Translation-Aufgabe in diesem Fall schwieriger umzusetzen ist als eine Translation-Aufgabe. Da Anchortexte in einer Vielzahl existieren und zu kurzen Queries übersetzt werden müssen, entspricht dies nicht der gängigen Ausgangslage, bei der Ausgangs- und Zieltext zweier Sprachen von ungefähr gleicher Länge sind. Das auf Translation spezialisierte Pre-Training einiger verwendeter Transformer-Modelle (vlg. Abschnitt 4.2) motiviert aber dennoch diese Betrachtungsweise.

Kapitel 4

Experimente

Zur experimentellen Untersuchung der Generierung von Queries mit Hilfe von Anchortexten werden mehrere gleich strukturierte Verfahrensweisen auf unterschiedlichen Datensätzen mittels Machine-Learning betrachtet. Dabei basieren die nachfolgenden Experimente auf den drei Überlegungen, dass (1) verschiedene Datensätze auch verschiedene Charakteristiken aufweisen, welche für das automatische Erlernen von Generierungsregeln von Vor- oder Nachteil sein können, (2) verschiedene Machine-Learning-Modelle die Problematik mittels einer abweichenden Methode bearbeiten (Summarization oder Translation), die einen Einfluss auf die Effektivität der Modelle hat, und dass (3) die Menge von Trainingsdaten und die Anzahl der Modellparameter ebenfalls Einfluss auf die Modelleffektivität nimmt. In den nachfolgenden Kapiteln wird auf die einzelnen Aspekte dieser Überlegungen eingegangen. Abschnitt 4.1 gibt einen Überblick über die verwendeten Datensätze und stellt einzelne Charakteristiken heraus. Abschnitt 4.2 enthält die betrachteten Machine-Learning-Modelle und liefert eine Auflistung der wichtigsten Merkmale. Abschnitt 4.3 erläutert die Überlegungen für die Verwendung diverser Evaluationsmetriken. Der allgemeine experimentelle Aufbau wird in Abschnitt 4.4 betrachtet, welcher auf die sich ähnelnde Struktur der Experimente sowie die unterschiedlichen Evaluationsstrategien der Modelle eingeht. Abschnitt 4.5 wirft einen Blick auf die Ergebnisse der einzelnen Experimente.

4.1 Datensätze

Die, in dieser Arbeit verwendeten, Teildatensätze entspringen den Sammlungen *MAchine Reading COmprehension*: MS MARCO [NRS+16] und ClueWeb09[1]. Bei Ersterem handelt es sich in seiner Ursprungsform um einen Question-

[1] `https://lemurproject.org/clueweb09/`

Tabelle 4.1: Verwendete Bestandteile der Datensätze: MS MARCO und ClueWeb09. Enthält unter anderem Angaben zu Query-Dokument-Relationen (Qrels) und Anchortexten (AT). Bei Letzterem bezieht sich die Anzahl auf angereicherte Dokumente.

Datensatz (`ir_datasets`-ID)	**Anzahl**			
	Passagen/ Dokumente	Queries	Qrels	AT
`msmarco-document`	3.213.835			
`/train`		367.013	367.013	
`/orcas`		10.405.342	18.823.602	
`/trec-dl-2019`		200	16.258	
`/trec-dl-2019/judged`		43	16.258	
`/anchor-text`				1.703.834
`msmarco-passage`	8.841.823			
`/train`		808.731	532.761	
`/train/judged`		502.939	532.761	
`/dev`		101.093	59.273	
`/dev/small`		6.980	7.437	
`/trec-dl-2019`		200	9.260	
`/trec-dl-2019/judged`		43	9.260	
`clueweb09`	1.040.859.705			
`/trec-web-2009`	503.903.810	50	23.601	

Answering-Datensatz (`msmarco-qna`), welcher aus Query-Logs der Suchmaschine Bing erzeugt wurde. Er umfasst 1,01 Millionen Fragen zu 8,84 Millionen Textpassagen, welche aus 3,56 Millionen Web-Dokumenten extrahiert wurden. Dieser Datensatz kann aufgrund seiner Passagen und Dokumenten in zwei Teildatensätze getrennt werden, welche jeweils für das Training und die Evaluation verschiedener Modelle genutzt werden. Der ClueWeb09-Datensatz ist ein Webcrawl, bestehend aus 1,04 Milliarden Dokumenten und wurde unter anderem im TREC Web Track der Jahre 2009 bis 2012 verwendet [CCS09, CCSC10, CCSV11, CCV12]. In den folgenden Absätzen wird näher auf die einzelnen Charakteristiken dieser Datensätze eingegangen. Des Weiteren wird erläutert, welche davon ausgehenden Teildatensätze Anwendung in dieser Arbeit finden. Eine Einbindung der Datensätze in die Experimente erfolgt durch die `ir_datasets`-Bibliothek[2].

MS MARCO: Dokumente Eine direkte Zuordnung von Query zu Dokument wird im Teildatensatz `msmarco-document` festgehalten. Dieser teilt sich

[2] `https://ir-datasets.com/`

in einen Trainings-, Entwicklungs- und Testdatensatz ein, von denen in dieser Arbeit lediglich der Trainingsdatensatz (`msmarco-document/train`) Anwendung findet und von dem eine Teilmenge zur Evaluation abgespalten wird. Ebenfalls wird der aus `msmarco-document` resultierende Datensatz des TREC 2019 Deep Learning Track [CMY+20] für die Betrachtung von Retrieval-Szenarien verwendet. Eine Reduzierung durch das Entfernen von Queries, zu denen keine Relevanzbewertungen vorliegen, führt zur `judged`-Variante. Darüber hinaus liefert der ORCAS-Datensatz [CCM+20] 18,82 Millionen zusätzliche Query-Dokument-Paare. Diese entspringen aus realen Bedingungen, da sie aus Query-Logs entnommen wurden. Somit sind eine Vielzahl an Queries enthalten, die fehlerbehaftet sind oder nur aus einem Wort bestehen (z.B. `ecxchange rates` oder `forum`). Hinzu kommt, dass `msmarco-document` die URL und den Titel jedes Dokumentes enthält. Eine Übersicht ist Tabelle 4.1 zu entnehmen.

MS MARCO: Passagen Der Teildatensatz `msmarco-passage` umfasst einzelne Passagen aus den zuvor erwähnten Dokumenten und deren zugehörige Queries. Im Gegensatz zu `msmarco-document` weisen alle Queries die Besonderheit auf, dass sie eine wohlgeformte Frage sind beziehungsweise einer großteils grammatikalisch korrekten Form folgen (z.B. `where did last names originate` oder `biggest island of hawaii`). Wie `msmarco-document` teilt sich `msmarco-passage` in einen Trainings-, Entwicklungs- und Testdatensatz ein, von denen der Trainings- und Entwicklungsdatensatz (`msmarco-passage/train` und `msmarco-passage/dev`) verwendet werden. Letzterer wird dabei zur Evaluation benutzt, da der deutlich geringere Umfang des Trainingsdatensatzes keine Aufteilung erlaubt. Von diesen Datensätzen können ebenfalls kleinere Varianten (`judged`und `small`), durch den Wegfall von Queries ohne Relevanzbewertung zu Dokumenten, abgeleitet werden. Zur Evaluation von Retrieval-Szenarien wird ebenfalls der Datensatz des TREC 2019 Deep Learning Tracks als `judged`-Variante verwendet. Eine Übersicht des `msmarco-passage` Datensatzes ist in Tabelle 4.1 zu sehen.

MS MARCO: Anchortexte Da die zuvor erwähnten Teildatensätze keine Anchortexte enthalten, werden sie mittels des Webis MS MARCO Anchor Text 2022 Datensatzes [FGP+22] erweitert. Dieser fügt für 1,70 Millionen Dokumente des `msmarco-document`-Datensatzes Anchortexte hinzu. Diese Anchortexte wurden aus Common-Crawl-Snapshots zwischen 2016 und 2021 extrahiert und pro Dokument auf 1.000 Stück limitiert. Insgesamt sind 6% aller Dokumente von dieser Limitierung betroffen. Da die Zuordnung von Anchortexten über die Identifikatoren der Dokumente aus `msmarco-document` erfolgt und `msmarco-passage` diese nicht enthält, wird hierbei ein Mapping des `msmarco-qna`-Datensatzes verwendet. Dieses Mapping ordnet Passagen zu

deren Ausgangsdokumenten zu und ermöglicht somit ebenfalls eine Zuordnung von Anchortexten.

ClueWeb09 Die Verwendung des `clueweb09`-Datensatzes ergibt sich aus der Tatsache, dass die Retrieval-Datensätze aus dem TREC 2019 Deep Learning Track für `msmarco-passage` und `msmarco-document` zum Großteil aus wohlgeformten Queries bestehen. Daher wird zusätzlich der dem `clueweb09` entspringende TREC 2009 Web Track Datensatz [CCS09] für die Betrachtung von Retrieval-Szenarien verwendet. Dieser enthält stichwortartige Queries, deren Eigenschaften ähnlich zu denen des ORCAS-Datensatzes [CCM+20] sind (z.B. `dinosaurs` oder `website design hosting`). Somit dient er zum Nachteilsausgleich der `msmarco-document` Modelle, bei denen das Erlernen von weniger wohlgeformten Queries aufgrund der ORCAS-Trainingsdaten erwartet wird. Der TREC 2009 Web Track Datensatz bezieht sich lediglich auf englischsprachige Dokumente. Eine Zuordnung von Anchortexten erfolgt mittels dedizierter Sammlungen[3] von Hiemstra [Hie10], die 88% aller `clueweb09`-Dokumente der Kategorie A des TREC 2009 Web Tracks [CCS09] abdecken. Pro Dokument wurden maximal 10 Megabyte Anchortexte gesammelt.

4.2 Machine-Learning-Modelle

Neben den unterschiedlichen Teildatensätzen werden in dieser Arbeit diverse Machine-Learning-Modelle, basierend auf der Transformer-Architektur [VSP+17] (vgl. Abschnitt 3.1), eingesetzt. Diese betrachten die Generierung von Queries durch Anchortexte aus den verschiedenen Blickwinkeln einer Summarization- beziehungsweise Translation-Aufgabe (vgl. Abschnitt 3.3). Die verwendeten Transformer-Modelle umfassen: *BERT* [DCLT18], *BART* [LLG+20] und *T5* [RSR+20] in den jeweiligen Größenvarianten `large` und `base`. In Hinblick auf die Evaluation der Modelle werden diese mit *Doc2Query* [NYLC19, FNLE19] verglichen, einem Transformer-Modell, welches künstliche Queries aus Web-Dokumenten erzeugen kann. Die folgenden Absätze erläutern die Besonderheiten der einzelnen Modelle und betrachten die Gründe für die Verwendung.

BERT Als eines der ersten großen Transformer-Modelle mit Erfolg in den verschiedensten Einsatzgebieten [NYLC19, NC19, ARTL19, SQXH19], entwickelten Devlin et al. [DCLT18]: *Bidirectional Encoder Representations from Transformers* (BERT). Dabei orientiert sich BERT an der originalen Transformer-Architektur von Vaswani et al. [VSP+17]. Durch die Verwendung des

[3] `https://djoerdhiemstra.com/2010/anchor-text-for-clueweb09-category-a/`

Tabelle 4.2: Übersicht der verwendeten Machine-Learning-Modelle. Eingabelimits sind in Token angegeben. (* - Hardwarelimitierung, theoretisch unlimitiert)

Modell	Jahr	Referenz	Eingabelimit	Parameter	
				`base`	`large`
BERT	2018	[DCLT18]	512	110M	340M
BART	2019	[LLG+20]	1024	140M	400M
T5	2019	[RSR+20]	512*	220M	770M
D2Q	2019	[NYLC19, FNLE19]	512*	220M	—

Self-Attention-Mechanismus in einer bidirektionalen Encoder-Architektur, ist BERT in der Lage, Textstücke auf einmal und mit vollständigem Kontext zu verarbeiten. Im Gegensatz zu Modellen wie LSTMs [HS97] kann BERT kontextuelle Bezüge innerhalb eines Textes auch über lange Distanzen erhalten. Ein Thema am Textende kann somit im Kontext des Textanfangs stehen. Mittels Pre-Training wurde BERT auf ungelabelten Daten trainiert. Dies erlaubt durch Fine-Tuning eine Abstimmung der Parameter mittels gelabelter Daten für eine spezielle Aufgabe wie zum Beispiel Question-Answering. Neben der Unterteilung in `large` und `base` existieren die Varianten `uncased` und `cased`. Die `cased`-Variante ist in der Lage, zwischen Groß- und Kleinschreibung zu unterscheiden. Aufgrund der Tatsache, dass die verwendeten Datensätze sich auf die englische Sprache fokussieren und insbesondere Queries mehrheitlich klein geschrieben werden (vgl. Abschnitt 3.2), wird die `uncased`-Variante verwendet.
Der Einsatz von BERT wird neben diesen Gründen durch die Popularität im Bereich der Textverarbeitung motiviert. Zusätzlich dient BERT als Grundlage von weiteren, in dieser Arbeit untersuchten, Modellen. Eine genauere Spezifikation der verwendeten Modelle kann der Tabelle 4.2 entnommen werden.

BART Folgend auf BERT entwickelten Lewis et al.: *Bidirectional and Auto-Regressive Transformers* (BART) [LLG+20]. BART leitet sich ebenfalls von der Transformer-Architektur von Vaswani et al. [VSP+17] ab. Im Gegensatz zu BERT nutzt BART aber zusätzlich einen autoregressiven Decoder, welcher in der Lage ist, anhand von Wortwahrscheinlichkeiten das nächste Token der Ausgabe vorherzusagen. Zusätzlich unterscheidet sich das Pre-Training von BART. BERT nutzte Masked-Language-Modeling, wobei zufällige Eingabetoken maskiert wurden und vom Modell erraten werden mussten. BART hingegen nutzte zusätzlich eine Vielzahl von Funktionen, die die Struktur des Eingabetextes stärker veränderten. So wurden beispielsweise Sätze in eine zufällige Reihenfolge gebracht oder Token gelöscht, sodass BART diese identifizieren

musste. All dies erlaubt BART ein noch komplexeres Verständnis der Sprache und lässt es, insbesondere in Bereichen, in denen komplexe Sprachmuster verarbeitet werden müssen (z.B. Abstractive Summarization), besser agieren als BERT [LLG+20].
Die Betrachtungsweise der Generierung von künstlichen Queries durch Anchortexte als Summarization-Aufgabe motiviert die Verwendung von BART. Insbesondere wird die Möglichkeit der Verarbeitung von längeren Textstücken durch ein größeres Eingabelimit als vielversprechend bewertet. Einen Überblick zu BART und ein Vergleich zu anderen Modellen ist Tabelle 4.2 zu entnehmen.

T5 Der Erfolg von Transformer-Modellen auf den unterschiedlichsten Anwendungsgebieten [PVU+18, DXX18, WDS+20] motivierte Raffel et al. zur Entwicklung eines universell einsetzbaren Modells: *Transfer Learning with a Unified Text-To-Text Transformer* (T5) [RSR+20]. Dieses ist in der Lage, verschiedene Aufgaben wie beispielsweise Summarization oder Translation zu bearbeiten. Ermöglicht wird dies über Präfixe, die dem Modell zur Unterscheidung übergeben werden. Die Eingabesequenz wird an den Präfix angehangen und als neue Eingabe an das T5-Modell übergeben. T5 orientiert sich wie BART am Masked-Language-Modeling von BERT, erweitert dies aber ebenfalls um zusätzliche Korruptionsfunktionen.
Ein großer Vorteil von T5 ist die Implementierung des *Relative-Attention-Mechanismus* [SUV18]. Dieser ist eine Abwandlung des Self-Attention-Mechanismus von Transformern und erlaubt beliebig lange Eingaben. Einzige Limitierung ist hierbei die Hardware, da die Speicheranforderung sich quadratisch zur Eingabelänge verhält. T5 übertraf in 18 von 24 betrachteten Disziplinen die Effektivität der bis dato besten Modelle. Die Tatsache, dass es im Pre-Training für Summarization und Translation optimiert wurde, macht es zum nächsten Kandidaten für die Generierung künstlicher Queries. Ein Überblick findet sich in Tabelle 4.2. Es ist hierbei anzumerken, dass T5 in fünf Ausführungen existiert und Modelle mit bis zu 11 Milliarden Parametern existieren. Da diese für die verwendete Hardware einer NVIDIA A100 Grafikkarte zu groß sind, wurde sich ebenfalls auf die `base`- und `large`-Varianten beschränkt.

D2Q Um unter anderem die Effektivität der, in dieser Arbeit, trainierten Modelle zu vergleichen, wird das von Nogueira et al. [NYLC19, FNLE19] entworfene Modell *Doc2Query* (D2Q) herangezogen. Dieses wurde entwickelt, um dem Vocabulary-Mismatch-Problem im Information Retrieval entgegenzuwirken und generiert zu einem Eingabedokument, passende Queries, die das Dokument beantworten kann. Diese können dann vor der Indexierung an das Ursprungsdokument angehangen werden.
In seiner Ursprungsform handelt es sich bei Doc2Query um ein Transformer-

Modell, basierend auf der Struktur von Vaswani et al. [VSP+17]. Dieses wurde auf den Datensätzen `msmarco-passage` und TREC-CAR [DVRC17] trainiert und ist in der Lage, wohlgeformte Queries wie beispielsweise: `what is the latitude on a map` oder `how long to bake a muffin` zu erzeugen. Eine Erweiterung des Modells fand durch den Ersatz des ursprünglich verwendeten Transformer-Modells gegen T5 statt [FNLE19]. Hierbei wurde das vorangegangene Trainingsschema repliziert und dennoch eine Effektivitätssteigerung beobachtet. Diese beruhte, laut Nogueira und Lin, auf dem Pre-Training und nicht der gestiegenen Modellkomplexität seitens T5 [FNLE19].
Die Ähnlichkeit der unterliegenden Aufgabe macht Doc2Query zu einem idealen Vergleichskandidaten für diese Arbeit. Der durch TREC-CAR erlangte Vorteil eines umfangreicheren Pre-Trainings kann mit den verwendeten Datensätzen nicht gelindert werden. Dennoch lassen sich Vergleiche in Bezug auf syntaktische oder semantische Überlegenheit zwischen Doc2Query und anderen Modellen hin zur Zielquery durchführen. Ein technischer Überblick von Doc2Query lässt sich Tabelle 4.2 entnehmen.

4.3 Metriken

Zur Effektivitätsbewertung der trainierten Modelle auf unbekannten Daten kommen diverse Metriken zum Einsatz. Diese Auswahl an Metriken soll unter anderem eine möglichst große Vergleichbarkeit mit anderen Studien gewährleisten und Schwächen einzelner Metriken ausgleichen. Die Metriken unterteilen sich dabei in zwei Bewertungskriterien. Um die syntaktische beziehungsweise semantische Ähnlichkeit von generierten und echten Queries zu bewerten, werden Metriken zur Qualitätsprüfung verwendet. Darunter zählen *BLEU* [PRWZ02], *ROUGE* [Lin04] und *BERTScore* [ZKW+19]. Zusätzlich werden inhaltliche Beziehungen zwischen generierten Queries und Dokumenten innerhalb der Retrieval-Szenarien durch Ranking-Metriken bewertet. Zu diesen gehören Precision-basierte Metriken wie: *nDCG@k* [JK02], *MRR* [KV00] und *Precision@k* [VR79], aber auch Recall-basierte Metriken wie: *mAP* [Har92a] und *Recall@k*. In den folgenden Abschnitten werden diese einzelnen Metriken näher beschrieben, kritisch betrachtet und deren Verwendung in dieser Arbeit motiviert.

BLEU Entwickelt zur automatisierten Bewertung von maschinell übersetzten Textstücken, ist *BiLingual Evaluation Understudy* (BLEU) [PRWZ02] heute eine der am häufigsten verwendeten Metriken auf dem Gebiet der Machine-Translation. Mittels n-Grammen können generierte Kandidatentexte mit mensch-

[4] `https://huggingface.co/spaces/evaluate-metric/sacrebleu`

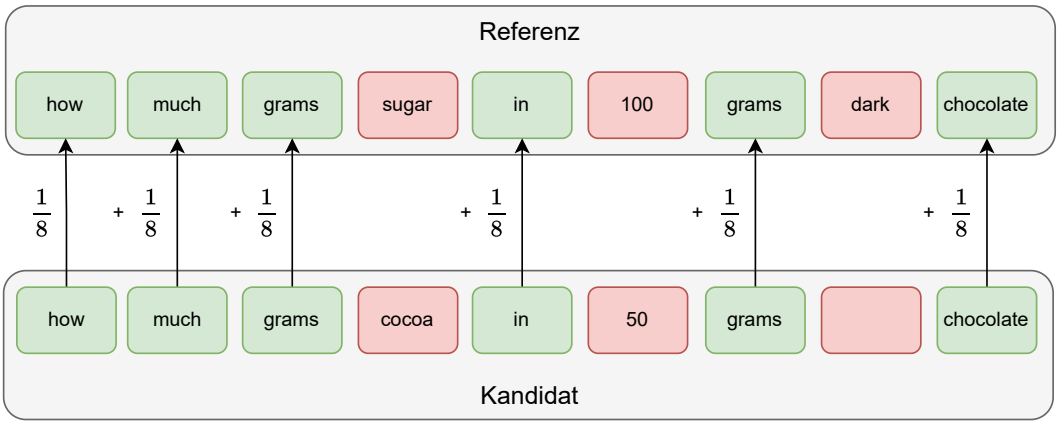

Abbildung 4.1: Beispielhafte Berechnung eines BLEU-Scores zwischen einer Referenz-Query und einer Kandidaten-Query mittels Unigrammen. Gekennzeichnet sind Übereinstimmungen (grün) und Unterschiede (rot).

Tabelle 4.3: Parameterübersicht von BLEU beziehungsweise SacreBLEU für die HuggingFace-API.[4]

Parameter	Wert
`smooth_method`	`'exp'`
`tokenize`	`'13a'`
`lowercase`	`False`
`force`	`False`
`use_effective_order`	`False`

lich erzeugten Referenztexten verglichen werden. BLEU ermittelt die Menge von Termen eines Kandidaten, welche innerhalb der zugehörigen Referenzen auftreten. Es agiert somit Precision-basiert und ermittelt die Überlappung mehrerer Arten von n-Grammen. Veranschaulicht wird dies durch die Abbildung 4.1. Der BLEU-Score von 0,75 ergibt sich im Beispiel als Division der Anzahl an Übereinstimmungen geteilt durch die Länge des Kandidatentextes. Zusätzlich zu der dargestellten Funktionsweise besitzt BLEU noch einen *Brevity-Penalty*. Dieser soll verhindern, dass aufgrund der Berechnung kurze Kandidatentexte automatisch bessere Scores erhalten.

Die alleinige Betrachtung von Überlappungen durch n-Gramme führt zu diverser Kritik bezüglich des populären Einsatzes von BLEU innerhalb des Gebiets der Machine-Translation. Zwar berichten Papineni et al. in ihrer Arbeit, dass BLEU stark mit dem menschlichen Urteilsvermögen korreliert [PRWZ02], darauf folgende Veröffentlichungen zweifeln aber an dieser Aussage. Callison-

Burch et al. [COK06] geben an, dass bessere BLEU-Werte allein noch kein Indiz für ein besseres Translation-Modell sind. Isozaki et al. [IHD+10] identifizieren BLEU als ungeeignete Metrik für die Bewertung von Distant-Language-Pairs. Mathur et al. [MBC20] und Freitag et al. [FRM+22] propagieren sogar die Abschaffung von BLEU als Standardmetrik für Machine-Translation. Dennoch besteht die Verwendung von BLEU in aktuellen Veröffentlichungen und wird daher aus Gründen der Vergleichbarkeit in dieser Arbeit ebenfalls benutzt.

Durch die inkonsistente Parametrisierung kommt es beim Einsatz von BLEU zwischen zahlreichen wissenschaftlichen Arbeiten, zu Inkonsistenzen bei der Angabe von Ergebnissen [Pos18]. Die Parametrisierung von BLEU, durch die beispielsweise die Länge der n-Gramme oder Glättungsfunktionen bestimmt werden, macht eine Vergleichbarkeit bei abweichender Parameterwahl ungenau. Um dagegen Abhilfe zu schaffen, propagiert Post [Pos18] *SacreBLEU*, welche der empfohlenen Implementation der *Conference on Machine Translation*[5] folgt. Zusätzlich empfiehlt Post eine genaue Angabe der verwendeten Parameter. Die in dieser Arbeit verwendeten Parameter können, dieser Empfehlung folgend, der Tabelle 4.3 entnommen werden.

Trotz der ambivalenten Meinungen soll BLEU in dieser Arbeit als repräsentativer Wert für die Bewertung von künstlich erzeugten Queries zum Einsatz kommen. Durch den Einsatz von BART-Modellen kann die Generierung als Übersetzung von Anchortexten hin zu Queries angesehen werden. Somit kann BLEU, nach Empfehlung von Callisson-Burch et al. [COK06], zum internen Vergleich ähnlicher Translation-Modelle verwendet werden. Hinzu kommt die Tatsache, dass weitere Metriken zur Evaluation verwendet werden. Alle angegebenen BLEU-Scores basieren auf der Implementierung von SacreBLEU, werden aus Übersichtlichkeitsgründen aber unter BLEU aufgeführt.

ROUGE Eine weitere, sehr verbreitete Metrik ist *Recall-Oriented Understudy for Gisting Evaluation* (ROUGE) [Lin04]. Im Gegensatz zum Precision-basierten BLEU, ist ROUGE Recall-basiert und vertauscht dadurch die Rolle von Kandidaten- und Referenztext. ROUGE berechnet die Menge an Termen innerhalb der Referenzen, welche im Kandidaten vorhanden sind. ROUGE kommt besonders im Bereich der Text-Summarization zum Einsatz. Trotz dieser Unterschiede ist die prinzipielle Berechnung von ROUGE ähnlich zu der von BLEU. So benutzt auch ROUGE n-Gramme zur Evaluation, wie in Abbildung 4.2 zu sehen ist. Der ROUGE-Score von 0,67 berechnet sich im Beispiel als Verhältnis von Übereinstimmungen zur Länge des Referenztextes. Zusätz-

[5] `https://statmt.org/wmt22/`

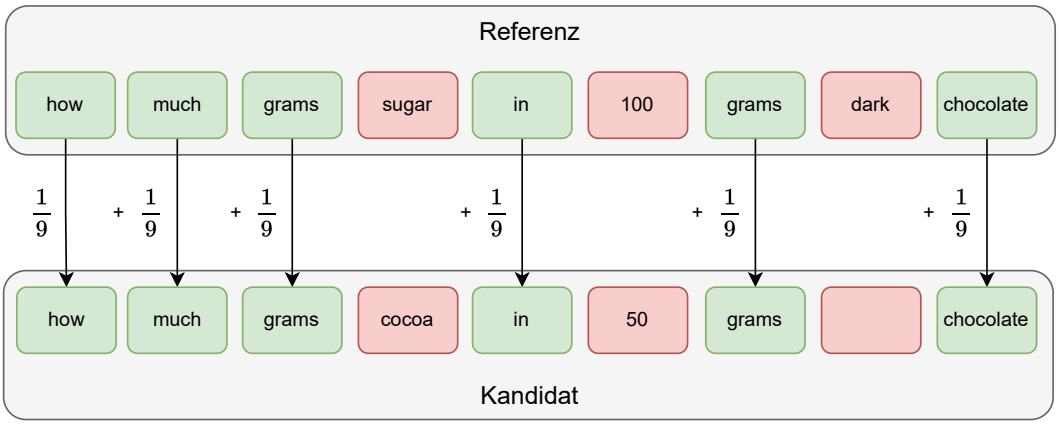

Abbildung 4.2: Beispielhafte Berechnung eines ROUGE-1 Recall-Scores zwischen einer Referenz-Query und einer Kandidaten-Query mittels Unigrammen. Gekennzeichnet sind Übereinstimmungen (grün) und Unterschiede (rot).

lich enthalten aktuelle ROUGE-Implementierungen neben dem Recall auch Precision- und F1-Scores. Je nach Art der n-Gramme unterscheidet sich die Berechnung von ROUGE. In der Praxis werden am häufigsten ROUGE-1 und ROUGE-2 verwendet, welche in ihrer unterliegenden Berechnung Unigramme bzw. Bigramme nutzen [GCS22]. Ebenfalls wird häufig ROUGE-L angegeben, welches mit n-Grammen arbeitet, die der Länge der größten gemeinsamen Überlappung zwischen Referenz- und Kandidatentext entsprechen.

Eine 2022 veröffentlichte Erhebung gab an, dass 100 % aller Veröffentlichungen von neuen Summarization-Modellen auf Computer-Linguistic-Konferenzen im Jahr 2021 ROUGE verwenden [GCS22]. Dennoch hat auch ROUGE, insbesondere im Kontext der Summarization, seine Grenzen. So können Extractive Summarizations mittels ROUGE besser überprüft werden als Abstractive Summarizations [NA15], da letztere Wörter verwenden, welche im Ausgangstext gar nicht vorkommen und somit eine lexikographische Überlappung unwahrscheinlich ist. Hierbei sei angemerkt, dass diese Überlegungen stark davon abhängig sind, welche Methode für das Zusammenfassen des Referenztextes genutzt wurden. Kryscinski et al. [KKM$^+$19] kritisieren eine geringe Korrelation zwischen menschlichem Urteilsvermögen und ROUGE. Sie sehen in der fehlenden Überprüfung von sachlicher Korrektheit einen wichtigen Mangel. Zusätzlich identifizieren Fabri et al. [FKM$^+$20] ROUGE-3 und ROUGE-4 als geeignetere Evaluationsmethode für Zusammenfassungen als ROUGE-1 und -2. Die beiden kritischen Betrachtungen untersuchen dabei hauptsächlich große Textstücke wie Nachrichtenartikel. Allgemein betrachtet das Teilgebiet der Summarization hauptsächlich die Zusammenfassung längerer Texte, die

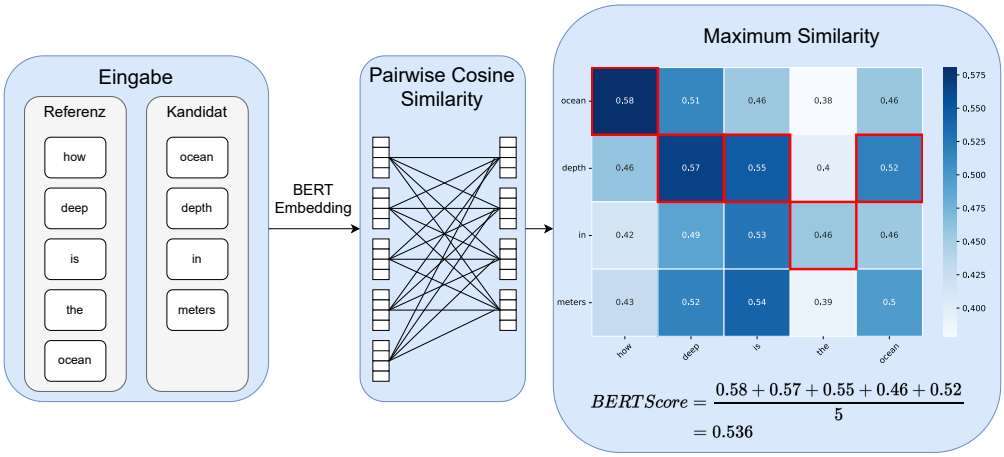

Abbildung 4.3: Beispielhafte Berechnung eines BERTScores zwischen einer Referenz-Query und einer Kandidaten-Query.

somit ebenfalls innerhalb der Evaluation herangezogen werden. Da ROUGE in dieser Arbeit aber zum Vergleich kurzer Queries verwendet wird, werden die Kritikpunkte als nicht relevant erachtet. Durch den Einsatz von Modellen wie T5, welches in seinem Training Summarization-basierte Methoden erlernt hat [RSR+20], ist ROUGE eine geeignete Metrik zum Vergleich der verschiedenen Modelle. Wie bei BLEU muss ROUGE immer in Verbindung mit anderer Metriken betrachtet werden, um die erwähnten potentiellen Schwächen auszugleichen.

BERTScore Als eine der neusten Metriken zur automatischen Evaluierung von maschinell erzeugten Texten entwickelten Zhan et al. [ZKW+19] den sogenannten *BERTScore*. Dieser kann sowohl für Translation- als auch Summarization-Aufgaben eingesetzt werden und soll die Probleme der etablierten Metriken BLEU und ROUGE umgehen. Dies funktioniert über Contextual-Embeddings, welche mittels eines BERT-Modells [DCLT18] erzeugt werden. Dabei werden sowohl der Referenz- als auch der Kandidatentext innerhalb eines Vektorraumes eingebettet. Durch *Pairwise-Cosine-Similarity* wird der Winkel zwischen diesen Vektoren verglichen. Je kleiner dieser Winkel ist, desto ähnlicher sind die Textstücke. Im Gegensatz zu n-Gramm-basierten Verfahren können die Embeddings einen stark kontextuellen Bezug zwischen den Wörtern modellieren. Dies erlaubt es, dass auch Kandidatentexte mit Synonymen einen hohen BERTScore erhalten.

Die Embeddings hängen stark vom verwendeten Machine-Learning-Modell ab. Je besser dieses Modell für eine spezifische Aufgabe angepasst wurde, umso

besser sind die Embeddings. Hanna und Bojar [HB21] konnten jedoch feststellen, dass BERTScore nicht in der Lage ist, schlechten Kandidaten einen niedrigen Score zu geben, wenn dieser eine hohe lexikographische Überlappung zum Referenztext aufweist. BERTScore findet in dieser Arbeit seinen Einsatz, da dieser durch den kontextuellen Vergleich zwischen Referenz und Kandidat eine weitere Evaluationsebene schafft. Neben einer syntaktischen Ähnlichkeit, sollten künstlich generierte Queries ebenfalls in ihrer semantischen Ähnlichkeit gleiche Merkmale wie echte Queries aufweisen.

Precision Bereits seit der ersten TREC-Konferenz im Jahr 1992 wird die Ranking-Metrik *Precision@k* (P@k) [VR79] zur Evaluation von Retrieval-Systemen verwendet [Har92b]. Sie berechnet sich, für eine gefundene Dokumentenliste, als Quotient der Anzahl der retrievten relevanten Dokumente d_{rel} geteilt durch die Anzahl aller retrievten Dokumente d_{ret}:

$$Precision@k = \frac{|d_{rel}| \cap |d_{ret}|}{|d_{ret}|} \tag{4.1}$$

Der Parameter k bestimmt hierbei bis zu welchem Rang in der Ergebnisliste die Precision gemessen wird. Je größer dieses k gewählt wird, umso umfänglicher bewertet die Metrik das Retrieval-System. Im Kontext dieser Arbeit kommt sie zum Einsatz für den Modellvergleich innerhalb des `clueweb09`-Datensatzes und aus Gründen der Vergleichbarkeit innerhalb des TREC 2009 Web Tracks [CCS09].

MRR Der *Mean Reciprocal Rank* (MRR) ist eine weitere simple Variante zur Evaluierung der Effektivität von Retrieval-Systemen [KV00] und ist ebenfalls Precision-basiert. Wie bei P@k, betrachtet MRR die gefundene Ergebnisliste, gegeben einer Query. Innerhalb dieser besitzt jedes Dokument einen Rang r_i, welcher die Position angibt, an der das Dokument in dieser Liste auftaucht. Der MRR berechnet sich durch die reziproken Ränge des ersten relevanten Dokumentes innerhalb dieser Liste. Für n Queries berechnet sich der Mittelwert wie folgt:

$$MRR = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{r_i} \tag{4.2}$$

Die Verwendung des MRR wird durch den Einsatz im TREC 2019 Deep Learning Track [CMY+20] motiviert. Dessen Evaluationsdatensatz findet in dieser Arbeit ebenfalls eine Anwendung innerhalb der Retrieval-Szenarien auf `msmarcodocument` und `msmarco-passage`.

nDCG Die Ranking-Metrik *normalised Discounted Cumulative Gain* (nDCG@k) [JK00, JK02] wird für die Bewertung von Retrieval-Experimenten innerhalb des Information Retrieval verwendet.

Die Evaluation beim Retrieval von Dokumenten durch Queries erfolgt beim nDCG@k mittels zwei Voraussetzungen, welche mit der menschlichen Intuition beim Bewerten von Suchergebnislisten einhergehen: (1) Sehr relevante Dokumente sollten besonders früh auftreten und (2) weniger relevante Dokumente sollten mit absteigender Relevanz immer später in der Ergebnisliste auftauchen. Der nDCG@k leitet sich demnach aus dem *Cumulative Gain* (CG) und weiterführend dem *Discounted Cumulative Gain* (DCG) ab. Für das Retrieval einer Dokumentenliste der k besten Dokumente durch Eingabe einer Query berechnet sich der CG als Summe aller Relevanzlabels l_i bis zum Index n:

$$CG_k = \sum_{i=1}^{k} l_i \tag{4.3}$$

Der DCG führt eine logarithmische Bewertungsfunktion zur Abstrafung von später auftauchenden, relevanten Dokumenten ein:

$$DCG_k = \sum_{i=1}^{k} \frac{l_i}{\log_2 i + 1} \tag{4.4}$$

Schließlich wird der DCG mittels des *Ideal Discounted Cumulative Gain* (IDCG) normalisiert. Der IDCG entspricht dem Optimalfall, dass die Dokumentenliste nach absteigender Relevanz geordnet ist. Somit ergibt sich der nDCG@k als:

$$nDCG@k = \frac{DCG_k}{IDCG_k} \tag{4.5}$$

Durch seine Korrelation zur menschlichen Intuition eignet sich der nDCG@k als Metrik zur Relevanzbewertung zwischen Dokumenten und echten beziehungsweise generierten Queries. Der nDCG@k für die Vergleichbarkeit innerhalb des TREC 2019 Deep Learning Tracks [CMY+20] verwendet.

Recall Wie schon P@k wird auch *Recall@k* (R@k) seit 1992 in TREC-Konferenzen zur Evaluation eingesetzt [Har92b]. Neben den zuvor erwähnten Precision-basierten Metriken, werden ebenfalls Recall-basierte Metriken verwendet. Dabei legen diese einen stärkeren Fokus auf die Abdeckung eines

Tabelle 4.4: Hyperparameterwahl für das Modelltraining in Abschnitt 4.4

Hyperparameter	**Werte**	
	`msmarco-document`	`msmarco-passage`
Epochen	5	2, 5, 100
Batch-Size	4, 8	8, 16
Gradient-Accumulation	4, 8	-
Learning-Rate	$1e-4$	$1e-4$
Warmup-Ratio	0.06	0.06
Step-Size	1000	1000
Early-Stopping (Steps)	3	3

Retrieval-Systems. R@k betrachtet hierbei, wie groß der Anteil an retrievten, relevanten Dokumenten im Verhältnis zu allen relevanten Dokumenten ist:

$$Recall@k = \frac{|d_{rel} \cap d_{ret}|}{|d_{rel}|} \tag{4.6}$$

Die Nutzung des R@k wird insbesondere für die Vergleichbarkeit mit Doc2Query [NYLC19, FNLE19] motiviert.

mAP Die Verwendung von *mean Average Precision* (mAP) wird hauptsächlich aus Gründen der Vergleichbarkeit, durch den Einsatz als Metrik im TREC 2019 Deep Learning Track [CMY+20], motiviert. Für einen vollumfänglichen Vergleich wird diese deshalb ebenfalls angegeben. Der mAP berechnet sich als Mittelwert aller Queries N über die iterativ betrachteten Precision-Werte und legt, neben der Anzahl der relevanten Dokumente, einen Fokus auf die Ordnung dieser innerhalb der Rückgabe:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP@i \tag{4.7}$$

$$AP@k = \frac{1}{|d_{rel}|} \sum_{k=1}^{d_{ret}} P@k * relevant(k) \tag{4.8}$$

$$relevant(k) = \begin{cases} 1, \text{wenn Dokument an Position k relevant ist} \\ 0, \text{sonst} \end{cases} \tag{4.9}$$

4.4 Experimenteller Aufbau

Um die zuvor dargestellten Datensätze, Machine-Learning-Modelle und Metriken zusammenzuführen, orientiert sich die Struktur der Experimente im

Wesentlichen an den folgenden vier Phasen: (1) Datensatzauswahl und Vorverarbeitung, (2) Modelltraining, (3) Erzeugung künstlicher Queries, und (4) Evaluation. Im Folgenden wird näher auf die einzelnen Experimente eingegangen und dargestellt, wie die genannten Aspekte innerhalb dieser umgesetzt wurden.

4.4.1 Training und Evaluation auf `msmarco-document`

Die Erweiterung von `msmarco-document` [NRS+16] durch die Teildatensätze: ORCAS [CCM+20] und Webis MS MARCO Anchor Text 2022 [FGP+22], macht ihn zum größten und realitätsnächsten Datensatz dieser Arbeit (vgl. Abschnitt 4.1). Allerdings bedeutet dies auch die umfänglichste Vorverarbeitung. Da die ORCAS-Queries von echten Nutzern in echten Situationen erstellt wurden, ergeben sich verrauschte Daten durch andersprachige Texte oder Nicht-ASCII-Symbole. Bei verrauschten Daten handelt es sich um Einträge, die für das Modelltraining, innerhalb dieser Arbeit, zur Beeinträchtigung des Fine-Tuning der Modelle führen können. Dem Entfernen dieser Daten schreiben wir eine große Bedeutung zu, da bereits existierende Forschung den negativen Effekt dieses Rauschens untersucht hat [KKP06, HLX15, GG19].

Zur Reduzierung des Anteils der verrauschten Daten entfernen wir Einträge, bei denen die Query: (1) kürzer als fünf Zeichen ist (1,07%), (2) länger als 50 Zeichen ist (0,21%), oder (3) Nicht-ASCII Symbole enthält (0,27%). Um Doppelungen oder verrauschte Bestandteile innerhalb der Anchortexte zu entfernen, wird eine zusätzliche Vorverarbeitung durchgeführt. Dabei werden keine kompletten Einträge im Datensatz entfernt, sondern einzelne Anchortexte innerhalb der Einträge. Ein Anchortext wird entfernt, wenn er entweder schon einmal innerhalb eines Eintrags vorkommt, oder ein Nicht-ASCII-Symbol enthält. Nachträglich werden die übrigen Anchortexte konkateniert. Da bei dieser Bereinigung Einträge ohne Anchortexte entstehen können, werden im Nachgang alle Einträge entfernt, bei denen die konkatenierten Anchortexte kürzer als fünf Zeichen sind (0,53%). Somit bleiben am Ende der Vorverarbeitung 16.878.897 Query-Dokument-Paare übrig, wobei jedem Dokument eine Reihe von Anchortexten zugeordnet ist. Da zu einer Query mehrere Dokumente gehören können, wird, zur Bildung der finalen Datensätze, zufällig ein Dokument pro Query ausgewählt. Am Ende wird ein Trainings-, Validierungs- und Testdatensatz mit jeweils 1.000.000, 20.000 beziehungsweise 20.000 zufällig gewählten Query-Dokument-Paaren erstellt. Um die Vergleichbarkeit des vor- und unverarbeiteten Datensatzes zu erhöhen, werden beide Datensätze aus denselben Einträgen erstellt.

Das Training von BERT, BART und T5 wird über die Transformers[6]- beziehungsweise Simple-Transformers[7]-Bibliotheken realisiert. Untersucht werden jeweils die `base`- und `large`-Variante der Transformer-Modelle, auf denen das Fine-Tuning mit unterschiedlich vielen Query-Dokument-Paaren (10.000, 25.000, 50.000, 100.000, 500.000 und 1.000.000) erfolgt. Zusätzlich werden für jedes Transformer-Modell und jede Trainingsdatenanzahl vier unterschiedliche Eingabemethoden überprüft: (1) Vorverarbeitete Anchortexte, (2) unverarbeitete Anchortexte, (3) vorverarbeitete Anchortexte mit Dokumententext und Titel, und (4) unverarbeitete Anchortexte mit Dokumententext und Titel. Motiviert wird dies durch die Überlegung, dass Modelle wie Doc2Query [NYLC19, FNLE19] aus Dokumententexten schon gute künstliche Queries produzieren können. Eine Anreicherung von Anchortexten mit Dokumententexten könnte hierbei eine zusätzliche Diversität einführen, welche sich in den generierten Queries bemerkbar macht. Die T5-Modelle besitzen eine weitere Besonderheit, da sie durch ihr Pre-Training in der Lage sind, sich durch verschiedene Befehle auf unterschiedliche NLP-Aufgaben zu fokussieren. Einer dieser Befehle lautet: `summarize`. Zu den zuvor genannten Eingabemethoden werden T5-`base`-Modelle aller Größen zusätzlich in einer Variante mit diesem Befehl trainiert. Insgesamt werden in diesem Experiment 150 Modelle trainiert.

Zur Evaluation werden zwei verschiedene Betrachtungsweisen verfolgt. Beginnend mit der Analyse syntaktischer und semantischer Ähnlichkeiten zu echten Queries, werden, mit den 20.000 Einträgen des Testdatensatzes, künstliche Queries durch die trainierten Modelle erzeugt. Hierbei werden zwei verschiedene Decoding-Strategien untersucht. Um die bestmögliche Query in Bezug auf Wortwahrscheinlichkeiten zu generieren, wird *Greedy-Decoding* verwendet. Der Decoder wählt dabei immer das wahrscheinlichste Token, welches als nächstes für die Ausgabesequenz generiert wird. Im Gegensatz dazu erzeugt *Top-k-Sampling* diversere Queries, indem die Wahrscheinlichkeiten der besten $k + 1$ Token auf Null gesetzt und Ausgabesequenzen nur mittels Variierung der besten k Token generiert werden. Die Ähnlichkeiten werden durch die zuvor erwähnten Metriken: ROUGE, BLEU und BERTScore (vgl. Abschnitt 4.3) bewertet.

Des Weiteren werden die inhaltlichen Beziehungen der künstlichen Queries zu den Ausgangsdokumenten untersucht. Dies erfolgt in Retrieval-Szenarien auf dem `msmarco-document/trec-dl-2019/judged` Datensatz (vgl. Abschnitt 4.1). Da die Generierung künstlicher Queries für alle 3,21 Millionen Dokumente einen enormen zeitlichen Aufwand bedeuten würde, werden

[6] `https://huggingface.co/docs/transformers`

[7] `https://simpletransformers.ai/`

diese nur für 9.032 Relationen (55,6%) mit zugeordneten Anchortexten erzeugt. Diese umfassen 8.923 Dokumente. Die Evaluation berücksichtigt Unterschiede im Retrieval auf der vollständigen und reduzierten Relationsmenge. Den Angaben von Nogueira et al. [NYLC19, FNLE19] folgend, werden für die Retrieval-Szenarien pro Transformer-Modell und pro Dateneintrag 80 Queries mittels Top-k-Sampling generiert. Beam-Search wurde nicht betrachtet, da Top-k-Sampling bei Doc2Query zu besseren Queries führt. [NYLC19, FNLE19]

Bezüglich der Wahl der Hyperparameter werden alle Transformer-Modelle über 5 Epochen mit Early-Stopping und einer Step-Size von 1000 trainiert. Es wird eine Learning-Rate von 0,0001 und eine Batch-Size von 4 beziehungsweise 8 verwendet. Durch die Nutzung von Gradient-Accumulation wird eine effektive Batch-Size von 32 betrachtet. Das Training findet auf einer NVIDIA A100 Grafikkarte statt. Eine Übersicht der verwendeten Hyperparameter ist der Tabelle 4.4 zu entnehmen.

4.4.2 Training und Evaluation auf `msmarco-passage`

Die weniger wohlgeformten und kurzen Queries des `msmarco-document`-Datensatzes können trotz der realistischen Umstände, denen sie entspringen, zu Problemen beim Fine-Tuning der Transformer-Modelle führen. Um aus der Betrachtungsweise einer Summarization-Aufgabe gute Zusammenfassungen zu produzieren, müssen die Transformer-Modelle in der Lage sein, nur die wichtigsten Aspekte der Eingaben zu identifizieren. Diese in der Ausgabe wiederzugeben wird umso schwieriger, je weniger Wörter diese enthalten darf. So ist es beispielsweise schwieriger, einen ganzen Text in nur zehn Wörtern zusammenzufassen als in einem kurzen Paragraphen. Eine Analyse der Queries des `msmarco-document`-Datensatzes ergibt, dass jene im Durchschnitt aus drei Wörtern bestehen. Dies ist ein unterschwelliges Signal, welches Transformer-Modelle sich beim Fine-Tuning aneignen können, sodass generierte Queries auf unbekannten Daten ebenfalls nur aus ungefähr drei Wörtern bestehen. Im Gegensatz dazu haben die Queries von `msmarco-passage` eine durchschnittliche Länge von sechs Wörtern. Die Verdopplung dieses Merkmals verringert somit die zuvor genannte potentielle Schwäche und bietet die Möglichkeit einer einfacheren Anpassung der Transformer-Modelle.

Ein weiterer wichtiger Aspekt offenbart sich bei der Betrachtung der zusätzlichen Eingabedaten. Da neben Anchortexten ebenfalls Dokumententexte verwendet werden, spielen deren Charakteristiken gleichermaßen eine wichtige Rolle. Die Passagen von `msmarco-passage` entspringen den Dokumenten aus `msmarco-document`. Es handelt sich bei ihnen aber um gekapselte und abge-

schlossene Abschnitte. Dabei umfassen Passagen meist einen Kernaspekt des Ausgangsdokumentes und sind in ihrer Länge viel kürzer als Dokumententexte. Die in Abschnitt 4.2 angesprochenen Limitierungen der Eingabelängen von BERT, BART und T5 wurden zuvor durch die Konkatenation von Anchortext, Titel und Dokumententext bei weitem überschritten und an der maximal erlaubten Eingabelänge abgeschnitten. Dies hat den Nachteil, dass potentiell wichtiger Dokumententext nicht Teil der Eingabemenge wird. Hinzu kommt eine Inkonsistenz der Eingabekomposition. Der betrachtete Anteil des Dokumentes korreliert mit der Länge der Anchortexte. Je kürzer diese sind, umso mehr Dokumententext schafft es in die Eingabe und umgekehrt. Im Durchschnitt haben die Konkatenationen auf `msmarco-document` eine Länge von 1.861 Wörtern und übersteigen somit selbst das größte Token-Limit von BART mit 1.024 Token (vgl. Tabelle 4.2). Im Gegensatz dazu enthalten die Konkatenationen auf `msmarco-passage` durchschnittlich 306 Wörter und können somit selbst von den Modellen mit den geringsten Limits von 512 Token (vgl. Tabelle 4.2) verarbeitet werden. Diese Eigenschaft und die Tatsache, dass Passagen in sich abgeschlossen sind, können somit ein Vorteil beim Fine-Tuning der Modelle sein und motiviert dadurch die Betrachtungen dieses Experimentes.

Im Gegensatz zu `msmarco-document` gibt es bei `msmarco-passage` keine direkte Zuordnung von Anchortexten zu Passagen. Da aber Passagen einem Dokument entspringen und der ursprüngliche `msmarco-qna` Datensatz ein Mapping beider Elemente enthält, kann die Zuordnung der Anchortexte darüber realisiert werden. Von den ungefähr 500.000 Query-Passage-Paaren aus `msmarco-passage/train/judged` (vgl. Tabelle 4.1) können somit ca. 300.000 Anchortexte zugeordnet werden. Es ergibt sich daher ein deutlicher Unterschied in der verfügbaren Menge von Trainingsdaten im Vergleich zum vorherigen Experiment. Um die Datenmenge so groß wie möglich zu halten, wird ein Trainingsdatensatz aus 300.000 Einträgen erzeugt und zur Evaluation der `msmarco-passage/dev/judged`-Datensatz verwendet. Um zusätzlich eine Vergleichbarkeit mit den Transformer-Modellen aus `msmarco-document` zu schaffen, welche auf 500.000 Daten trainiert wurden, wird ein Trainingsdatensatz aus 500.000 Einträgen erzeugt. Da nicht alle Einträge über Anchortexte verfügen, wird im Training bei fehlenden Anchortexten nur der Passagentext betrachtet. Die Vorverarbeitung geschieht bezüglich der Anchortexte äquivalent zum vorherigen Experiment. Eine Filterung der Queries wird durch die wohlgeformte Natur dieser und die mangelnde Datenmenge nicht durchgeführt. Ebenfalls wird das Training von BERT, BART und T5 über die Transformers[8]-Bibliothek durchgeführt. Die Betrachtung der `base`- und `large`-Varianten so-

[8] `https://huggingface.co/docs/transformers`

wie die verschiedenen Eingabemethoden sind äquivalent zum vorherigen Experiment. In Summe werden somit 62 Modelle trainiert. Bezüglich der Evaluation werden dieselben Aspekte betrachtet wie zuvor.

Die Evaluation gleicht in ihren Schritten dem vorherigen Experiment. Die syntaktische und semantische Analyse erfolgt auf 4.096 Einträgen (55,1%) des `msmarco-passage/dev/small`-Datensatzes, für die Anchortexte zugeordnet werden konnten. Da die Generierung für 8,84 Millionen Passagen die zeitlichen Limitationen überschreiten würde, werden für die Retrieval-Szenarien 5.554 Relationen (60,0%) des `msmarco-passage/trec-dl-2019/judged`-Datensatzes zur Erzeugung künstlicher Queries verwendet. Die Wahl der Hyperparameter weicht leicht zu den Modellen, welche auf `msmarco-document` trainiert wurden, ab. Gradient-Accumulation wurde nicht benutzt, da dies die Verwendung das Fine-Tuning stark verlangsamt. Die `large`-Varianten werden mit einer Batch-Size von 8 und die `base`-Varianten mit einer Batch-Size von 16 trainiert. Eine Abweichung hierbei ergibt sich aus den Modellgrößen, die die verfügbaren Hardwareressourcen ausreizen. Sonstige Hyperparameter sind äquivalent und können der Tabelle 4.4 entnommen werden. Das Training findet auf einer NVIDIA A100 Grafikkarte statt.

4.4.3 Evaluation auf `clueweb09`

Als nicht verwendete Trainingsgrundlage bietet der `clueweb09`-Datensatz eine weitere Evaluationsmöglichkeit aller erstellten Transformer-Modelle. Durch das Auftauchen von sowohl sehr kurzen als auch wohlgeformten Queries werden charakteristische Merkmale des Fine-Tunings aller bisherigen Transformer-Modelle abgedeckt. Aufgrund der geringen Menge von Queries für `clueweb09`, werden syntaktische und semantische Ähnlichkeiten nicht betrachtet. Ergänzend zu den vorangegangenen Experimenten werden weitere Retrieval-Szenarien durchgeführt.

Für die Betrachtung der Retrieval-Szenarien wird der TREC 2009 Web Track Teildatensatz (vgl. Abschnitt 4.1) verwendet. Da die Traversierung des 30 TB großen `clueweb09`-Datensatzes mit seinen 1,04 Milliarden Dokumenten die zeitlichen Limitierungen übersteigt, wird eine Teilmenge von 48.982 Dokumenten entnommen. Ein Vergleich der Effektivität durch vollständige Indexierung verschiedener Dokumentenvarianten war aufgrund des damit verbundenen hohen Rechenaufwands nicht möglich. Unter Betrachtung der Dokumententeilmenge können jedoch 1.891 Dokumente (8,0%) mit Anchortexten angereichert werden. Retrieval wird nur auf dieser Relationsteilmenge durchgeführt. Zur Generierung der künstlichen Queries werden die besten Transformer-

Modelle der vorangegangenen Experimente untersucht.

4.5 Experimentelle Ergebnisse

Ergebnisse auf `msmarco-document` Eine Ergebnisauswahl des ersten Experimentes in Hinblick auf die Evaluierung syntaktischer und semantischer Ähnlichkeiten von generierten und echten Queries ist der Tabelle 4.5 zu entnehmen. Eine vollständige Ergebnisübersicht aller Transformer-Modelle enthält Anhang A. Tabelle 4.5 beschränkt sich auf die Betrachtung von `base`-Varianten, da eine Verbesserung der Metriken durch `large`-Varianten nicht beobachtet werden konnte. Für die Betrachtungen von Top-k-Sampling erzeugt jedes Modell 80 Queries pro Dateneintrag für $k = 10$. Die Berechnungen der Metriken verwenden die erste dieser 80 Queries als Kandidatentext (vgl. Abschnitt 4.3). Die Durchschnittsbildung über alle Queries führt zu keiner deutlichen Verbesserung der Metrikwerte, aber zu einem zeitlich erheblichen Mehraufwand.

Das effektivste untersuchte Modell ist T5-`base` mit Training auf 1.000.000 Query-Dokument-Paaren und einer Query-Generierung durch Greedy-Decoding. Zusätzlich verwendet das T5-`base`-Modell vorverarbeitete Anchortexte und konkateniert diese mit den Dokumententiteln und -texten. Es erzielt einen ROUGE-L F1-Score von 0,49, einen BLEU-Score von 0,21 und einen F1-BERTScore von 0,51. Das T5-`base`-Modell sorgt somit für eine annähernde Verdopplung von ROUGE-L und BERTScore und eine Vervierfachung von BLEU im Vergleich zu Doc2Query (ROUGE-L F1: 0,23; BLEU: 0,05; BERTScore F1: 0,30). Doc2Query verwendet Top-k-Sampling für die Generierung von Queries. Daher wird ein T5-`base`-Modell betrachtet, welches ebenfalls Top-k-Sampling einsetzt und in der sonstigen Konfiguration dem vorherigen T5-`base`-Modell entspricht. T5-`base` mit Top-k-Sampling erreicht einen ROUGE-L F1-Score von 0,39, einen BLEU-Score von 0,21 und einen F1-BERTScore von 0,43. Es übertrifft damit ebenfalls die Wertungen von Doc2Query, aber in einem geringeren Maße als das T5-`base`-Modell mit Greedy-Decoding. Dies lässt sich auf die Funktionsunterschiede von Greedy-Decoding und Top-k-Sampling zurückführen. Top-k-Sampling setzt die Wahrscheinlichkeit ab dem Token $k + 1$ auf Null und erzeugt Permutationen der besten k Token. Greedy-Decoding wählt bei der Generierung von Sequenzen immer das wahrscheinlichste Token als nächstes. Top-k-Sampling ist kaum in der Lage, an die wahrscheinlichste Query aus Greedy-Decoding heranzureichen.

Tabelle 4.5: Übersicht einer auf `msmarco-document` trainierten Modellauswahl mit ROUGE-L, BLEU und BERTScores der generierten Queries und Angaben zu Precision (P), Recall (R) und F1-Score (F1). Gekennzeichnet sind, in Bezug auf Modelltraining und -eingabe, un- oder vorverarbeitete Anchortexte (U/V), die Kombination der Anchortexte mit Dokumententitel und -text (ATD) sowie Greedy-Decoding (G) und Top-k-Sampling (T).

Ansatz		**ROUGE-L**			**BLEU**	**BERTScore**		
		P	R	F1		P	R	F1
Baselines								
Dokumente		0,01	**0,79**	0,01	0,00	0,00	0,01	0,00
Titel		0,37	0,41	0,35	0,01	0,27	0,20	0,23
Anchortexte (U)		0,06	0,58	0,08	0,00	0,00	0,07	0,00
Anchortexte (V)		0,14	0,57	0,17	0,00	0,00	0,17	0,01
D2Q		0,23	0,40	0,28	0,05	0,27	0,34	0,30
BERT-base								
10K	(U, ATD, G)	0,26	0,24	0,23	0,03	0,27	0,24	0,25
500K	(U, ATD, G)	0,51	0,42	0,44	0,14	0,48	0,44	0,46
1000K	(V, ATD, G)	0,49	0,44	0,44	0,15	0,48	0,44	0,46
1000K	(U, ATD, G)	0,50	0,41	0,43	0,14	0,48	0,42	0,45
1000K	(V, ATD, T)	0,39	0,39	0,36	0,09	0,40	0,39	0,39
BART-base								
10K	(V, ATD, G)	0,44	0,42	0,41	0,15	0,45	0,43	0,44
500K	(U, ATD, G)	0,52	0,46	0,47	0,19	0,51	0,47	0,49
1000K	(V, ATD, G)	0,51	0,45	0,46	0,18	0,51	0,46	0,49
1000K	(U, ATD, G)	0,53	0,44	0,46	0,18	0,51	0,46	0,48
1000K	(V, ATD, T)	0,51	0,45	0,46	0,18	0,51	0,46	0,48
T5-base								
10K	(V, ATD, G)	0,48	0,44	0,43	0,17	0,47	0,45	0,46
500K	(V, ATD, G)	0,54	0,46	0,48	0,20	0,52	0,48	0,50
1000K	(V, ATD, G)	**0,55**	0,48	**0,49**	**0,21**	**0,53**	**0,49**	**0,51**
1000K	(U, ATD, G)	0,54	0,47	0,48	**0,21**	0,52	0,48	0,50
1000K	(V, ATD, T)	0,43	0,41	0,39	0,13	0,43	0,42	0,43

Im Gegensatz zu Greedy-Decoding kann Top-k-Sampling somit aber mehr als eine Query generieren. Beispiele von Generierungen verschiedener Modelle sind in Tabelle 4.6 zu sehen.

Die Reduzierung der Trainingsdaten des T5-`base`-Modells um die Hälfte auf 500.000 Query-Dokument-Paare führt zu einer Verschlechterung der Metriken um maximal 2 Prozentpunkte. Die Reduzierung der Trainingsdaten auf 10.000 Query-Dokument-Paare (1,0%) führt lediglich zu einer minimalen Verschlechterung der Metriken um maximal 7 Prozentpunkte (ROUGE-L F1: 0,43; BLEU: 0,17; BERTScore F1: 0,46). Für die Generierung einer Query mittels Greedy-Decoding ist somit das Modelltraining auf selbst wenigen Daten genügend, um ähnlich gute Queries zu generieren, wie bei der Verwendung der hundertfachen Menge von Trainingsdaten. Die Betrachtung der Tabelle 4.6 zeigt aber, dass beispielsweise das T5-`base`-Modell mit 10.000 Query-Dokument-Paaren nur leichte Variationen mittels Top-k-Sampling generiert im Vergleich zum Modell mit 1.000.000 Query-Dokument-Paaren.

Im Hinblick auf andere Modelle zeigt sich, dass BERT-`base` ähnliche Restriktionen wie T5-`base` besitzt. Top-k-Sampling (ROUGE-L F1: 0,36; BLEU: 0,09; BERTScore F1: 0,39) führt zu einer Verschlechterung der Metriken im Vergleich zu Greedy-Decoding (ROUGE-L F1: 0,44; BLEU: 0,15; BERTScore F1: 0,46). Die Reduktion der Trainingsdaten um die Hälfte hat leicht schlechtere BLEU-Scores zur Folge. Auffällig sind hingegen starke Verschlechterungen durch die Reduktion der Trainingsdaten auf 10.000 Query-Dokument-Paare (ROUGE-L F1: 0,23; BLEU: 0,03; BERTScore F1: 0,25). Dies lässt sich vermutlich auf die geringere Modellkomplexität und das weniger ausführliche Pre-Training von BERT im Vergleich zu T5 erklären. Im Gegensatz zu T5-`base` führen bei BERT-`base`-Modellen hauptsächlich unverarbeitete Anchortexte zu besseren Ergebnissen. Es fällt auf, dass die besten Modellvarianten erst mit zunehmender Modellkomplexität einen Vorteil aus der Vorverarbeitung der Anchortexte ziehen (vgl. Tabelle 4.5). Die Unterschiede zwischen der Verwendung von vor- oder unverarbeiteten Anchortexten führt aber bei allen Modellen nur zu minimalen Abweichungen. Selbiges gilt für die Kombination von Anchor- mit Dokumententexten (vgl. Anhang A). Eine mögliche Erklärung hierfür liefert der Self-Attention-Mechanismus. Dieser identifiziert die wichtigsten Elemente einer Eingabe durch die Bezugnahme jedes Wortes mit jedem anderen Wort. Somit scheinen die Modelle in der Lage zu sein, über Redundanzen, wie sie in Anchortexten auftauchen, selbstständig hinwegzusehen.

Die BART-`base`-Modelle zeichnen sich insbesondere durch eine geringe Abweichung der Ergebniswerte bezüglich der Verwendung unterschiedlich vie-

Tabelle 4.6: Beispiele generierter Queries für das Dokument D3314578 und die echte Query: `docker stop all containers`.

Dokumententext: Much of the focus of Docker is on the process of packaging and running your application in an isolated container.There are countless tutorials that explain how to run your application in a Docker container, but very few that discuss how properly stop your containerized app.That may seem like a silly topic – who cares how you stop a container, right? [. . .]

Modell		**Generierte Queries**
D2Q	(1)	`what happens if you shut down a docker container`
	(2)	`how to stop a docker container`
	(3)	`how do you stop a Docker application`
		...
T5-base - 1000K (V, ATD, G)	(1)	`docker stop`
T5-base - 1000K (V, ATD, T)	(1)	`how to stop a docker container`
	(2)	`docker stop command`
	(3)	`docker terminal stop`
		...
T5-base - 10K (V, ATD, G)	(1)	`how to stop a docker container`
T5-base - 10K (V, ATD, T)	(1)	`how to stop a docker`
	(2)	`how to stop docker`
	(3)	`how to shutdown docker`
		...
BERT-base - 1000K (V, ATD, G)	(1)	`docker stop docker`
	(2)	`docker port`
	(3)	`how to stop docker`
		...
BART-base - 1000K (V, ATD, T)	(1)	`stop Docker`
	(2)	`stop Docker`
	(3)	`stop Docker`
		...

ler Query-Dokument-Paare aus. Top-k-Sampling (ROUGE-L F1: 0,46; BLEU: 0,18; BERTScore F1: 0,48) führt zu annähernd keiner Verschlechterung im Vergleich zu Greedy-Decoding (ROUGE-L F1: 0,46; BLEU: 0,18; BERTScore F1: 0,49). Die Reduktion auf 10.000 Query-Dokument-Paare folgt diesem

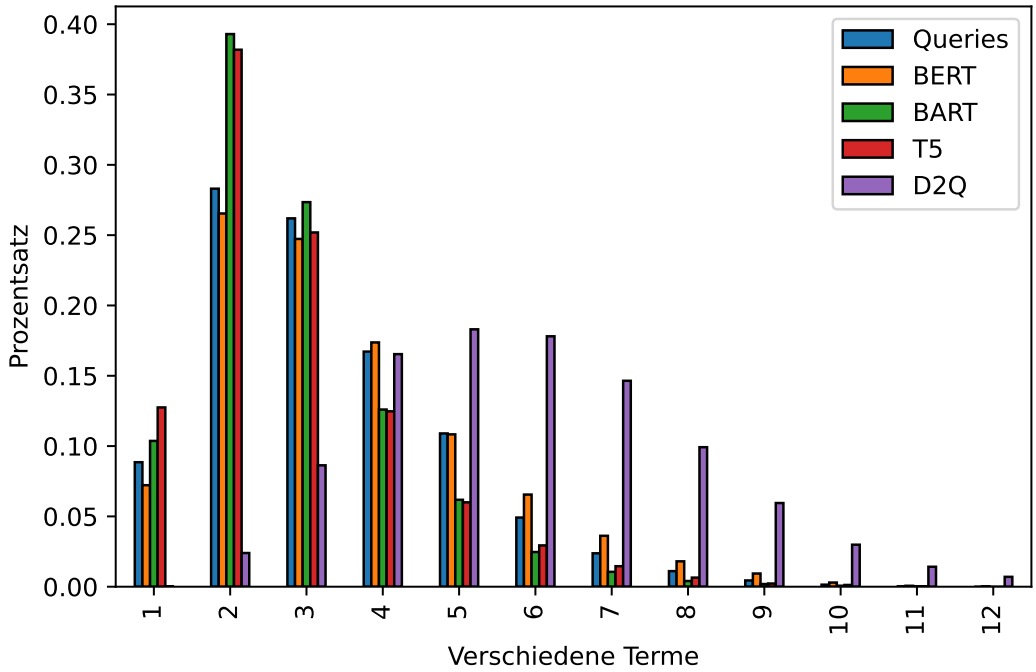

Abbildung 4.4: Verteilungen der Anzahl von verschiedenen Wörtern für echte und generierte Queries. Die Modellauswahl umfasst die besten Modelle aus Tabelle 4.5 mit Greedy-Decoding.

Beispiel (ROUGE-L F1: 0,41; BLEU: 0,15; BERTScore F1: 0,44). Es fällt auf, dass das BART-`base`-Modell mit 500.000 Query-Dokument-Paaren minimal bessere Metrikwerte erzielt als das BART-`base`-Modell mit 1.000.000 Query-Dokument-Paaren. Die Ursache dafür wird in der Verwendung von Early-Stopping während des Fine-Tunings angenommen, da Modelle dadurch nicht über eine einheitliche Menge an Batches trainiert wurden. Die Beispiele aus Tabelle 4.6 zeigen, dass BART-`base` bei der Generierung mittels Top-k-Sampling nahezu immer dieselbe Query wiederholt. Dies lässt sich vermutlich durch die Fokussierung auf Summarization von BART im Pre-Training erklären. Dies führt zu einer stärkeren Verschiebung der Token-Wahrscheinlichkeiten, sodass viele Token schon vor Top-k-Sampling eine Wahrscheinlichkeit von 0,0% besitzen. Somit bleiben nicht genügend Token für Permutationen übrig und BART generiert mit hoher Wahrscheinlichkeit dieselben Queries.

Alle Modelle aus Tabelle 4.5 generieren die besten Queries mit Hilfe von Anchortexten und der zusätzlichen Verwendung des Dokumententextes und -titel. Modelle, die ausschließlich auf Anchortexten trainiert wurden, erzielen in der Regel geringere Metrikwerte (vgl. Anhang A). Diese Abweichun-

gen sind aber oftmals minimal. Die Vermutung liegt an einer unterschiedlich hohen Informationsdichte von Dokumenten- und Anchortexten. Bei Betrachtung der Baselines fällt auf, dass Anchortexte geringere ROUGE-L Recall-Scores besitzen als Dokumententexte. Es scheinen daher mehr Informationen für die Generierung von Queries in Web-Dokumenten als Anchortexten enthalten zu sein. Nichtsdestotrotz haben Dokumenten- und Anchortexte die höchsten ROUGE-L Recall-Scores und bestätigen damit die Annahme, dass neben Web-Dokumenten auch Anchortexte eine geeignete Grundlage zur Generierung von Queries bilden. Es fällt ebenfalls auf, dass Dokumententitel eine Ähnlichkeit zu Queries aufweisen. In Bezug auf ROUGE-L ist diese Ähnlichkeit sogar höher als bei Doc2Query-Queries. Erklärt wird dies durch einen ähnlichen Gedankenprozess bei der Erstellung von Web-Dokumenttiteln und Queries, den Jin et al. [JHZ02] bereits 2002 als möglichen Grund für gefundene Ähnlichkeiten anführten. Doc2Query neigt dazu, längere und wohlgeformtere Queries zu generieren (vgl. Tabelle 4.6). Eine Eigenschaft, die es sich vermutlich im Training auf `msmarco-passage` und TREC-CAR [DVRC17] angeeignet hat. TREC-CAR umfasst Wikipedia-Artikel, welche vermutlich am stärksten dazu beigetragen haben. Da Complex-Answer-Retrieval der Fokus für diesen Datensatz ist, erlernen somit darauf trainierte Modelle vermutlich ebenfalls komplexe Satzstrukturen.

Eiron und McCurley [EM03] sowie Fröbe et al. [FGP+22] folgend, werden ebenfalls statistische Merkmale wie Wortverteilung und -häufigkeit untersucht. Die Abbildung 4.4 zeigt, dass Doc2Query meist mehr Wörter generiert als echte Queries im Durchschnitt enthalten. Die auf `msmarco-document` trainierten Modelle ähneln der Wortverteilung echter Queries stärker und BERT spezifisch am stärksten. Gründe dafür sind abermals auf die, im Fine-Tuning verwendeten, unterschiedlichen Datensätze zurückzuführen. So zeigt eine Analyse, dass Queries aus `msmarco-passage` und TREC-CAR im Schnitt sechs Wörter beinhalten. Queries aus `msmarco-document` enthalten durchschnittlich drei Wörter. Der `msmarco-document`-Datensatz entspricht eher den Wortverteilungen von Queries in diversen Forschungen als der `msmarco-passage`-Datensatz [SHMM99, JP01].

Um die Ergebniswerte besser einordnen zu können, werden BLEU- und ROUGE-L-Scores mit den Leistungen anderer Modelle auf dem WMT2014-English-German-[9] und CNN/Daily-Mail-Benchmark[10] [BBF+14, NZdS+16] verglichen. Beide dienen zur Evaluation für Translation- beziehungsweise Summarization-Aufgaben. Es ist im Hinterkopf zu behalten, dass die Datensätze

[9] `https://paperswithcode.com/sota/machine-translation-on-wmt2014-english-german`
[10] `https://paperswithcode.com/sota/document-summarization-on-cnn-daily-mail`

nicht direkt mit `msmarco-document` vergleichbar sind. Vielmehr soll dadurch ein nachvollziehbarer Qualitätseindruck der generierten Queries vermittelt werden, ohne dass lediglich Doc2Query für diese Betrachtung verwendet wird. Das beste Benchmark-Modell für WMT2014 erreicht einen BLEU-Score von 0,34 und ein natives T5-Modell erzielt eine Leistung von 0,27. Das beste T5-`base`-Modell zur Generierung von Queries erreicht einen BLEU-Score von 0,21. Die Unterschiede von bis zu 12,62 Prozentpunkten sind in erster Linie auf die Betrachtung verschiedener Datensätze zurückzuführen. Dennoch zeigt eine Analyse von WMT2014, dass die Ausgangs- und Zielsätze von ungefähr gleicher Länge sind, da eine Übersetzung von Englisch auf Deutsch strukturell nahezu gleichlange Sätze erfordert. In `msmarco-document` hingegen stellt die Betrachtung aus dem Blickwinkel einer Translation-Aufgabe die Umwandlung längerer Ausgangs- hin zu kürzeren Zieltexten eine schwierigere Aufgabe dar. Das Erlernen komplexerer Ersetzungsregeln sowie das Auslassen von Informationen ist somit erforderlich und führt daher zu weniger guten Ergebnissen.

Dass die Bewältigung dieser Aufgabe als Translation-Aufgabe weniger geeignet ist als eine Summarization-Aufgabe, zeigt der Blick auf den CNN-Benchmark. Das beste Benchmark-Modell ist hier in der Lage, einen ROUGE-L F1-Score von 0,44 zu erzielen. Ein T5-Modell ohne Fine-Tuning erreicht einen Score von 0,19 und ein BART-Modell ohne Fine-Tuning 0,41. Das beste T5-`base`-Modell zur Query-Generierung ist um 5 Prozentpunkte besser als das beste Benchmark-Modell. Erneut sind hauptsächliche Unterschiede auf den Einsatz verschiedener Datensätze zurückzuführen.

Neben der Betrachtung semantischer und syntaktischer Äquivalenz werden die inhaltlichen Beziehungen der generierten Queries zu den Ausgangsdokumenten mittels Retrieval-Szenarien ermittelt. Die Ergebnisse diesbezüglich finden sich in Tabelle 4.7. Betrachtet werden alle `base`-Modelle mit 1.000.000 Query-Dokument-Paaren, die in ihrem Training zusätzlich mit Dokumententexten und -titeln trainiert wurden. Unterschieden wird zwischen Modellen, die vor- und unverarbeitete Anchortexte verwenden. Die Ergebnisse sind außerdem zweigeteilt, da nicht für alle betrachteten Dokumente Anchortexte existieren.

Tabelle 4.7: BM25-Retrieval-Effektivität einer auf `msmarco-document` trainierten Modellauswahl für verschiedene indexierte Daten auf `msmarco-document/trec-dl-2019/judged`. Gekennzeichnet sind vor- oder unverarbeitete Anchortexte (V/U).

Indexierte Daten	Modell	mAP	MRR	nDCG@10	R@1000
Alle Dokumente					
Dokumente	-	0,49	0,86	0,53	0,91
Titel	-	0,20	0,83	0,48	0,40
Anchortexte (U)	-	0,12	0,75	0,40	0,27
Queries	D2Q	0,41	0,87	**0,61**	0,76
Queries	BERT (U)	0,17	0,77	0,44	0,39
Queries	BERT(V)	0,17	0,74	0,43	0,40
Queries	BART (U)	0,11	0,74	0,37	0,26
Queries	BART (V)	0,12	0,73	0,39	0,28
Queries	T5 (U)	0,18	0,77	0,45	0,40
Queries	T5 (V)	0,18	0,73	0,42	0,40
Dokument, Queries	D2Q	**0,53**	**0,89**	**0,61**	**0,92**
Dokument, Queries	BERT (U)	0,51	0,85	0,56	**0,92**
Dokument, Queries	BERT (V)	0,51	0,86	0,57	**0,92**
Dokument, Queries	BART (U)	0,49	0,84	0,52	0,91
Dokument, Queries	BART (V)	0,50	0,87	0,54	0,91
Dokument, Queries	T5 (U)	0,51	0,87	0,56	**0,92**
Dokument, Queries	T5 (V)	0,51	0,88	0,56	**0,92**
Nur Dokumente mit Anchortexten					
Dokumente	-	0,30	0,83	0,46	0,56
Titel	-	0,10	0,73	0,37	0,23
Anchortexte (U)	-	0,12	0,76	0,40	0,27
Queries	D2Q	0,24	0,80	0,52	0,48
Queries	BERT (U)	0,17	0,75	0,44	0,39
Queries	BERT (V)	0,17	0,72	0,43	0,40
Queries	BART (U)	0,11	0,74	0,36	0,26
Queries	BART (V)	0,12	0,72	0,38	0,28
Queries	T5 (U)	0,18	0,77	0,45	0,40
Queries	T5 (V)	0,18	0,72	0,42	0,40
Dokument, Queries	D2Q	**0,31**	0,84	**0,53**	**0,57**
Dokument, Queries	BERT (V)	**0,31**	0,86	0,50	**0,57**
Dokument, Queries	BERT (U)	**0,31**	0,83	0,49	**0,57**
Dokument, Queries	BART (U)	0,30	0,82	0,45	0,56
Dokument, Queries	BART (V)	0,30	0,84	0,47	0,56
Dokument, Queries	T5 (V)	**0,31**	**0,88**	0,49	**0,57**
Dokument, Queries	T5 (U)	**0,31**	0,85	0,50	**0,57**

Als beste Kombination für das Retrieval, in Anbetracht aller Dokumente, ergeben sich die Konkatenation von Dokumententexten und durch Doc2Query generierte Queries. Diese erzielen eine mAP von 0,53, einen MRR von 0,89, einen nDCG@10 von 0,61 und einen R@1000 von 0,92. Die restlichen indexierten Daten, mit Konkatenierung generierter Queries, besitzen geringe Abweichungen von diesen Werten um maximal 7 Prozentpunkte. Das Fine-Tuning der Modelle auf `msmarco-document` erzielt somit nahezu identische inhaltliche Zusammenhänge zwischen Queries und Dokumenten wie Doc2Query. Dies kann durch eine starke Ähnlichkeit zwischen den generierten Queries der Modelle erklärt werden. Wie in Tabelle 4.6 zu erkennen ist, enthalten sowohl Doc2Query-Queries als auch T5-`base`-Queries die wichtigsten Stichwörter wie `stop` und `docker container`. Diese sind ausschlaggebend für den inhaltlichen Bezug zwischen Query und Dokument. Da Stopwörter wie beispielsweise `what` oder `how` im Retrieval nicht beachtet werden, da diese oftmals entfernt werden, erreichen die Modelle eine ähnlliche Effektivität. Alle indexierten Paare von Dokumenten und generierten Queries übertreffen das Retrieval auf Dokumententexten und sind somit in der Lage, wie Doc2Query, dem Vocabulary-Mismatch-Problem entgegenzuwirken. Es kann demnach angenommen werden, dass die generierten Queries eine inhaltliche Beziehung zu den Dokumenten besitzen. Das Retrieval auf der Dokumententeilmenge, bei der alle Dokumente Anchortexte besitzen, weist ebenfalls nur geringe Abweichungen auf. In Bezug auf mAP und R@1000 sind die Werte bei fast allen Kombinationen gleich. Die Gründe dafür überdecken sich mit der zuvor genannten Ähnlichkeit aller Queries sowie dem Besitz von essentiellen Begriffen.

Bei der Betrachtung von Queries als indexierte Daten für alle Dokumente, weisen Doc2Query-Queries die besten Werte auf. Die größten Unterschiede zu den Queries der restlichen Modelle existieren bei Recall-basierten Metriken wie mAP und R@1000. Der Unterschied lässt sich vermutlich auf den Nachteil zurückführen, dass nicht für alle Dokumente Queries durch fehlende Anchortexte generiert wurden. Retrieval auf der Teilmenge von Dokumenten mit Anchortexten bestätigt diese Überlegung, da die Unterschiede dabei nicht so gravierend ausfallen. Allgemein scheint hierbei Doc2Query durch diversere Queries eine bessere Effektivität im Retrieval zu erzielen. Wie in Tabelle 4.6 zu sehen ist, enthält die dritte Doc2Query-Query den Begriff `application`. Dieser Term tritt ebenfalls früh im angegebenen Dokumententext auf. In den ersten Queries der restlichen Modelle ist der Term nicht zu finden. Daher besteht die Annahme, dass Doc2Query-Queries ein umfänglicheres Vokabular aus dem Zieldokument extrahieren können, wodurch eine verbesserte Retrieval-Effektivität resultiert. Sowohl bei der Betrachtung aller Dokumente als auch bei der Betrachtung nur von Dokumenten mit Anchortext, offenbaren sich die BART-`base`-Modelle als

Tabelle 4.8: Übersicht einer, auf `msmarco-passage` trainierten, Modellauswahl mit ROUGE-L, BLEU und BERTScores der generierten Queries und Angaben zu Precision (P), Recall (R) und F1-Score (F1). Gekennzeichnet sind, in Bezug auf Modelltraining und Generierungseingabe, un- oder vorverarbeitete Anchortexte (U/V), die Kombination dieser mit Passagentext (AP) sowie Trainingsepochen (E) und `summarize` als Trainingsbefehl für T5-Modelle (*).

Ansatz	**ROUGE-L**			**BLEU**	**BERTScore**		
	P	R	F1		P	R	F1
Baselines							
Passagen	0,06	**0,50**	0,10	0,00	0,00	0,22	0,04
Titel	0,41	0,27	0,29	0,01	0,29	0,20	0,24
Anchortexte (U)	0,06	0,37	0,07	0,00	0,00	0,06	0,00
Anchortexte (V)	0,14	0,35	0,14	0,00	0,00	0,14	0,00
D2Q	0,40	0,39	0,38	0,11	0,47	0,45	0,46
BERT-base							
300K (V, AP, E:2)	0,36	0,36	0,35	0,09	0,42	0,41	0,41
500K (U, AP, E:2)	0,38	0,37	0,36	0,09	0,44	0,43	0,43
BART-base							
300K (V, AP, E:2)	**0,50**	0,47	**0,47**	**0,17**	**0,56**	**0,52**	**0,54**
500K (V, AP, E:100)	**0,50**	0,46	**0,47**	**0,17**	**0,56**	**0,51**	**0,53**
T5-base							
300K (V, AP, E:2)	0,39	0,38	0,37	0,10	0,45	0,44	0,44
500K (V, AP, E:2)	0,41	0,40	0,39	0,11	0,47	0,46	0,46
Fine-Tuned D2Q							
300K (V, AP, E:2)*	0,40	0,39	0,38	0,11	0,46	0,45	0,45
500K (V, AP, E:2)*	0,40	0,40	0,38	0,11	0,46	0,45	0,45

schwächeres Generierungsmodell. Dies lässt sich darauf zurückführen, dass die BART-Modelle fast ausschließlich gleiche Queries generieren. Eine breite Abdeckung diverser Terme, wodurch ein Vorteil im Retrieval entstehen kann, ist dadurch nicht möglich.

Ergebnisse auf `msmarco-passage` Die Ergebnisse des zweiten Experimentes bezüglich der Metriken ROUGE, BLEU und BERTScore finden sich in Tabelle 4.8. Wie zuvor werden lediglich die `base`-Varianten der Modelle betrachtet, da die `large`-Varianten keine Verbesserungen erzielen. Eine Unterscheidung zwischen Greedy-Decoding und Top-k-Sampling wird nicht durchgeführt. Alle Modelle generieren mittels Top-k-Sampling für $k = 10$. Eine

vollständige Übersicht ist dem Anhang B zu entnehmen.

Im Gegensatz zu den auf `msmarco-document` trainierten Modellen ergibt sich bei `msmarco-passage` BART-`base` als bestes Modell zur Generierung von Queries. Im Vergleich mit Doc2Query (ROUGE-L F1: 0,38; BLEU: 0,11; BERTScore F1: 0,46) erzielt BART-`base` mit 300.000 Query-Passage-Paaren in allen drei Metriken eine Steigerung um mindestens 6 Prozentpunkte (ROUGE-L F1: 0,47; BLEU: 0,17; BERTScore F1: 0,54). Dabei verwendet es vorverarbeitete Anchortexte sowie Passagentitel und -texte als konkatenierte Eingabe zur Erzeugung. Eine Steigerung der Trainingsdaten auf 500.000 Query-Passage-Paaren führt zu keiner Verbesserung und ist vermutlich auf die nicht gestiegene Anzahl von Anchortexten zurückzuführen.

Die BERT-`base`-Modelle erzielen minimal geringere Metrikwerte im Vergleich zu Doc2Query. Die T5-`base`-Modelle haben eine nahezu äquivalente Effektivität wie Doc2Query. Ein möglicher Grund ist, dass Doc2Query und T5-`base` auf derselben Modellarchitektur basieren. Obwohl Doc2Query unter anderem auf dem kompletten `msmarco-passage`-Datensatz trainiert wurde, sind diese Ähnlichkeiten der Metrikwerte überraschend, da sich dieser Datenvorteil in den Generierungen widerspiegeln sollte. Es ist daher anzunehmen, dass für die Replikation der Charakteristiken echter Queries schon wenige Daten ausreichen. Dies deckt sich mit den Ergebnissen aus dem ersten Experiment. Die Beobachtung der Transformer-Modelle, welche auf Queries aus dem ORCAS-Datensatz trainiert wurde, deutet darauf hin, dass die Qualität der verwendeten Daten wichtiger ist als die Datenmenge. Dies spiegelt sich in besseren Metrikwerten von ROUGE, BLEU und BERTScore wider. Ein zusätzliches Fine-Tuning von Doc2Query erzielt ebenfalls keine Verbesserungen. Die zusätzlich hinzugefügten Anchortexte scheinen daher keinen großen Einfluss auf die Anpassung von Doc2Query zu nehmen. Sie widersprechen vermutlich zu stark den wohlgeformten Daten aus dem Pre-Training von Doc2Query. Beispielhafte Generierungen sind der Tabelle 4.10 zu entnehmen. Anhand dieser sind die Ähnlichkeiten in der Generierung zwischen Doc2Query und T5-`base` ebenfalls zu erkennen.

Tabelle 4.9: Vergleich ausgewählter indexierter Daten für BM25-Retrieval auf `msmarco-passage/trec-dl-2019/judged` unter Verwendung des besten 300K BART- und T5-Modells (BART/T5) aus Tabelle 4.8. Gekennzeichnet sind Doc2Query−− [GMM23] (−−), die Query-Generierung auf Anchortexten (*AT*) und das Modelltraining auf unverarbeiteten Anchortexten (U).

Indexierte Daten	**Modell**	**mAP**	**MRR**	**nDCG@10**	**R@1000**
		Alle Passagen			
Passage	-	0,49	0,77	0,45	0,84
Titel	-	0,30	0,83	0,47	0,48
Anchortexte (U)	-	0,22	0,77	0,43	0,38
Queries	D2Q	0,55	0,90	0,61	0,84
Queries	BART	0,40	0,86	0,50	0,65
Queries	T5	0,56	0,90	0,59	0,84
Passage, Queries	D2Q	**0,60**	0,91	0,63	**0,87**
Passage, Queries$_{(--)}$	D2Q	0,58	**0,95**	**0,66**	0,86
Passage, Queries	T5	**0,60**	0,90	0,61	**0,87**
Passage, Queries$_{(--)}$	T5	0,57	0,90	0,65	0,86
		Passagen mit Anchortexten			
Passage	-	0,30	0,70	0,39	0,53
Titel	-	0,17	0,80	0,38	0,28
Anchortexte	-	0,21	0,76	0,42	0,38
Queries	D2Q	0,34	0,84	0,53	0,54
Queries	BART	0,24	0,80	0,44	0,41
Queries	T5	0,34	**0,88**	0,53	0,54
Queries$_{(AT)}$	D2Q	0,22	0,75	0,38	0,40
Queries$_{(AT)}$	T5-base 300K (U)	0,25	0,82	0,46	0,47
Passage, Queries	D2Q	0,37	0,83	0,55	**0,56**
Passage, Queries	T5	**0,38**	0,85	0,54	**0,56**
Passage, Queries, Titel	D2Q, BART, T5	**0,38**	0,84	**0,56**	**0,56**

Wie bei `msmarco-document` generiert BART-`base` auf `msmarco-passage` nahezu identische Queries aufgrund seiner Summarization-Aufgabe im Pre-Training. BERT-`base` neigt dazu Queries zu generieren, welche der Struktur von echten Queries folgen. Diese Queries ergeben sich bei genauerer Betrachtung aber als sinnfreie Sätze. Beispielsweise generiert BERT-`base` für ein Dokument über den Bürgerkrieg der USA die Query: `what were the states between each other and two`. BART- und T5-`base` hingegen generieren nahezu äquivalente Queries zu: `what states were part of the union`. Ein weniger komplexes Pre-Training führt zu einem geringerem Textverständnis von BERT im Vergleich zu BART und T5. Dieses Defizit wird als hauptsächlicher Grund für die Generierung von teilweise sinnfreien Sätzen angenommen.

Die Recall-Werte von ROUGE-L der Passagen zeigen, dass diese die höchste Informationsdichte für die Generierung der Queries enthalten. Im Vergleich zu den Dokumenten von `msmarco-document` enthalten sie aber insgesamt weniger Informationen. Da die Passagen den Dokumenten entspringen und sich nur mit Teilaspekten beschäftigen, ist dies nachvollziehbar. Weniger verständlich sind die gesunkenen Recall-Werte von Anchortexten im Vergleich zu denen von `msmarco-document`. Eine mögliche Erklärung ist der kleinere Testdatensatz, welcher im Durchschnitt weniger Anchortexte pro Eintrag zur Verfügung stellt. Allgemein sind die Metriken, insbesondere BLEU, der `msmarco-passage`-Modelle, im Vergleich zu `msmarco-document`-Modellen gesunken. Eine mutmaßliche Erklärung sind die gesunkenen Informationsdichten der Anchortexte und Passagen. Diese sorgt für eine mangelhafte Anpassung im Training und somit zur Generierung weniger äquivalenter Queries.

Die Betrachtung der Retrieval-Szenarien der `msmarco-passage`-Modelle ist der Tabelle 4.9 zu entnehmen. Wie zuvor werden die besten `base`-Modelle aus Tabelle 4.8 betrachtet. Dabei wird sich auf die Modelle mit 300.000 Query-Passage-Paaren beschränkt. Zusätzlich wird das auf Doc2Query aufbauende Doc2Query−− [GMM23] und der Modelleinsatz in datenarmen Szenarien unter der alleinigen Verwendung von Anchortexten betrachtet. Die Experimente sind erneut in zwei Teilmengen aufgeteilt. Eine, bei der alle Passagen Anchortexte besitzen, sowie eine mit allen Passagen.

T5-`base` und Doc2Query erzielen, unter Verwendung der Konkatenation von Passagentexten und generierten Queries, die besten Werte für die Recall-basierten Metriken mAP und R@1000. Für die Precision-basierten Metriken MRR und nDCG@10 hat Doc2Query die besten Ergebnisse. Insgesamt sind die Abweichungen der Modelle minimal um höchstens 2 Prozentpunkte. Eine mögliche Erklärung ist wie zuvor dieselbe Modellarchitektur durch T5. Überra-

Tabelle 4.10: Beispiele von generierten Queries für die Passage 7505112 und die echte Query: `what is a qph file`.

Passagentext: The QPH file type is primarily associated with 'Quicken' by Intuit Inc.. The Quicken Price History (QPH) file contains a record of all of the prices that have been entered or downloaded for all of your securities. Usually named QDATA.QPH. Primary association: Quicken.

Modell		**Generierte Queries**
D2Q	(1)	`what is quicken.qph`
	(2)	`what is a quicken price record format`
	(3)	`quicken price history`
		...
BERT-`base` - 300K (V, ATP, T)	(1)	`what is qdata`
		`what is a qph file?`
		`what is a qph file c # d. file`
		...
BART-`base` - 300K (V, ATP, T)	(1)	`what is a qph file`
		`what is a qph file`
		`what is a qph file`
		...
T5-`base` - 300K (V, ATP, T)	(1)	`what type of file is quicken`
		`types of quicken pricing history files`
		`what is a qph file in stocks`
		...

schend sind die fast identische Effektivität trotz eines geringeren Datenumfangs im Fine-Tuning. Der Einsatz von Doc2Query−− auf beiden Modellen, führt bei Doc2Query zu einer Steigerung der nDCG@10- und MRR-Werte um 3 beziehungsweise 4 Prozentpunkte. Bei T5-`base` steigert sich nur der nDCG@10 um 4 Prozentpunkte. Die Recall-basierten Metriken sinken bei beiden Modellen. Zurückzuführen ist dies auf die Funktionsweise von Doc2Query−−. Mittels einer Bewertungsfunktion wird die Relevanz der generierten Queries zu den Ausgangsdokumenten ermittelt. Wenig relevante Dokumente werden entfernt, weshalb, durch weniger Text, der Recall sinkt. Die Precision steigt hingegen, da die Menge der generierten Queries eindeutiger auf relevante Dokumente zeigt.

Queries als indexierte Daten erzielen ähnliche Ergebnisse wie Retrieval auf Konkatenationen von Passagen und Queries. Minimale Verschlechterungen zeichnen sich in allen Metriken ab. Dies zeigt, dass die Modelle scheinbar

Tabelle 4.11: Beispiele von generierten Queries für das Dokument `clueweb09-en0030-85-25684` und die echte Query: `euclid`.

Dokumententext: History of Horticulture - Euclid 323-285 B.C. Euclid 323-285 B.C. Euclid was an eminent mathematician who founded a school of Mathematics at Alexandria during the reign of Ptolemy I. His greatest work was the Elements of Geometry which remained in use practically unchanged for 2,000 years.[. . .]

Modell		**Generierte Queries**
D2Q	(1)	`what was euclid's school of mathematics like`
	(2)	`what year did euclid live in the nys`
	(3)	`when was euclid's period`
		...
BERT-`base` - 1000K (V, AD, T)	(1)	`when was euclid discovered`
		`history of euclid`
		`euclid`
		...
BART-`base` - 1000K (V, AD, T)	(1)	`who was euclid`
		`euclid history`
		`who was euclid`
		...
T5-`base` - 1000K (V, AD, T)	(1)	`who invented horticulture`
		`euclid math`
		`euclid facts`
		...

in der Lage sind, alle wichtigen Informationen der Passagen in Queries umzuformulieren, wodurch die Konkatenation mit den Passagen selbst zu keiner groß gesteigerten Retrieval-Effektivität führt. BART-`base` zeigt die größten Verschlechterungen in allen Metriken gegenüber Doc2Query und T5-`base` für Retrieval auf generierten Queries. Wie zuvor liegt dies vermutlich am Fokus auf Summarization, wodurch weniger abwechslungsreiche Queries mit Top-k-Sampling generiert werden.

Die Betrachtung der Passagenmenge, bei der alle Passagen Anchortexte besitzen, zeigt ebenfalls kaum Unterschiede in der Retrieval-Effektivität zwischen Doc2Query und T5-`base`. Die Bildung der Vereinigung aller generierten Queries von Doc2Query, BART und T5 mit anschließender Entfernung von Duplikaten weist ebenfalls keine Verbesserungen auf. Die Analyse der Queries zeigt, dass Doc2Query, T5-`base` und BART-`base` lediglich leicht unterschiedliche Queries generieren. So übertrifft in den Betrachtungen zuvor, kein Modell

Tabelle 4.12: Vergleich ausgewählter indexierter Daten für BM25-Retrieval auf `clueweb09/trec-web-2009`. Betrachtet werden die besten 1000K-`base`-Modelle aus Tabelle 4.5. Gekennzeichnet sind vor- und unverarbeitete Anchortexte (U/V) sowie die Hinzunahme von Dokumententexten (AD).

Indexierte Daten	Modell	mAP	MRR	P@5	nDCG@10	R@1000
Dokumente	-	0,04	0,36	0,26	0,18	**0,15**
Anchortexte (U)		0,03	0,52	0,27	0,18	0,11
Queries	D2Q	0,02	0,21	0,12	0,09	0,13
Queries	BERT (U, AD)	**0,09**	**0,69**	**0,50**	**0,34**	0,13
Queries	BERT (V, AD)	**0,09**	0,57	0,47	0,33	0,14
Queries	BART (U, AD)	0,08	0,65	0,45	0,33	0,12
Queries	BART (V, AD)	0,08	0,61	0,49	0,33	0,12
Queries	T5 (U, AD)	**0,09**	0,62	0,46	0,33	0,14
Queries	T5 (V, AD)	**0,09**	0,62	0,46	0,32	0,14
$\text{Queries}_{(AT)}$	D2Q	0,02	0,30	0,17	0,11	0,10
$\text{Queries}_{(AT)}$	BERT (U)	0,07	0,58	0,45	0,32	0,12
$\text{Queries}_{(AT)}$	BERT (V)	0,08	0,63	0,49	**0,34**	0,12
$\text{Queries}_{(AT)}$	BART (U)	0,07	0,61	0,46	0,31	0,11
$\text{Queries}_{(AT)}$	BART (V)	0,07	0,63	0,47	0,31	0,11
$\text{Queries}_{(AT)}$	T5 (U)	0,08	0,63	0,46	0,33	0,13
$\text{Queries}_{(AT)}$	T5 (V)	0,08	0,62	0,46	0,33	0,13

die von BART-`base` generierten Queries in Bezug auf ROUGE, BLEU und BERTScore. Die Vereinigung der generierten Queries aller Modelle und die anschließende Entfernung von Duplikaten erreicht keine Steigerung der Metrikwerte. Hinzu kommt die Tatsache, dass die Generierung von n Queries durch drei Modelle einen Anstieg der Verarbeitungsdauer bedeutet.

Eine weitere Untersuchung bezog sich auf datenarme Szenarien. Solche, bei denen kaum Passagentext vorhanden ist, um daraus Queries durch beispielsweise Doc2Query zu generieren. Vergleichbar sind solche Szenarien mit Entry-Page-Aufgaben. Solche Entry-Pages können wenig Textinhalt besitzen, wodurch Volltext-Retrieval oder die Generierung von Queries nicht möglich sind. Durch Verlinkungen zu solchen Entry-Pages sind allerdings Anchortexte verfügbar, was die genannte Untersuchung motiviert. Ein T5-`base` Modell, welches nur auf Anchortexten trainiert wurde und nur aus Anchortexten Queries generiert, erzielt bessere Metrikwerte als Doc2Query mit Steigerungen um 2,9 bis 6,8 Prozentpunkte. Eine abschließende Betrachtung der Baselines im Vergleich zum vorherigen Experiment zeigt, dass Anchortexte in der Lage sind, bessere Ergebnisse für MRR und nDCG@10 zu erzielen.

Ergebnisse auf `clueweb09` Die Ergebnisse des letzten Experimentes sind der Tabelle 4.12 zu entnehmen. Diese beschränkt sich auf `msmarco-document`-Modelle, da `msmarco-passage`-Modelle keine Verbesserungen erzielen. Ein vollständiger Überblick kann dem Anhang C entnommen werden, welcher zusätzliche Eingabemethoden und ein T5-Modell des `msmarco-passage`-Datensatzes betrachtet. Alle `msmarco-document`-Modelle sind in der `base`-Variante und wurden auf 1.000.000 Query-Dokument-Paaren trainiert. Unterschieden wird zwischen der Verwendung von Anchortexten mit beziehungsweise ohne Dokumententext und vor- und unverarbeitete Anchortexte. Im Vergleich zu den vorherigen Experimenten wird die Dokumentenmenge nicht zweigeteilt untersucht, da die Reduzierung auf Dokumente mit Anchortexten zu einschränkend wäre.

Allgemein lässt sich erkennen, dass die Indexierung der generierten Queries aller BERT-, BART- und T5-`base`-Modelle das Volltext-Retrieval auf Dokumententexten sowie die Indexierung von Doc2Query-Queries um das Doppelte übertreffen. Einzige Ausnahme bildet dahingehend R@1000. Dies lässt sich aber auf die betrachtete Dokumentenmenge zurückführen, da diese nur einen kleinen Teil von `clueweb09` ausmacht.

Die Verarbeitung von Anchortexten führt zu keiner Verbesserung der Metrikwerte. Eine Analyse der Anchortextmengen zeigt, dass viele Einträge viele doppelte Anchortexte enthalten, welche durch die Vorverarbeitung auf einen Anchortext reduziert werden. Durch Self-Attention sind die Modelle in der Lage, redundante Terme selbstständig zu ignorieren. Somit lassen sich die nahezu äquivalenten Effektivitäten erklären. In Anbetracht eines datenarmen Szenarios, mit der Generierung ohne Zuhilfenahme zusätzlicher Dokumentinformationen, steigert die Vorverarbeitung der Anchortexte hingegen die Metrikwerte um bis zu 5 Prozentpunkte. Der Vorteil des Fine-Tunings auf Anchortexten scheint dafür der ausschlaggebende Grund zu sein. Beispiele für generierte Queries lassen sich der Tabelle 4.11 entnehmen.

Retrieval auf Anchortexten zeigt eine äquivalente Effektivität zu Volltext-Retrieval. Somit scheinen die Anchortexte eine hohe Informationsdichte zu enthalten. Zumindest auf `clueweb09` schaffen sie es somit sogar ohne Umwandlung durch Machine-Learning-Modelle das Vocabulary-Mismatch-Problem zu mildern. Die daraus generierten Queries sind in der Lage, die enthaltenen Informationen zu bündeln und somit noch bessere Ergebnisse zu erzielen.

Kapitel 5

Fazit

In diesem abschließenden Kapitel werden nochmals die erzielten Beiträge dieser Arbeit zusammengefasst und ein Ausblick für zukünftige Untersuchungen gegeben.

5.1 Beiträge

Im Kontext des Information Retrieval können Queries für direkte oder indirekte Verbesserungen von Algorithmen und Methoden verwendet werden. Da dies aber eine Vielzahl von verfügbaren Query-Log-Daten voraussetzt, können Modelle wie Doc2Query [NYLC19, FNLE19] künstliche Queries mittels externer Daten wie Dokumententexte erzeugen. Diese Arbeit untersucht daher die Eignung von Anchortexten für die Generierung von künstlichen Queries mittels verschiedener Machine-Learning-Modelle. Die Verwendung des Datensatzes MS MARCO [NRS+16] führt zum kategorischen Training auf zwei Varianten des Datensatzes. Doc2Query folgend werden Query-Passage-Paare aus `msmarco-passage` betrachtet. Zusätzlich werden Query-Dokument-Paare betrachtet, welche aus `msmarco-document` stammen. Bezüglich der Evaluation werden die Modelle auf Testdatensätzen ihres jeweiligen Teildatensatzes bewertet. Syntaktische und semantische Ähnlichkeiten von generierten und echten Queries werden durch die Metriken ROUGE, BLEU und BERTScore bewertet. Inhaltliche Beziehungen werden in Retrieval-Szenarien auf zugehörigen TREC-Datensätzen analysiert. Schlussendlich werden die besten Modelle beider Varianten in Retrieval-Szenarien auf dem neutralen Datensatz ClueWeb09[1] evaluiert. Die Ergebnisse lassen sich in diese drei Betrachtungen einteilen.

[1] `https://lemurproject.org/clueweb09/`

Ergebnisse auf `msmarco-document` Ein auf Anchor- und Dokumententexten trainiertes T5-`base`-Modell erzielt eine Vervierfachung des BLEU-Scores auf 0,21 im Vergleich zu Doc2Query. ROUGE-Score und BERTScore sind dabei nahezu verdoppelt. Allgemein übertreffen alle Modelle die Metrikwerte von Doc2Query. Die von T5-`base` generierten Queries sind somit um ein vielfaches ähnlicher zu echten Queries als durch Doc2Query generierte Queries. Ein Fine-Tuning von T5-`base` auf 10.000 Query-Dokument-Paaren führt lediglich zu minimalen Verschlechterungen der Metrikwerte. In den Retrieval-Szenarien ist die Retrieval-Effektivität der T5-`base`-Queries in Konkatenation mit Dokumententexten gleich zu der Retrieval-Effektivität mit Doc2Query generierten Queries. Eine manuelle Analyse zeigt, das T5-`base` kurze Queries generiert, die den Charakteristiken von echten Queries eher entsprechen als Doc2Query-Queries.

Ergebnisse auf `msmarco-passage` Für die Verwendung von Query-Passage-Paaren erzielt BART-`base` mit Training auf Anchor- und Web-Dokumententexten die größte Effektivität. Es erzielt einen BLEU-Score von 0,17 und ist das einzige Modell, welches Doc2query auf `msmarco-passage` übertrifft. T5 und BERT erzielen generieren durch die wohlgeformten Queries des Datensatzes ähnliche Queries wie Doc2Query, obwohl lediglich 300.000 Query-Passage-Paare zum Fine-Tuning verwendet werden. Die Retrieval-Szenarien zeigen ähnliche Verhältnisse wie auf `msmarco-passage`. Die Betrachtung von datenarmen Szenarien mit der Generierung von Queries nur durch Anchortexte, zeigt aber eine höhere Retrieval-Effektivität durch T5-`base` als durch Doc2Query.

Ergebnisse auf `clueweb09` Eine Mischung aus längeren wohlgeformten und kurzen Queries macht `clueweb09` zum idealen Test für die Modelle von `msmarco-document` und `msmarco-passage`. Beide sind markante Charakteristiken der einzelnen Datensätze und haben maßgeblichen Einfluss auf das Fine-Tuning der Modelle. Ein Unterschied in der Effektivität des besten T5-`base`-Modells auf `msmarco-document` und `msmarco-passage` ist nicht festzustellen. Auffällig hingegen sind starke Verbesserungen der Retrieval-Effektivität im Vergleich zu Doc2Query. Die Queries der erzeugten Modelle scheinen somit bessere inhaltliche Bezüge zu den Ausgangsdokumenten herzustellen.

5.2 Ausblick

Diese Arbeit zeigt durch die Erstellung diverser Modelle, dass die Generierung von künstlichen Queries durch Anchortexte eine valide Alternative zur Generierung von Queries durch Dokumententexte darstellt. Dennoch ergaben sich

bei der Untersuchung einige Einschränkungen die für Betrachtungen zukünftiger Arbeiten dienen können.

Die Beschränkung der Eingabe der Transformer-Modelle erlaubt keine vollständige Übergabe von Dokumenten- und Anchortexten. Somit muss die Eingabe an der maximalen Tokenlänge abgeschnitten werden. Dadurch entstehen variierende Eingabeverhältnisse zwischen Dokumenten- und Anchortexten. Kurze Dokumente werden mit mehr Anchortexten aufgefüllt und bei langen Dokumenten reichen schon wenige Anchortexte bis das Eingabelimit erreicht ist. Eine Vereinheitlichung dieser Verhältnisse könnte somit zu Verbesserungen der Ergebnisse führen. Zusätlich würden sich ebenfalls Transformer-Modelle mit größerer Eingabegröße anbieten. Longformer-Modelle [BPC20] sind beispielsweise durch optimierte Self-Attention-Mechanismen in der Lage anstatt eines quadratischen, ein lineares Speicherwachstum zu ermöglichen.

Neben der Betrachtung von automatischen Metriken wie BLEU, ROUGE und BERTScore zur Evaulierung der Ähnlichkeit von generierten zu echten Queries, würde die Betrachtung einer menschlichen Evaluation mehr Aufschluss über die Qualität der generierten Queries liefern. Beispielsweise könnte nach dem Jeopardy-Prinzip von Annotatoren verlangt werden, passende Queries zu einem Dokumententext zu generieren. Diese könnten dann mit den Referenz-Queries der Datensätze und den generierten Queries verglichen werden. Die Betrachtungen dieser Arbeit limitierten Query-Dokument-Paare, sodass in den Datensätzen kein Dokument mehrere Queries besitzen konnte. Da Queries als eine Art Label für ein Dokument agieren, lag die Vermutung nahe, dass ein Dokument mit unterschiedlichen Queries beim Fine-Tuning der Modelle zu Widersprüchen führen könnte. Weiter Untersuchgen könnten sich dieser Überlegung annehmen und Modelle beispielsweise auf dem gesamten Korpus an Query-Dokument-Paaren trainieren. Die Verwendung eines künstlich generierten Datensatzes als Trainingsgrundlage von beispielweise Deep-Learning-Retrieval-Modellen würde weitere Aufschlüsse über die inhaltliche Qualität der generierten Queries liefern. Es würde sich eine Betrachtung anbieten, bei der Modelle auf echten Queries von MS MARCO und künstlich generierten Queries durch MS MARCO betrachtet werden.

Schlussendlich wäre noch die Betrachtung weiterer Sprachmodelle eine weitere Vertiefung der, in dieser Arbeit, betrachteten Aspekte. Durch den Aufschwung von Large-Language-Models wie GPT-4 [Ope23] ist die Parameteranzahl stark angestiegen, wodurch herausragende Leistungen im Bereich der Textverarbeitung erzielt werden. Eine mögliche Überlegung wäre somit die

Generierung von Queries durch Konversations-Modelle wie ChatGPT[2]. Durch Eingaben mit Präfixen die ähnliche zu den Präfixen von T5 sind, lassen sich bereits Queries erzeugen. Eine erste Analyse zeigt aber, dass diese wenig Variation in ihrer Syntax enthalten. Eine geschickte Manipulation der Eingaben wäre somit erforderlich, um solche Aspekte auszugleichen.

[2] https://openai.com/blog/chatgpt

Anhang A

Ergebnisstabellen

`msmarco-document`

Tabelle A.1: Übersicht der ROUGE-Scores für auf `msmarco-document` trainierte BERT-`base`-Modelle. Angegeben sind Precision (P), Recall (R) und F1-Score (F1). Zusätzliche Kennzeichungen sind vor- und unverarbeitete Anchortexte (V/U) und die Konkatenation von Anchortexten mit Dokumententitel- und text (ATD).

Ansatz	**ROUGE-1**			**ROUGE-2**			**ROUGE-L**		
	P	R	F1	P	R	F1	P	R	F1
BERT-base 10K (U)	0,17	0,18	0,17	0,04	0,04	0,03	0,17	0,18	0,16
BERT-base 10K (U, ATD)	0,21	0,21	0,20	0,05	0,05	0,04	0,21	0,21	0,19
BERT-base 10K (V)	0,18	0,19	0,17	0,04	0,04	0,04	0,17	0,18	0,17
BERT-base 10K (V, ATD)	0,18	0,19	0,17	0,04	0,04	0,04	0,17	0,18	0,16
BERT-base 25K (U)	0,16	0,17	0,15	0,04	0,04	0,03	0,16	0,16	0,15
BERT-base 25K (U, ATD)	0,29	0,29	0,27	0,09	0,09	0,08	0,28	0,28	0,26
BERT-base 25K (V)	0,20	0,20	0,18	0,05	0,05	0,04	0,20	0,19	0,18
BERT-base 25K (V, ATD)	0,18	0,19	0,17	0,06	0,06	0,05	0,18	0,19	0,17
BERT-base 50K (U)	0,16	0,17	0,15	0,05	0,05	0,05	0,16	0,16	0,15
BERT-base 50K (U, ATD)	0,25	0,25	0,23	0,08	0,08	0,07	0,24	0,24	0,22
BERT-base 50K (V)	0,28	0,28	0,26	0,09	0,09	0,08	0,27	0,27	0,25
BERT-base 50K (V, ATD)	0,36	0,37	0,34	0,14	0,14	0,13	0,35	0,36	0,34
BERT-base 100K (U)	0,31	0,31	0,29	0,12	0,11	0,11	0,31	0,30	0,29
BERT-base 100K (U, ATD)	0,37	0,36	0,34	0,15	0,14	0,13	0,36	0,35	0,34
BERT-base 100K (V)	0,32	0,31	0,29	0,12	0,11	0,10	0,31	0,30	0,28
BERT-base 100K (V, ATD)	0,37	0,37	0,35	0,15	0,14	0,13	0,37	0,36	0,34
BERT-base 500K (U)	0,37	0,34	0,33	0,14	0,13	0,13	0,36	0,33	0,32
BERT-base 500K (U, ATD)	0,41	0,38	0,37	0,17	0,16	0,15	0,40	0,37	0,36
BERT-base 500K (V)	0,37	0,34	0,33	0,15	0,13	0,13	0,36	0,33	0,32
BERT-base 500K (V, ATD)	0,25	0,21	0,21	0,07	0,06	0,06	0,24	0,21	0,21
BERT-base 1000K (U)	0,37	0,34	0,33	0,14	0,13	0,13	0,36	0,33	0,32
BERT-base 1000K (U, ATD)	0,40	0,37	0,36	0,16	0,15	0,14	0,39	0,36	0,35
BERT-base 1000K (V)	0,39	0,36	0,35	0,16	0,14	0,14	0,38	0,35	0,34
BERT-base 1000K (V, ATD)	0,40	0,40	0,37	0,17	0,17	0,15	0,39	0,39	0,36

Tabelle A.2: Übersicht der ROUGE-Scores für auf `msmarco-document` trainierte BERT-`large`-Modelle. Angegeben sind Precision (P), Recall (R) und F1-Score (F1). Zusätzliche Kennzeichungen sind vor- und unverarbeitete Anchortexte (V/U) und die Konkatenation von Anchortexten mit Dokumententitel- und text (ATD).

Ansatz	**ROUGE-1**			**ROUGE-2**			**ROUGE-L**		
	P	R	F1	P	R	F1	P	R	F1
BERT-large 10K (U)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
BERT-large 10K (U, ATD)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
BERT-large 10K (V)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
BERT-large 10K (V, ATD)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
BERT-large 25K (U)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
BERT-large 25K (U, ATD)	0,01	0,00	0,01	0,00	0,00	0,00	0,01	0,00	0,01
BERT-large 25K (V)	0,01	0,00	0,00	0,00	0,00	0,00	0,01	0,00	0,00
BERT-large 25K (V, ATD)	0,01	0,00	0,00	0,00	0,00	0,00	0,01	0,00	0,00
BERT-large 50K (U)	0,06	0,03	0,03	0,00	0,00	0,00	0,06	0,03	0,03
BERT-large 50K (U, ATD)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
BERT-large 50K (V)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
BERT-large 50K (V, ATD)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
BERT-large 100K (U)	0,03	0,01	0,02	0,00	0,00	0,00	0,03	0,01	0,02
BERT-large 100K (U, ATD)	0,02	0,00	0,01	0,00	0,00	0,00	0,02	0,00	0,01
BERT-large 100K (V)	0,04	0,03	0,03	0,00	0,00	0,00	0,04	0,03	0,03
BERT-large 100K (V, ATD)	0,09	0,05	0,06	0,00	0,00	0,00	0,09	0,05	0,06
BERT-large 500K (U)	0,04	0,04	0,03	0,00	0,00	0,00	0,04	0,03	0,03
BERT-large 500K (U, ATD)	0,22	0,20	0,20	0,04	0,04	0,04	0,21	0,20	0,19
BERT-large 500K (V)	0,08	0,04	0,05	0,00	0,00	0,00	0,08	0,04	0,05
BERT-large 500K (V, ATD)	0,16	0,14	0,14	0,01	0,01	0,01	0,16	0,13	0,13
BERT-large 1000K (U)	0,22	0,18	0,19	0,04	0,04	0,04	0,22	0,18	0,18
BERT-large 1000K (U, ATD)	0,15	0,14	0,13	0,01	0,01	0,01	0,14	0,13	0,13
BERT-large 1000K (V)	0,13	0,10	0,11	0,01	0,01	0,01	0,12	0,10	0,10
BERT-large 1000K (V, ATD)	0,20	0,18	0,18	0,03	0,03	0,03	0,19	0,18	0,17

Tabelle A.3: Übersicht von BLEU-Scores und BERTScores für auf `msmarco-document` trainierte BERT-`base`-Modelle. Angegeben sind Precision (P), Recall (R) und F1-Score (F1). Zusätzliche Kennzeichnungen sind vor- und unverarbeitete Anchortexte (V/U) und die Konkatenation von Anchortexten mit Dokumententitel- und text (ATD).

Ansatz	**BLEU**	**BERTScore**		
		P	R	F1
BERT-base 10K (U)	0,01	0,19	0,17	0,18
BERT-base 10K (U, ATD)	0,02	0,22	0,20	0,21
BERT-base 10K (V)	0,01	0,19	0,18	0,18
BERT-base 10K (V, ATD)	0,02	0,21	0,19	0,20
BERT-base 25K (U)	0,01	0,20	0,17	0,18
BERT-base 25K (U, ATD)	0,04	0,30	0,27	0,29
BERT-base 25K (V)	0,02	0,22	0,20	0,21
BERT-base 25K (V, ATD)	0,03	0,26	0,24	0,25
BERT-base 50K (U)	0,02	0,25	0,21	0,23
BERT-base 50K (U, ATD)	0,04	0,29	0,27	0,28
BERT-base 50K (V)	0,03	0,30	0,27	0,29
BERT-base 50K (V, ATD)	0,07	0,37	0,36	0,37
BERT-base 100K (U)	0,06	0,33	0,31	0,32
BERT-base 100K (U, ATD)	0,08	0,38	0,36	0,37
BERT-base 100K (V)	0,05	0,33	0,31	0,32
BERT-base 100K (V, ATD)	0,08	0,38	0,36	0,37
BERT-base 500K (U)	0,07	0,37	0,34	0,35
BERT-base 500K (U, ATD)	0,10	0,40	0,38	0,39
BERT-base 500K (V)	0,07	0,37	0,34	0,35
BERT-base 500K (V, ATD)	0,03	0,31	0,26	0,28
BERT-base 1000K (U)	0,07	0,37	0,34	0,35
BERT-base 1000K (U, ATD)	0,09	0,39	0,36	0,38
BERT-base 1000K (V)	0,08	0,39	0,36	0,37
BERT-base 1000K (V, ATD)	0,09	0,40	0,39	0,39

Tabelle A.4: Übersicht von BLEU-Scores und BERTScores für auf `msmarco-document` trainierte BERT-`large`-Modelle. Angegeben sind Precision (P), Recall (R) und F1-Score (F1). Zusätzliche Kennzeichungen sind vor- und unverarbeitete Anchortexte (V/U) und die Konkatenation von Anchortexten mit Dokumententitel- und text (ATD).

Ansatz	**BLEU**	**BERTScore**		
		P	R	F1
BERT-large 10K (U)	0,00	0,00	0,00	0,00
BERT-large 10K (U, ATD)	0,00	0,00	0,00	0,00
BERT-large 10K (V)	0,00	0,07	0,00	0,00
BERT-large 10K (V, ATD)	0,00	0,03	0,00	0,00
BERT-large 25K (U)	0,00	0,00	0,00	0,00
BERT-large 25K (U, ATD)	0,00	0,20	0,00	0,02
BERT-large 25K (V)	0,00	0,22	0,00	0,03
BERT-large 25K (V, ATD)	0,00	0,22	0,00	0,04
BERT-large 50K (U)	0,00	0,17	0,00	0,05
BERT-large 50K (U, ATD)	0,00	0,19	0,00	0,02
BERT-large 50K (V)	0,00	0,00	0,00	0,00
BERT-large 50K (V, ATD)	0,00	0,00	0,00	0,00
BERT-large 100K (U)	0,00	0,17	0,00	0,03
BERT-large 100K (U, ATD)	0,00	0,17	0,00	0,01
BERT-large 100K (V)	0,00	0,04	0,00	0,00
BERT-large 100K (V, ATD)	0,00	0,17	0,00	0,04
BERT-large 500K (U)	0,00	0,00	0,00	0,00
BERT-large 500K (U, ATD)	0,01	0,17	0,13	0,15
BERT-large 500K (V)	0,00	0,13	0,00	0,03
BERT-large 500K (V, ATD)	0,00	0,14	0,05	0,09
BERT-large 1000K (U)	0,01	0,20	0,13	0,16
BERT-large 1000K (U, ATD)	0,00	0,11	0,06	0,08
BERT-large 1000K (V)	0,00	0,10	0,01	0,05
BERT-large 1000K (V, ATD)	0,01	0,15	0,11	0,13

Tabelle A.5: Übersicht der ROUGE-Scores für auf `msmarco-document` trainierte BART-`base`-Modelle. Angegeben sind Precision (P), Recall (R) und F1-Score (F1). Zusätzliche Kennzeichnungen sind vor- und unverarbeitete Anchortexte (V/U) und die Konkatenation von Anchortexten mit Dokumententitel- und text (ATD).

Ansatz	**ROUGE-1**			**ROUGE-2**			**ROUGE-L**		
	P	R	F1	P	R	F1	P	R	F1
BART-base 10K (U)	0,38	0,37	0,35	0,16	0,15	0,15	0,37	0,36	0,34
BART-base 10K (U, ATD)	0,43	0,43	0,41	0,20	0,19	0,18	0,42	0,42	0,39
BART-base 10K (V)	0,40	0,38	0,37	0,18	0,16	0,16	0,39	0,37	0,36
BART-base 10K (V, ATD)	0,45	0,43	0,42	0,21	0,19	0,19	0,44	0,42	0,40
BART-base 25K (U)	0,44	0,39	0,39	0,19	0,16	0,16	0,43	0,38	0,38
BART-base 25K (U, ATD)	0,44	0,42	0,41	0,20	0,19	0,18	0,43	0,41	0,40
BART-base 25K (V)	0,45	0,38	0,39	0,20	0,16	0,16	0,44	0,37	0,38
BART-base 25K (V, ATD)	0,46	0,44	0,43	0,22	0,20	0,20	0,45	0,42	0,42
BART-base 50K (U)	0,44	0,39	0,39	0,20	0,17	0,17	0,43	0,38	0,38
BART-base 50K (U, ATD)	0,49	0,44	0,45	0,23	0,20	0,20	0,48	0,43	0,43
BART-base 50K (V)	0,44	0,39	0,39	0,20	0,17	0,17	0,43	0,39	0,38
BART-base 50K (V, ATD)	0,48	0,46	0,44	0,24	0,21	0,21	0,47	0,44	0,43
BART-base 100K (U)	0,44	0,40	0,40	0,20	0,17	0,18	0,43	0,39	0,39
BART-base 100K (U, ATD)	0,50	0,46	0,46	0,24	0,22	0,22	0,48	0,45	0,44
BART-base 100K (V)	0,45	0,40	0,40	0,21	0,18	0,18	0,44	0,39	0,39
BART-base 100K (V, ATD)	0,50	0,46	0,46	0,24	0,22	0,21	0,49	0,45	0,44
BART-base 500K (U)	0,50	0,42	0,43	0,23	0,19	0,20	0,49	0,41	0,42
BART-base 500K (U, ATD)	0,54	0,47	0,48	0,27	0,23	0,23	0,53	0,46	0,47
BART-base 500K (V)	0,49	0,42	0,43	0,23	0,19	0,20	0,48	0,41	0,42
BART-base 500K (V, ATD)	0,51	0,46	0,46	0,25	0,22	0,22	0,49	0,45	0,44
BART-base 1000K (U)	0,49	0,42	0,43	0,23	0,19	0,20	0,48	0,41	0,42
BART-base 1000K (U, ATD)	0,54	0,45	0,47	0,26	0,21	0,22	0,53	0,44	0,46
BART-base 1000K (V)	0,47	0,41	0,42	0,22	0,18	0,19	0,46	0,40	0,41
BART-base 1000K (V, ATD)	0,53	0,46	0,47	0,26	0,22	0,22	0,52	0,45	0,46

Tabelle A.6: Übersicht der ROUGE-Scores für auf `msmarco-document` trainierte BART-`large`-Modelle. Angegeben sind Precision (P), Recall (R) und F1-Score (F1). Zusätzliche Kennzeichungen sind vor- und unverarbeitete Anchortexte (V/U) und die Konkatenation von Anchortexten mit Dokumententitel- und text (ATD).

Ansatz	**ROUGE-1**			**ROUGE-2**			**ROUGE-L**		
	P	R	F1	P	R	F1	P	R	F1
BART-large 10K (U)	0,41	0,37	0,37	0,17	0,15	0,15	0,40	0,37	0,36
BART-large 10K (U, ATD)	0,41	0,45	0,41	0,19	0,21	0,19	0,40	0,44	0,39
BART-large 10K (V)	0,41	0,36	0,36	0,17	0,15	0,15	0,40	0,36	0,36
BART-large 10K (V, ATD)	0,41	0,45	0,41	0,19	0,20	0,18	0,40	0,43	0,39
BART-large 25K (U)	0,41	0,38	0,37	0,18	0,16	0,15	0,41	0,37	0,36
BART-large 25K (U, ATD)	0,43	0,43	0,41	0,20	0,19	0,18	0,42	0,42	0,40
BART-large 25K (V)	0,41	0,39	0,38	0,18	0,17	0,16	0,40	0,38	0,37
BART-large 25K (V, ATD)	0,44	0,43	0,41	0,21	0,19	0,19	0,43	0,42	0,40
BART-large 50K (U)	0,42	0,37	0,37	0,18	0,16	0,16	0,41	0,37	0,36
BART-large 50K (U, ATD)	0,49	0,45	0,44	0,23	0,20	0,20	0,47	0,44	0,43
BART-large 50K (V)	0,43	0,40	0,39	0,20	0,17	0,17	0,42	0,39	0,38
BART-large 50K (V, ATD)	0,49	0,45	0,44	0,23	0,21	0,21	0,47	0,43	0,43
BART-large 100K (U)	0,38	0,37	0,35	0,17	0,15	0,15	0,37	0,36	0,35
BART-large 100K (U, ATD)	0,50	0,46	0,46	0,24	0,22	0,22	0,49	0,45	0,44
BART-large 100K (V)	0,07	0,04	0,05	0,00	0,00	0,00	0,07	0,04	0,04
BART-large 100K (V, ATD)	0,49	0,44	0,44	0,23	0,20	0,20	0,48	0,42	0,43
BART-large 500K (U)	0,43	0,42	0,41	0,21	0,19	0,19	0,42	0,41	0,40
BART-large 500K (U, ATD)	0,51	0,45	0,46	0,25	0,21	0,21	0,50	0,44	0,45
BART-large 500K (V)	0,44	0,41	0,40	0,21	0,19	0,19	0,43	0,40	0,39
BART-large 500K (V, ATD)	0,49	0,44	0,44	0,24	0,20	0,20	0,48	0,43	0,43
BART-large 1000K (U)	0,48	0,42	0,42	0,22	0,19	0,19	0,46	0,41	0,41
BART-large 1000K (U, ATD)	0,53	0,46	0,47	0,26	0,22	0,22	0,51	0,45	0,45
BART-large 1000K (V)	0,48	0,41	0,42	0,22	0,19	0,19	0,47	0,40	0,41
BART-large 1000K (V, ATD)	0,50	0,44	0,45	0,24	0,20	0,21	0,49	0,43	0,43

Tabelle A.7: Übersicht von BLEU-Scores und BERTScores für auf `msmarco-document` trainierte BART-`base`-Modelle. Angegeben sind Precision (P), Recall (R) und F1-Score (F1). Zusätzliche Kennzeichnungen sind vor- und unverarbeitete Anchortexte (V/U) und die Konkatenation von Anchortexten mit Dokumententitel- und text (ATD).

Ansatz	**BLEU**	**BERTScore**		
		P	R	F1
BART-base 10K (U)	0,09	0,38	0,36	0,37
BART-base 10K (U, ATD)	0,13	0,43	0,42	0,43
BART-base 10K (V)	0,11	0,40	0,38	0,39
BART-base 10K (V, ATD)	0,14	0,44	0,42	0,43
BART-base 25K (U)	0,12	0,45	0,40	0,42
BART-base 25K (U, ATD)	0,14	0,45	0,42	0,43
BART-base 25K (V)	0,12	0,44	0,38	0,41
BART-base 25K (V, ATD)	0,15	0,46	0,44	0,45
BART-base 50K (U)	0,11	0,43	0,39	0,41
BART-base 50K (U, ATD)	0,16	0,47	0,45	0,46
BART-base 50K (V)	0,11	0,43	0,39	0,41
BART-base 50K (V, ATD)	0,16	0,48	0,45	0,46
BART-base 100K (U)	0,12	0,43	0,40	0,42
BART-base 100K (U, ATD)	0,17	0,49	0,46	0,47
BART-base 100K (V)	0,13	0,44	0,41	0,42
BART-base 100K (V, ATD)	0,17	0,49	0,46	0,47
BART-base 500K (U)	0,13	0,47	0,42	0,44
BART-base 500K (U, ATD)	0,19	0,50	0,47	0,49
BART-base 500K (V)	0,14	0,46	0,42	0,44
BART-base 500K (V, ATD)	0,17	0,48	0,45	0,47
BART-base 1000K (U)	0,14	0,46	0,43	0,45
BART-base 1000K (U, ATD)	0,17	0,51	0,45	0,48
BART-base 1000K (V)	0,13	0,45	0,41	0,43
BART-base 1000K (V, ATD)	0,18	0,51	0,46	0,48

Tabelle A.8: Übersicht von BLEU-Scores und BERTScores für auf `msmarco-document` trainierte BART-`large`-Modelle. Angegeben sind Precision (P), Recall (R) und F1-Score (F1). Zusätzliche Kennzeichungen sind vor- und unverarbeitete Anchortexte (V/U) und die Konkatenation von Anchortexten mit Dokumententitel- und text (ATD).

Ansatz	**BLEU**	**BERTScore**		
		P	R	F1
BART-large 10K (U)	0,11	0,41	0,39	0,40
BART-large 10K (U, ATD)	0,12	0,42	0,43	0,42
BART-large 10K (V)	0,11	0,41	0,38	0,39
BART-large 10K (V, ATD)	0,12	0,42	0,43	0,42
BART-large 25K (U)	0,11	0,42	0,38	0,40
BART-large 25K (U, ATD)	0,13	0,43	0,42	0,42
BART-large 25K (V)	0,11	0,40	0,39	0,40
BART-large 25K (V, ATD)	0,14	0,44	0,43	0,43
BART-large 50K (U)	0,11	0,41	0,37	0,39
BART-large 50K (U, ATD)	0,15	0,47	0,45	0,46
BART-large 50K (V)	0,11	0,43	0,39	0,41
BART-large 50K (V, ATD)	0,17	0,47	0,45	0,46
BART-large 100K (U)	0,10	0,41	0,38	0,39
BART-large 100K (U, ATD)	0,18	0,48	0,46	0,47
BART-large 100K (V)	0,00	0,00	0,00	0,00
BART-large 100K (V, ATD)	0,16	0,47	0,43	0,45
BART-large 500K (U)	0,14	0,42	0,41	0,41
BART-large 500K (U, ATD)	0,17	0,49	0,45	0,47
BART-large 500K (V)	0,12	0,42	0,39	0,41
BART-large 500K (V, ATD)	0,16	0,47	0,43	0,45
BART-large 1000K (U)	0,15	0,32	0,28	0,30
BART-large 1000K (U, ATD)	0,18	0,50	0,46	0,48
BART-large 1000K (V)	0,14	0,46	0,41	0,43
BART-large 1000K (V, ATD)	0,17	0,48	0,44	0,46

Tabelle A.9: Übersicht der ROUGE-Scores für auf `msmarco-document` trainierte T5-`base`-Modelle. Angegeben sind Precision (P), Recall (R) und F1-Score (F1). Zusätzliche Kennzeichnungen sind vor- und unverarbeitete Anchortexte (V/U), die Konkatenation von Anchortexten mit Dokumententitel- und text (ATD) und `summarize` als Eingabebefehl (*).

Ansatz	**ROUGE-1**			**ROUGE-2**			**ROUGE-L**		
	P	R	F1	P	R	F1	P	R	F1
T5-base 10K (U)	0,34	0,32	0,31	0,13	0,12	0,11	0,33	0,31	0,30
T5-base 10K (U, ATD)	0,39	0,38	0,36	0,16	0,15	0,15	0,38	0,37	0,35
T5-base 10K (V)	0,36	0,33	0,32	0,14	0,13	0,12	0,35	0,33	0,32
T5-base 10K (V, ATD)	0,40	0,39	0,38	0,17	0,16	0,15	0,39	0,38	0,36
T5-base 10K (V, ATD)*	0,37	0,36	0,34	0,15	0,14	0,13	0,36	0,35	0,33
T5-base 25K (U)	0,35	0,32	0,31	0,13	0,12	0,11	0,34	0,31	0,30
T5-base 25K (U, ATD)	0,39	0,38	0,36	0,15	0,15	0,14	0,37	0,36	0,35
T5-base 25K (V)	0,36	0,33	0,32	0,14	0,13	0,12	0,35	0,32	0,32
T5-base 25K (V, ATD)	0,41	0,40	0,38	0,17	0,16	0,15	0,40	0,38	0,37
T5-base 25K (V, ATD)*	0,38	0,37	0,35	0,15	0,14	0,13	0,37	0,36	0,34
T5-base 50K (U)	0,36	0,33	0,32	0,14	0,12	0,12	0,35	0,32	0,32
T5-base 50K (U, ATD)	0,41	0,39	0,37	0,17	0,16	0,15	0,40	0,38	0,36
T5-base 50K (V)	0,37	0,34	0,33	0,15	0,13	0,13	0,36	0,33	0,33
T5-base 50K (V, ATD)	0,41	0,39	0,38	0,17	0,16	0,15	0,40	0,38	0,37
T5-base 50K (V, ATD)*	0,40	0,37	0,36	0,16	0,15	0,14	0,39	0,36	0,35
T5-base 100K (U)	0,36	0,34	0,33	0,14	0,13	0,13	0,35	0,33	0,32
T5-base 100K (U, ATD)	0,41	0,40	0,38	0,17	0,16	0,15	0,40	0,38	0,37
T5-base 100K (V)	0,37	0,35	0,34	0,15	0,13	0,13	0,36	0,34	0,33
T5-base 100K (V, ATD)	0,41	0,41	0,39	0,18	0,17	0,16	0,40	0,39	0,38
T5-base 100K (V, ATD)*	0,40	0,39	0,37	0,16	0,16	0,15	0,39	0,38	0,36
T5-base 500K (U)	0,38	0,35	0,34	0,15	0,14	0,14	0,37	0,34	0,33
T5-base 500K (U, ATD)	0,43	0,40	0,39	0,18	0,17	0,16	0,42	0,39	0,38
T5-base 500K (V)	0,39	0,36	0,35	0,16	0,14	0,14	0,38	0,35	0,34
T5-base 500K (V, ATD)	0,44	0,41	0,40	0,19	0,17	0,17	0,43	0,40	0,39
T5-base 500K (V, ATD)*	0,43	0,41	0,40	0,18	0,17	0,16	0,42	0,39	0,38
T5-base 1000K (U)	0,38	0,36	0,35	0,15	0,14	0,14	0,37	0,35	0,34
T5-base 1000K (U, ATD)	0,43	0,42	0,40	0,19	0,18	0,17	0,42	0,41	0,39
T5-base 1000K (V)	0,38	0,37	0,35	0,16	0,15	0,14	0,37	0,36	0,34
T5-base 1000K (V, ATD)	0,44	0,42	0,41	0,19	0,18	0,17	0,43	0,41	0,40
T5-base 1000K (V, ATD)*	0,43	0,41	0,39	0,18	0,17	0,16	0,42	0,40	0,38

Tabelle A.10: Übersicht der ROUGE-Scores für auf `msmarco-document` trainierte T5-`large`-Modelle. Angegeben sind Precision (P), Recall (R) und F1-Score (F1). Zusätzliche Kennzeichnungen sind vor- und unverarbeitete Anchortexte (V/U) und die Konkatenation von Anchortexten mit Dokumententitel- und text (ATD).

Ansatz	**ROUGE-1**			**ROUGE-2**			**ROUGE-L**		
	P	R	F1	P	R	F1	P	R	F1
T5-large 10K (U)	0,36	0,36	0,34	0,14	0,14	0,13	0,35	0,35	0,33
T5-large 10K (U, ATD)	0,41	0,42	0,39	0,18	0,18	0,17	0,40	0,40	0,38
T5-large 10K (V)	0,36	0,36	0,34	0,15	0,15	0,14	0,35	0,35	0,33
T5-large 10K (V, ATD)	0,42	0,43	0,40	0,19	0,18	0,17	0,41	0,41	0,39
T5-large 25K (U)	0,36	0,34	0,32	0,14	0,13	0,12	0,35	0,33	0,32
T5-large 25K (U, ATD)	0,40	0,38	0,37	0,17	0,16	0,15	0,39	0,37	0,36
T5-large 25K (V)	0,37	0,35	0,34	0,15	0,14	0,13	0,36	0,34	0,33
T5-large 25K (V, ATD)	0,40	0,40	0,38	0,17	0,17	0,16	0,39	0,39	0,37
T5-large 50K (U)	0,37	0,35	0,34	0,15	0,14	0,13	0,36	0,34	0,33
T5-large 50K (U, ATD)	0,43	0,41	0,40	0,18	0,17	0,16	0,42	0,40	0,38
T5-large 50K (V)	0,39	0,36	0,35	0,16	0,14	0,14	0,38	0,35	0,34
T5-large 50K (V, ATD)	0,43	0,42	0,40	0,19	0,18	0,17	0,42	0,40	0,39
T5-large 100K (U)	0,36	0,36	0,34	0,15	0,14	0,14	0,35	0,35	0,33
T5-large 100K (U, ATD)	0,42	0,42	0,40	0,18	0,18	0,17	0,41	0,41	0,38
T5-large 100K (V)	0,38	0,37	0,35	0,16	0,15	0,14	0,37	0,36	0,34
T5-large 100K (V, ATD)	0,42	0,43	0,40	0,18	0,18	0,17	0,41	0,41	0,39
T5-large 500K (U)	0,39	0,37	0,36	0,16	0,15	0,14	0,38	0,36	0,35
T5-large 500K (U, ATD)	0,44	0,43	0,41	0,20	0,19	0,18	0,43	0,41	0,40
T5-large 500K (V)	0,39	0,37	0,36	0,16	0,15	0,15	0,38	0,36	0,35
T5-large 500K (V, ATD)	0,45	0,43	0,41	0,20	0,19	0,18	0,43	0,42	0,40
T5-large 1000K (U, ATD)	0,44	0,43	0,41	0,19	0,18	0,18	0,43	0,41	0,40
T5-large 1000K (V)	0,39	0,38	0,37	0,16	0,15	0,15	0,38	0,37	0,36

Tabelle A.11: Übersicht von BLEU-Scores und BERTScores für auf `msmarco-document` trainierte T5-`base`-Modelle. Angegeben sind Precision (P), Recall (R) und F1-Score (F1). Zusätzliche Kennzeichnungen sind vor- und unverarbeitete Anchortexte (V/U), die Konkatenation von Anchortexten mit Dokumententitel- und text (ATD) und `summarize` als Eingabebefehl (*).

Ansatz	**BLEU**	**BERTScore**		
		P	R	F1
T5-base 10K (U)	0,08	0,34	0,32	0,33
T5-base 10K (U, ATD)	0,11	0,39	0,38	0,38
T5-base 10K (V)	0,08	0,35	0,34	0,35
T5-base 10K (V, ATD)	0,11	0,40	0,39	0,39
T5-base 10K (V, ATD)*	0,09	0,37	0,36	0,36
T5-base 25K (U)	0,08	0,34	0,32	0,33
T5-base 25K (U, ATD)	0,10	0,39	0,37	0,38
T5-base 25K (V)	0,09	0,36	0,33	0,34
T5-base 25K (V, ATD)	0,11	0,41	0,39	0,40
T5-base 25K (V, ATD)*	0,10	0,38	0,37	0,37
T5-base 50K (U)	0,08	0,36	0,33	0,34
T5-base 50K (U, ATD)	0,11	0,40	0,39	0,40
T5-base 50K (V)	0,09	0,37	0,35	0,36
T5-base 50K (V, ATD)	0,12	0,41	0,39	0,40
T5-base 50K (V, ATD)*	0,10	0,39	0,38	0,38
T5-base 100K (U)	0,09	0,36	0,34	0,35
T5-base 100K (U, ATD)	0,11	0,40	0,40	0,40
T5-base 100K (V)	0,09	0,37	0,35	0,36
T5-base 100K (V, ATD)	0,12	0,41	0,40	0,41
T5-base 100K (V, ATD)*	0,11	0,39	0,39	0,39
T5-base 500K (U)	0,10	0,38	0,36	0,37
T5-base 500K (U, ATD)	0,12	0,42	0,41	0,41
T5-base 500K (V)	0,09	0,38	0,36	0,37
T5-base 500K (V, ATD)	0,13	0,43	0,42	0,42
T5-base 500K (V, ATD)*	0,12	0,42	0,41	0,41
T5-base 1000K (U)	0,09	0,38	0,36	0,37
T5-base 1000K (U, ATD)	0,13	0,43	0,41	0,42
T5-base 1000K (V)	0,10	0,38	0,37	0,37
T5-base 1000K (V, ATD)	0,13	0,43	0,42	0,43
T5-base 1000K (V, ATD)*	0,13	0,42	0,40	0,41

Tabelle A.12: Übersicht von BLEU-Scores und BERTScores für auf `msmarco-document` trainierte T5-`large`-Modelle. Angegeben sind Precision (P), Recall (R) und F1-Score (F1). Zusätzliche Kennzeichungen sind vor- und unverarbeitete Anchortexte (V/U) und die Konkatenation von Anchortexten mit Dokumententitel- und text (ATD).

Ansatz	**BLEU**	**BERTScore**		
		P	R	F1
T5-large 10K (U)	0,09	0,36	0,36	0,36
T5-large 10K (U, ATD)	0,12	0,41	0,41	0,41
T5-large 10K (V)	0,10	0,37	0,37	0,37
T5-large 10K (V, ATD)	0,13	0,42	0,42	0,42
T5-large 25K (U)	0,08	0,35	0,34	0,35
T5-large 25K (U, ATD)	0,11	0,40	0,38	0,39
T5-large 25K (V)	0,09	0,37	0,35	0,36
T5-large 25K (V, ATD)	0,12	0,41	0,39	0,40
T5-large 50K (U)	0,09	0,37	0,35	0,36
T5-large 50K (U, ATD)	0,12	0,42	0,41	0,41
T5-large 50K (V)	0,10	0,38	0,36	0,37
T5-large 50K (V, ATD)	0,13	0,43	0,41	0,42
T5-large 100K (U)	0,09	0,37	0,36	0,37
T5-large 100K (U, ATD)	0,12	0,42	0,42	0,42
T5-large 100K (V)	0,10	0,38	0,37	0,37
T5-large 100K (V, ATD)	0,12	0,42	0,42	0,42
T5-large 500K (U)	0,10	0,39	0,37	0,38
T5-large 500K (U, ATD)	0,14	0,44	0,43	0,43
T5-large 500K (V)	0,10	0,39	0,37	0,38
T5-large 500K (V, ATD)	0,14	0,44	0,43	0,43
T5-large 1000K (U, ATD)	0,14	0,43	0,42	0,43
T5-large 1000K (V)	0,11	0,39	0,38	0,39

Anhang B

Ergebnisstabellen `msmarco-passage`

Tabelle B.1: Übersicht von ROUGE-Scores für auf `msmarco-passage` trainierte BERT-Modelle. Angegeben sind Precision (P), Recall (R) und F1-Score (F1). Zusätzliche Kennzeichungen sind vor- und unverarbeitete Anchortexte (V/U) und die Konkatenation von Anchortexten mit Passagentexten (AP). Alle Modelle wurden für zwei, fünf (5) oder einhundert (100) Epochen trainiert.

Ansatz	**ROUGE-1**			**ROUGE-2**			**ROUGE-L**		
	P	R	F1	P	R	F1	P	R	F1
BERT-base 300K (U)	0,21	0,20	0,19	0,06	0,06	0,05	0,20	0,20	0,19
BERT-base 300K (U, AP)	0,24	0,24	0,22	0,08	0,08	0,08	0,23	0,23	0,21
BERT-base 300K (V)	0,30	0,30	0,29	0,11	0,10	0,10	0,29	0,29	0,28
BERT-base 300K (V, AP)	0,38	0,38	0,36	0,15	0,15	0,15	0,36	0,36	0,35
BERT-base 300K (V, AP)$_5$	0,36	0,35	0,34	0,13	0,13	0,13	0,34	0,34	0,32
BERT-base 500K (U)	0,31	0,31	0,30	0,12	0,12	0,11	0,30	0,30	0,29
BERT-base 500K (U, AP)	0,40	0,39	0,38	0,16	0,16	0,15	0,38	0,37	0,36
BERT-base 500K (V)	0,32	0,32	0,31	0,12	0,11	0,11	0,31	0,31	0,29
BERT-base 500K (V, AP)$_{100}$	0,39	0,39	0,37	0,15	0,15	0,15	0,37	0,37	0,35
BERT-base 500K (V, AP)	0,26	0,28	0,26	0,09	0,10	0,09	0,25	0,27	0,25
BERT-base 500K (V, AP)$_5$	0,36	0,37	0,35	0,14	0,14	0,13	0,34	0,35	0,33
BERT-large 300K (U)	0,15	0,11	0,12	0,01	0,01	0,01	0,15	0,11	0,11
BERT-large 300K (U, AP)	0,16	0,18	0,15	0,02	0,02	0,02	0,15	0,17	0,15
BERT-large 300K (V)	0,12	0,17	0,13	0,01	0,02	0,01	0,11	0,15	0,12
BERT-large 300K (V, AP)	0,17	0,13	0,13	0,01	0,02	0,01	0,16	0,12	0,12
BERT-large 300K (V, AP)$_5$	0,19	0,17	0,16	0,03	0,03	0,02	0,18	0,17	0,16

Tabelle B.2: Übersicht der BLEU-Scores und BERTScores für auf `msmarco-passage` trainierte BERT-Modelle. Angegeben sind Precision (P), Recall (R) und F1-Score (F1). Zusätzliche Kennzeichungen sind vor- und unverarbeitete Anchortexte (V/U) und die Konkatenation von Anchortexten mit Passagentexten (AP). Alle Modelle wurden für zwei, fünf (5) oder einhundert (100) Epochen trainiert.

Ansatz	**BLEU**	**BERTScore**		
		P	R	F1
BERT-base 300K (U)	0,02	0,25	0,22	0,24
BERT-base 300K (U, AP)	0,04	0,34	0,31	0,32
BERT-base 300K (V)	0,05	0,34	0,33	0,34
BERT-base 300K (V, AP)	0,09	0,42	0,41	0,41
BERT-base 300K (V, AP)$_5$	0,07	0,38	0,37	0,38
BERT-base 500K (U)	0,06	0,37	0,35	0,36
BERT-base 500K (U, AP)	0,09	0,44	0,43	0,43
BERT-base 500K (V)	0,06	0,37	0,35	0,36
BERT-base 500K (V, AP)$_{100}$	0,08	0,43	0,42	0,42
BERT-base 500K (V, AP)	0,05	0,33	0,31	0,32
BERT-base 500K (V, AP)$_5$	0,08	0,40	0,39	0,39
BERT-large 300K (U)	0,00	0,05	0,00	0,01
BERT-large 300K (U, AP)	0,00	0,00	0,02	0,01
BERT-large 300K (V)	0,00	0,00	0,00	0,00
BERT-large 300K (V, AP)	0,00	0,04	0,00	0,00
BERT-large 300K (V, AP)$_5$	0,00	0,07	0,05	0,06

Tabelle B.3: Übersicht von ROUGE-Scores für auf `msmarco-passage` trainierte BART-Modelle. Angegeben sind Precision (P), Recall (R) und F1-Score (F1). Zusätzliche Kennzeichnungen sind vor- und unverarbeitete Anchortexte (V/U) und die Konkatenation von Anchortexten mit Passagentexten (AP). Alle Modelle wurden für zwei, fünf (5) oder einhundert (100) Epochen trainiert.

Ansatz	**ROUGE-1**			**ROUGE-2**			**ROUGE-L**		
	P	R	F1	P	R	F1	P	R	F1
BART-base 300K (U)	0,44	0,41	0,41	0,21	0,19	0,19	0,43	0,40	0,40
BART-base 300K (U, AP)	0,51	0,48	0,48	0,25	0,24	0,24	0,49	0,46	0,46
BART-base 300K (V)	0,44	0,41	0,41	0,20	0,19	0,19	0,42	0,39	0,39
BART-base 300K (V, AP)	0,52	0,49	0,49	0,27	0,25	0,25	0,50	0,47	0,47
BART-base 300K (V, AP)$_5$	0,47	0,47	0,45	0,23	0,22	0,22	0,45	0,45	0,43
BART-base 500K (U)	0,44	0,39	0,40	0,21	0,18	0,18	0,43	0,38	0,39
BART-base 500K (U, AP)	0,47	0,46	0,45	0,23	0,22	0,21	0,45	0,44	0,43
BART-base 500K (V)	0,44	0,39	0,40	0,21	0,18	0,18	0,43	0,38	0,39
BART-base 500K (V, AP)$_{100}$	0,52	0,48	0,49	0,27	0,25	0,25	0,50	0,46	0,47
BART-base 500K (V, AP)	0,47	0,47	0,46	0,23	0,23	0,22	0,46	0,45	0,44
BART-base 500K (V, AP)$_5$	0,50	0,45	0,46	0,24	0,22	0,22	0,48	0,44	0,44
BART-large 300K (U)	0,39	0,38	0,37	0,17	0,16	0,16	0,37	0,37	0,36
BART-large 300K (U, AP)	0,46	0,41	0,42	0,21	0,19	0,19	0,44	0,40	0,41
BART-large 300K (V)	0,40	0,39	0,38	0,18	0,17	0,17	0,39	0,38	0,37
BART-large 300K (V, AP)	0,46	0,45	0,44	0,23	0,22	0,21	0,44	0,43	0,42
BART-large 300K (V, AP)$_5$	0,49	0,45	0,45	0,24	0,21	0,21	0,47	0,43	0,44

Tabelle B.4: Übersicht der BLEU-Scores und BERTScores für auf `msmarco-passage` trainierte BART-Modelle. Angegeben sind Precision (P), Recall (R) und F1-Score (F1). Zusätzliche Kennzeichungen sind vor- und unverarbeitete Anchortexte (V/U) und die Konkatenation von Anchortexten mit Passagentexten (AP). Alle Modelle wurden für zwei, fünf (5) oder einhundert (100) Epochen trainiert.

Ansatz	**BLEU**	**BERTScore**		
		P	R	F1
BART-base 300K (U)	0,12	0,49	0,44	0,47
BART-base 300K (U, AP)	0,17	0,55	0,51	0,53
BART-base 300K (V)	0,12	0,49	0,44	0,46
BART-base 300K (V, AP)	0,17	0,56	0,52	0,54
BART-base 300K (V, AP)$_5$	0,14	0,52	0,49	0,50
BART-base 500K (U)	0,11	0,49	0,43	0,46
BART-base 500K (U, AP)	0,14	0,52	0,48	0,50
BART-base 500K (V)	0,11	0,49	0,42	0,46
BART-base 500K (V, AP)$_{100}$	0,17	0,56	0,51	0,53
BART-base 500K (V, AP)	0,14	0,53	0,48	0,51
BART-base 500K (V, AP)$_5$	0,14	0,54	0,49	0,51
BART-large 300K (U)	0,09	0,45	0,41	0,43
BART-large 300K (U, AP)	0,12	0,51	0,45	0,48
BART-large 300K (V)	0,10	0,47	0,42	0,44
BART-large 300K (V, AP)	0,14	0,51	0,48	0,50
BART-large 300K (V, AP)$_5$	0,14	0,53	0,48	0,51

Tabelle B.5: Übersicht von ROUGE-Scores für auf `msmarco-passage` trainierte T5-Modelle. Angegeben sind Precision (P), Recall (R) und F1-Score (F1). Zusätzliche Kennzeichnungen sind vor- und unverarbeitete Anchortexte (V/U), die Konkatenation von Anchortexten mit Passagentexten (AP) und `summarize` als Eingabebefehl (*). Alle Modelle wurden für zwei, fünf (5) oder einhundert (100) Epochen trainiert.

Ansatz	**ROUGE-1**			**ROUGE-2**			**ROUGE-L**		
	P	R	F1	P	R	F1	P	R	F1
T5-base 300K (U)	0,32	0,32	0,30	0,11	0,11	0,11	0,30	0,30	0,29
T5-base 300K (U, AP)	0,39	0,39	0,37	0,15	0,15	0,15	0,37	0,37	0,35
T5-base 300K (V)	0,32	0,32	0,31	0,11	0,11	0,11	0,31	0,31	0,29
T5-base 300K (V, AP)	0,41	0,40	0,39	0,17	0,17	0,16	0,39	0,38	0,37
T5-base 300K (V, AP)*	0,40	0,40	0,38	0,16	0,16	0,16	0,38	0,38	0,37
T5-base 300K (V, AP)$_5$	0,40	0,39	0,38	0,16	0,16	0,15	0,38	0,38	0,36
T5-base 500K (U, AP)	0,41	0,40	0,39	0,18	0,17	0,17	0,39	0,38	0,37
T5-base 500K (V)	0,34	0,33	0,32	0,13	0,13	0,12	0,32	0,32	0,31
T5-base 500K (V, AP)$_{100}$	0,41	0,41	0,39	0,17	0,17	0,16	0,40	0,39	0,38
T5-base 500K (V, AP)	0,43	0,42	0,41	0,18	0,18	0,17	0,41	0,40	0,39
T5-base 500K (V, AP)*	0,42	0,41	0,40	0,18	0,17	0,17	0,40	0,39	0,38
T5-base 500K (V, AP)$_5$	0,42	0,41	0,40	0,18	0,17	0,17	0,40	0,39	0,38
T5-large 300K (U)	0,32	0,32	0,30	0,12	0,12	0,11	0,31	0,30	0,29
T5-large 300K (U, AP)	0,40	0,38	0,37	0,17	0,16	0,16	0,38	0,36	0,36
T5-large 300K (V)	0,32	0,32	0,31	0,12	0,12	0,11	0,31	0,31	0,29
T5-large 300K (V, AP)	0,41	0,40	0,39	0,17	0,16	0,16	0,39	0,38	0,37
T5-large 300K (V, AP)*	0,40	0,40	0,38	0,16	0,16	0,16	0,38	0,38	0,37
T5-large 300K (V, AP)$_5$	0,38	0,38	0,36	0,15	0,15	0,14	0,36	0,36	0,34

Tabelle B.6: Übersicht der BLEU-Scores und BERTScores für auf `msmarco-passage` trainierte T5-Modelle. Angegeben sind Precision (P), Recall (R) und F1-Score (F1). Zusätzliche Kennzeichungen sind vor- und unverarbeitete Anchortexte (V/U), die Konkatenation von Anchortexten mit Passagentexten (AP) und `summarize` als Eingabebefehl (*). Alle Modelle wurden für zwei, fünf (5) oder einhundert (100) Epochen trainiert.

Ansatz	**BLEU**	**BERTScore**		
		P	R	F1
T5-base 300K (U)	0,06	0,37	0,36	0,36
T5-base 300K (U, AP)	0,09	0,43	0,43	0,43
T5-base 300K (V)	0,06	0,38	0,36	0,37
T5-base 300K (V, AP)	0,10	0,45	0,44	0,44
T5-base 300K (V, AP)*	0,10	0,44	0,43	0,44
T5-base 300K (V, AP)$_5$	0,09	0,44	0,43	0,43
T5-base 500K (U, AP)	0,11	0,46	0,44	0,45
T5-base 500K (V)	0,07	0,39	0,37	0,38
T5-base 500K (V, AP)$_{100}$	0,10	0,46	0,44	0,45
T5-base 500K (V, AP)	0,11	0,47	0,46	0,46
T5-base 500K (V, AP)*	0,11	0,46	0,45	0,46
T5-base 500K (V, AP)$_5$	0,10	0,46	0,44	0,45
T5-large 300K (U)	0,07	0,37	0,36	0,36
T5-large 300K (U, AP)	0,09	0,44	0,42	0,43
T5-large 300K (V)	0,07	0,37	0,36	0,37
T5-large 300K (V, AP)	0,10	0,45	0,44	0,44
T5-large 300K (V, AP)*	0,10	0,44	0,43	0,44
T5-large 300K (V, AP)$_5$	0,08	0,41	0,41	0,41

Tabelle B.7: Übersicht von ROUGE-Scores für auf `msmarco-passage` trainierte D2Q-Modelle. Angegeben sind Precision (P), Recall (R) und F1-Score (F1). Zusätzliche Kennzeichnungen sind vor- und unverarbeitete Anchortexte (V/U), die Konkatenation von Anchortexten mit Passagentexten (AP) und `summarize` als Eingabebefehl (*). Alle Modelle wurden für zwei, fünf (5) oder einhundert (100) Epochen trainiert.

Ansatz	**ROUGE-1**			**ROUGE-2**			**ROUGE-L**		
	P	R	F1	P	R	F1	P	R	F1
D2Q-base 300K (U)	0,34	0,34	0,33	0,13	0,12	0,12	0,33	0,32	0,31
D2Q-base 300K (U, AP)	0,39	0,39	0,37	0,16	0,16	0,15	0,37	0,37	0,35
D2Q-base 300K (V)	0,34	0,34	0,33	0,13	0,13	0,12	0,33	0,32	0,31
D2Q-base 300K (V, AP)	0,41	0,40	0,39	0,17	0,17	0,16	0,39	0,38	0,37
D2Q-base 300K (V, AP)*	0,42	0,41	0,40	0,18	0,17	0,17	0,40	0,39	0,38
D2Q-base 300K (V, AP)$_5$	0,41	0,41	0,40	0,17	0,17	0,17	0,39	0,39	0,38
D2Q-base 500K (U, AP)	0,41	0,40	0,39	0,18	0,17	0,17	0,39	0,38	0,37
D2Q-base 500K (V)	0,34	0,33	0,32	0,13	0,13	0,12	0,33	0,32	0,31
D2Q-base 500K (V, AP)$_{100}$	0,41	0,40	0,39	0,18	0,17	0,17	0,39	0,38	0,37
D2Q-base 500K (V, AP)	0,41	0,41	0,39	0,18	0,17	0,17	0,39	0,39	0,37
D2Q-base 500K (V, AP)*	0,42	0,41	0,40	0,18	0,18	0,17	0,40	0,40	0,38
D2Q-base 500K (V, AP)$_5$	0,41	0,41	0,40	0,18	0,17	0,17	0,40	0,39	0,38

Tabelle B.8: Übersicht der BLEU-Scores und BERTScores für auf `msmarco-passage` trainierte D2Q-Modelle. Angegeben sind Precision (P), Recall (R) und F1-Score (F1). Zusätzliche Kennzeichnungen sind vor- und unverarbeitete Anchortexte (V/U), die Konkatenation von Anchortexten mit Passagentexten (AP) und `summarize` als Eingabebefehl (*). Alle Modelle wurden für zwei, fünf (5) oder einhundert (100) Epochen trainiert.

Ansatz	**BLEU**	**BERTScore**		
		P	R	F1
D2Q-base 300K (U)	0,07	0,40	0,38	0,39
D2Q-base 300K (U, AP)	0,09	0,44	0,43	0,43
D2Q-base 300K (V)	0,07	0,40	0,38	0,39
D2Q-base 300K (V, AP)	0,10	0,46	0,44	0,45
D2Q-base 300K (V, AP)*	0,11	0,46	0,45	0,45
D2Q-base 300K $(V, AP)_5$	0,11	0,46	0,44	0,45
D2Q-base 500K (U, AP)	0,10	0,45	0,43	0,44
D2Q-base 500K (V)	0,07	0,39	0,38	0,38
D2Q-base 500K $(V, AP)_{100}$	0,10	0,45	0,44	0,45
D2Q-base 500K (V, AP)	0,11	0,45	0,44	0,45
D2Q-base 500K (V, AP)*	0,11	0,46	0,45	0,45
D2Q-base 500K $(V, AP)_5$	0,10	0,46	0,45	0,46

Anhang C

Ergebnisstabellen `clueweb09`

Tabelle C.1: Retrievalscores für BM25-Retrieval auf `clueweb09/trec-web-2009`. Indexiert werden die generierten Queries der `msmarco-document` (1000K) und `msmarco-passage` (300K) Modelle. Gekennzeichnet sind vor- und unverarbeitete Anchortexte (U/V), die Hinzunahme von Dokumententexten (AD) und `summarize` als Eingabebefehl (*).

Modell	**mAP**	**MRR**	**P@5**	**nDCG@10**	**R@1000**
T5-base 300K (U, AD)	0,03	0,28	0,18	0,10	0,13
T5-base 300K (V, AD)*	0,03	0,23	0,16	0,10	0,14
T5-base 300K (V, AD)	0,03	0,26	0,16	0,10	0,13
BERT-base 1000K (U)	0,07	0,58	0,45	0,32	0,12
BERT-base 1000K (U, AD)	0,09	0,69	0,50	0,34	0,13
BERT-base 1000K (V)	0,08	0,63	0,49	0,34	0,12
BERT-base 1000K (V, AD)	0,09	0,57	0,47	0,33	0,14
BART-base 1000K (U)	0,07	0,61	0,46	0,31	0,11
BART-base 1000K (U, AD)	0,08	0,65	0,45	0,33	0,12
BART-base 1000K (V)	0,07	0,63	0,47	0,31	0,11
BART-base 1000K (V, AD)	0,08	0,61	0,49	0,33	0,12
T5-base 1000K (U)	0,08	0,63	0,46	0,33	0,13
T5-base 1000K (U, AD))	0,09	0,62	0,46	0,33	0,14
T5-base 1000K (V)	0,08	0,62	0,46	0,36	0,13
T5-base 1000K (V, AD)	0,09	0,62	0,46	0,32	0,14

Literaturverzeichnis

[AMEL11] Nima Asadi, Donald Metzler, Tamer Elsayed, and Jimmy Lin. Pseudo test collections for learning web search ranking functions. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*, pages 1073–1082. ACM, 2011. 2

[ARTL19] Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. DocBERT: BERT for document classification. *arXiv*, abs/1904.08398, 2019. 4.2

[BBF+14] Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Ales Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation, WMT@ACL 2014, June 26-27, 2014, Baltimore, Maryland, USA*, pages 12–58. The Association for Computer Linguistics, 2014. 4.5

[BCB15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 2

[BCH03] Peter Bailey, Nick Craswell, and David Hawking. Engineering a multi-purpose test collection for web retrieval experiments. *Information Processing & Management*, 39(6):853–871, 2003. 2

[BCS06] Stefan Büttcher, Charles L. A. Clarke, and Ian Soboroff. The TREC 2006 terabyte track. In *Proceedings of the Fifteenth Text REtrieval Conference, TREC 2006, Gaithersburg, Maryland,*

USA, November 14-17, 2006, volume 500-272 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2006. 2

[BCYS07] Stefan Büttcher, Charles L. A. Clarke, Peter C. K. Yeung, and Ian Soboroff. Reliable information retrieval evaluation with incomplete and biased judgements. In *SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007*, pages 63–70. ACM, 2007. 2

[BHM04] Ricardo A. Baeza-Yates, Carlos A. Hurtado, and Marcelo Mendoza. Query recommendation using query logs in search engines. In *Current Trends in Database Technology - EDBT 2004 Workshops, EDBT 2004 Workshops PhD, DataX, PIM, P2P&DB, and ClustWeb, Heraklion, Crete, Greece, March 14-18, 2004, Revised Selected Papers*, volume 3268 of *Lecture Notes in Computer Science*, pages 588–596. Springer, 2004. 1

[BJR08] Cory Barr, Rosie Jones, and Moira Regelson. The linguistic structure of english web-search queries. In *2008 Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, Proceedings of the Conference, 25-27 October 2008, Honolulu, Hawaii, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1021–1030. ACL, 2008. 3.2

[BMR+20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 2

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998. 1, 2, 2

[BPC20] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv*, abs/2004.05150, 2020. 5.2

[Bro02] Andrei Z. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002. 2, 3.2

[BTdRM12] Richard Berendsen, Manos Tsagkias, Maarten de Rijke, and Edgar Meij. Generating pseudo test collections for learning to rank scientific articles. In *Information Access Evaluation. Multilinguality, Multimodality, and Visual Analytics - Third International Conference of the CLEF Initiative, CLEF 2012, Rome, Italy, September 17-20, 2012. Proceedings*, volume 7488 of *Lecture Notes in Computer Science*, pages 42–53. Springer, 2012. 2

[BTWdR13] Richard Berendsen, Manos Tsagkias, Wouter Weerkamp, and Maarten de Rijke. Pseudo test collections for training and tuning microblog rankers. In *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013*, pages 53–62. ACM, 2013. 2

[CB08] Suruchi Chawla and Punam Bedi. Improving information retrieval precision by finding related queries with similar information need using information scent. In *First International Conference on Emerging Trends in Engineering and Technology, ICETET '08, Nagpur, Maharashtra, India, July 16-18, 2008*, pages 486–491. IEEE Computer Society, 2008. 1

[CBFN13] Nick Craswell, Bodo Billerbeck, Dennis Fetterly, and Marc Najork. Robust query rewriting using anchor data. In *Sixth ACM International Conference on Web Search and Data Mining, WSDM 2013, Rome, Italy, February 4-8, 2013*, pages 335–344. ACM, 2013. 2

[CCM+20] Nick Craswell, Daniel Campos, Bhaskar Mitra, Emine Yilmaz, and Bodo Billerbeck. ORCAS: 18 million clicked query-document pairs for analyzing search. *arXiv*, abs/2006.05324, 2020. 4.1, 4.1, 4.4.1

[CCS09] Charles L. A. Clarke, Nick Craswell, and Ian Soboroff. Overview of the TREC 2009 web track. In *Proceedings of The Eighteenth Text REtrieval Conference, TREC 2009, Gaithersburg, Maryland,*

USA, November 17-20, 2009, volume 500-278 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2009. 1, 2, 4.1, 4.1, 4.3

[CCSC10] Charles L. A. Clarke, Nick Craswell, Ian Soboroff, and Gordon V. Cormack. Overview of the TREC 2010 web track. In *Proceedings of The Nineteenth Text REtrieval Conference, TREC 2010, Gaithersburg, Maryland, USA, November 16-19, 2010*, volume 500-294 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2010. 4.1

[CCSV11] Charles L. A. Clarke, Nick Craswell, Ian Soboroff, and Ellen M. Voorhees. Overview of the TREC 2011 web track. In *Proceedings of The Twentieth Text REtrieval Conference, TREC 2011, Gaithersburg, Maryland, USA, November 15-18, 2011*, volume 500-296 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2011. 4.1

[CCV12] Charles L. A. Clarke, Nick Craswell, and Ellen M. Voorhees. Overview of the TREC 2012 web track. In *Proceedings of The Twenty-First Text REtrieval Conference, TREC 2012, Gaithersburg, Maryland, USA, November 6-9, 2012*, volume 500-298 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2012. 4.1

[CGCB14] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv*, abs/1412.3555, 2014. 2

[CGH+06] Junghoo Cho, Hector Garcia-Molina, Taher H. Haveliwala, Wang Lam, Andreas Paepcke, Sriram Raghavan, and Gary Wesley. Stanford WebBase components and applications. *ACM Trans. Internet Techn.*, 6(2):153–186, 2006. 2

[CH02] Nick Craswell and David Hawking. Overview of the TREC-2002 web track. In *Proceedings of The Eleventh Text REtrieval Conference, TREC 2002, Gaithersburg, Maryland, USA, November 19-22, 2002*, volume 500-251 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2002. 2

[CHR01] Nick Craswell, David Hawking, and Stephen E. Robertson. Effective site finding using link anchor information. In *SIGIR 2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,*

September 9-13, 2001, New Orleans, Louisiana, USA, pages 250–257. ACM, 2001. 2

[CJ07] Ben Carterette and Rosie Jones. Evaluating search engines by modeling the relationship between relevance and clicks. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 217–224. Curran Associates, Inc., 2007. 2

[CMY+20] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. Overview of the TREC 2019 deep learning track. *arXiv*, abs/2003.07820, 2020. 1, 4.1, 4.3, 4.3, 4.3

[CO98] Kerry Coffman and Andrew M. Odlyzko. The size and growth rate of the Internet. *First Monday*, 3(10), 1998. 1

[COK06] Chris Callison-Burch, Miles Osborne, and Philipp Koehn. Re-evaluation the role of BLEU in machine translation research. In *EACL 2006, 11st Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference, April 3-7, 2006, Trento, Italy*. The Association for Computer Linguistics, 2006. 4.3

[CvMBB14] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*, pages 103–111. Association for Computational Linguistics, 2014. 2

[CvMG+14] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *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*, pages 1724–1734. ACL, 2014. 2

[CWNM02] Hang Cui, Ji-Rong Wen, Jian-Yun Nie, and Wei-Ying Ma. Probabilistic query expansion using query logs. In *Proceedings of*

the Eleventh International World Wide Web Conference, WWW 2002, May 7-11, 2002, Honolulu, Hawaii, USA, pages 325–332. ACM, 2002. 1

[DC10] Van Dang and W. Bruce Croft. Query reformulation using anchor text. In *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*, pages 41–50. ACM, 2010. 2

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv*, abs/1810.04805, 2018. 2, 2, 3.1, 3.3, 4.2, 4.2, 4.2, 4.3

[DD10] Na Dai and Brian D. Davison. Mining anchor text trends for retrieval. In *Advances in information retrieval, 32nd european conference on IR research, ECIR 2010, Milton Keynes, UK, March 28-31, 2010. Proceedings*, volume 5993 of *Lecture Notes in Computer Science*, pages 127–139. Springer, 2010. 2

[DVRC17] Laura Dietz, Manisha Verma, Filip Radlinski, and Nick Craswell. TREC complex answer retrieval overview. In *Proceedings of The Twenty-Sixth Text REtrieval Conference, TREC 2017, Gaithersburg, Maryland, USA, November 15-17, 2017*, volume 500-324 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2017. 4.2, 4.5

[DXX18] Linhao Dong, Shuang Xu, and Bo Xu. Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*, pages 5884–5888. IEEE, 2018. 4.2

[EM03] Nadav Eiron and Kevin S. McCurley. Analysis of anchor text for web search. In *SIGIR 2003: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 28 - August 1, 2003, Toronto, Canada*, pages 459–460. ACM, 2003. 1, 2, 2, 3.2, 3.2, 4.5

[FGP+22] Maik Fröbe, Sebastian Günther, Maximilian Probst, Martin Potthast, and Matthias Hagen. The power of anchor text in the neural retrieval era. In *Advances in Information Retrieval - 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway,*

April 10-14, 2022, Proceedings, Part I, volume 13185 of *Lecture Notes in Computer Science*, pages 567–583. Springer, 2022. 1, 2, 3.2, 3.2, 3.2, 4.1, 4.4.1, 4.5

[FKM+20] Alexander R. Fabbri, Wojciech Kryscinski, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir R. Radev. SummEval: Re-evaluating summarization evaluation. *arXiv*, abs/2007.12626, 2020. 4.3

[FLGD87] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971, 1987. 1, 2, 3.2

[FNLE19] Rodrigo Frassetto Nogueira, Jimmy Lin, and AI Epistemic. From doc2query to docTTTTTquery. *Online preprint*, 6, 2019. 1, 2, 3.2, 4.2, 4.2, 4.2, 4.3, 4.4.1, 5.1

[FRM+22] Markus Freitag, Ricardo Rei, Nitika Mathur, Chi-kiu Lo, Craig Stewart, Eleftherios Avramidis, Tom Kocmi, George F. Foster, Alon Lavie, and André F. T. Martins. Results of WMT22 metrics shared task: Stop using BLEU - neural metrics are better and more robust. In *Proceedings of the Seventh Conference on Machine Translation, WMT 2022, Abu Dhabi, United Arab Emirates (Hybrid), December 7-8, 2022*, pages 46–68. Association for Computational Linguistics, 2022. 4.3

[GCS22] Sebastian Gehrmann, Elizabeth Clark, and Thibault Sellam. Repairing the cracked foundation: A survey of obstacles in evaluation practices for generated text. *arXiv*, abs/2202.06935, 2022. 4.3

[GFGS06] Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of *ACM International Conference Proceeding Series*, pages 369–376. ACM, 2006. 2

[GG19] Shivani Gupta and Atul Gupta. Dealing with noise problem in machine learning data-sets: A systematic review. *Procedia Computer Science*, 161:466–474, 2019. 4.4.1

[GM22] Prashansa Gupta and Sean MacAvaney. On survivorship bias in MS MARCO. *arXiv*, abs/2204.12852, 2022. 1

[GMM23] Mitko Gospodinov, Sean MacAvaney, and Craig Macdonald. Doc2Query: When less is more. In *Advances in Information Retrieval - 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2-6, 2023, Proceedings, Part II*, volume 13981 of *Lecture Notes in Computer Science*, pages 414–422. Springer, 2023. 2, 4.9, 4.5

[GS04] Cathal Gurrin and Alan F. Smeaton. Replicating web structure in small-scale test collections. *Information Retrieval Journal*, 7(3-4):239–263, 2004. 2

[Har92a] Donna Harman. Evaluation issues in information retrieval. *Information Processing and Management*, 28(4):439 – –440, 1992. 4.3

[Har92b] Donna Harman. Overview of the first text retrieval conference (TREC-1). In *Proceedings of The First Text REtrieval Conference, TREC 1992, Gaithersburg, Maryland, USA, November 4-6, 1992*, volume 500-207 of *NIST Special Publication*, pages 1–20. National Institute of Standards and Technology (NIST), 1992. 4.3, 4.3

[Haw00] David Hawking. Overview of the TREC-9 web track. In *Proceedings of The Ninth Text REtrieval Conference, TREC 2000, Gaithersburg, Maryland, USA, November 13-16, 2000*, volume 500-249 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2000. 2

[HB21] Michael Hanna and Ondrej Bojar. A fine-grained analysis of BERTScore. In *Proceedings of the Sixth Conference on Machine Translation, WMT@EMNLP 2021, Online Event, November 10-11, 2021*, pages 507–517. Association for Computational Linguistics, 2021. 4.3

[HBCC19] Andrew Hoang, Antoine Bosselut, Asli Celikyilmaz, and Yejin Choi. Efficient adaptation of pretrained transformers for abstractive summarization. *arXiv*, abs/1906.00138, 2019. 3.3

[HC02] David Hawking and Nick Craswell. Overview of the TREC-2001 web track. *Nist Special Publication Sp*, (250):61–67, 2002. 2

[HC05] David Hawking and Nick Craswell. The very large collection and web tracks. In *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press, 2005. 2

[HCCU04] David Hawking, Francis Crimmins, Nick Craswell, and Trystan Upstill. How valuable is external link evidence when searching enterprise webs? In *Database Technologies 2004, Proceedings of the Fifteenth Australasian Database Conference, ADC 2004, Dunedin, New Zealand, 18-22 January 2004*, volume 27 of *CRPIT*, pages 77–84. Australian Computer Society, 2004. 2

[Hie10] Djoerd Hiemstra. ClueWeb09_anchors (anchor text derived from CMU's ClueWeb09 web crawl). *Technical Report TR-CTIT-10-15, Centre for Telematics and Information Technology University of Twente, Enschede. ISSN 1381-3625*, 2010. 4.1

[HLX15] Jianglin Huang, Yan-Fu Li, and Min Xie. An empirical analysis of data preprocessing for machine learning-based software cost estimation. *Information and Software Technology*, 67:108–127, 2015. 4.4.1

[HR18] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 328–339. Association for Computational Linguistics, 2018. 2

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 2, 3.1, 4.2

[HTO+16] Yunlong He, Jiliang Tang, Hua Ouyang, Changsung Kang, Dawei Yin, and Yi Chang. Learning to rewrite queries. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA,October 24-28, 2016*, pages 1443–1452. ACM, 2016. 1

[HVCB99] David Hawking, Ellen M. Voorhees, Nick Craswell, and Peter Bailey. Overview of the TREC-8 web track. In *Proceedings of The Eighth Text REtrieval Conference, TREC 1999, Gaithersburg, Maryland, USA, November 17-19, 1999*, volume 500-246 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 1999. 2

[HWBN20] Shuguang Han, Xuanhui Wang, Mike Bendersky, and Marc Najork. Learning-to-rank with BERT in tf-ranking. *arXiv*, abs/2004.08476, 2020. 1

[IHD+10] Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, 9-11 October 2010, MIT Stata Center, Massachusetts, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 944–952. ACL, 2010. 4.3

[JBS08] Bernard J. Jansen, Danielle L. Booth, and Amanda Spink. Determining the informational, navigational, and transactional intent of web queries. *Information Processsing & Management*, 44(3):1251–1266, 2008. 3.2

[JGP+05] Thorsten Joachims, Laura A. Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005*, pages 154–161. ACM, 2005. 2

[JHZ02] Rong Jin, Alexander G. Hauptmann, and ChengXiang Zhai. Title language model for information retrieval. In *SIGIR 2002: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 11-15, 2002, Tampere, Finland*, pages 42–48. ACM, 2002. 3.2, 4.5

[JK00] Kalervo Järvelin and Jaana Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *SIGIR 2000: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 24-28, 2000, Athens, Greece*, pages 41–48. ACM, 2000. 4.3

[JK02] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions of Information Systems*, 20(4):422–446, 2002. 4.3, 4.3

[Joa02] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July*

23-26, 2002, Edmonton, Alberta, Canada, pages 133–142. ACM, 2002. 2

[JP01] Bernard J. Jansen and Udo W. Pooch. A review of web searching studies and a framework for future research. *Journal of the Association for Information Science and Technology*, 52(3):235–246, 2001. 4.5

[KK10] Marijn Koolen and Jaap Kamps. The importance of anchor text for ad hoc search revisited. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010*, pages 122–129. ACM, 2010. 2

[KKM+19] Wojciech Kryscinski, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. Neural text summarization: A critical evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 540–551. Association for Computational Linguistics, 2019. 4.3

[KKP06] Sotiris B Kotsiantis, Dimitris Kanellopoulos, and Panagiotis E Pintelas. Data preprocessing for supervised learning. *International journal of computer science*, 1(2):111–117, 2006. 4.4.1

[KOM+20] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *arXiv*, abs/2004.04906, 2020. 1

[KT00] Mei Kobayashi and Koichi Takeda. Information retrieval on the web. *ACM Computing Surveys*, 32(2):144–173, 2000. 1

[KV00] Paul B. Kantor and Ellen M. Voorhees. The TREC-5 confusion track: Comparing retrieval methods for scanned text. *Information Retrieval Journal*, 2(2/3):165–176, 2000. 4.3, 4.3

[KW00] Wessel Kraaij and Thijs Westerveld. TNO-UT at TREC-9: How different are web documents? In *Proceedings of The Ninth Text REtrieval Conference, TREC 2000, Gaithersburg, Maryland,*

USA, November 13-16, 2000, volume 500-249 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2000. 2

[KZ04] Reiner Kraft and Jason Y. Zien. Mining anchor text for query refinement. In *Proceedings of the 13th international conference on World Wide Web, WWW 2004, New York, NY, USA, May 17-20, 2004*, pages 666–674. ACM, 2004. 2

[Lew23] Dirk Lewandowski. *The search engine market*, pages 165–173. Springer International Publishing, 2023. 1

[Lin04] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, july 2004. Association for Computational Linguistics. 1, 4.3, 4.3

[LLG+20] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics, 2020. 2, 3.1, 3.3, 3.3, 4.2, 4.2, 4.2

[LOG+19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv*, abs/1907.11692, 2019. 2

[MBC20] Nitika Mathur, Timothy Baldwin, and Trevor Cohn. Tangled up in BLEU: Reevaluating the evaluation of automatic machine translation evaluation metrics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4984–4997. Association for Computational Linguistics, 2020. 4.3

[MBXS17] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6294–6305, 2017. 2

[MDX+21] Zhengyi Ma, Zhicheng Dou, Wei Xu, Xinyu Zhang, Hao Jiang, Zhao Cao, and Ji-Rong Wen. Pre-training for ad-hoc retrieval: (h)yperlink is also you need. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, pages 1212–1221. ACM, 2021. 2

[MRR13] Shiva Imani Moghadasi, Sri Devi Ravana, and Sudharshan N. Raman. Low-cost evaluation techniques for information retrieval systems: A review. *Journal of Informetrics*, 7(2):301–312, 2013. 2

[NA15] Jun-Ping Ng and Viktoria Abrecht. Better summarization evaluation with word embeddings for ROUGE. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1925–1930. The Association for Computational Linguistics, 2015. 4.3

[NC19] Rodrigo Frassetto Nogueira and Kyunghyun Cho. Passage reranking with BERT. *arXiv*, abs/1901.04085, 2019. 1, 4.2

[NJL20] Rodrigo Frassetto Nogueira, Zhiying Jiang, and Jimmy Lin. Document ranking with a pretrained sequence-to-sequence model. *arXiv*, abs/2003.06713, 2020. 1

[NRS+16] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016. 1, 2, 3.2, 4.1, 4.4.1, 5.1

[NYCL19] Rodrigo Frassetto Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with BERT. *arXiv*, abs/1910.14424, 2019. 1

[NYLC19] Rodrigo Frassetto Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. Document expansion by query prediction. *arXiv*, abs/1904.08375, 2019. 1, 2, 3.2, 3.2, 4.2, 4.2, 4.2, 4.2, 4.3, 4.4.1, 5.1

[NZdS+16] Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 280–290. ACL, 2016. 4.5

[Odl03] Andrew M Odlyzko. Internet traffic growth: Sources and implications. In *Optical transmission systems and equipment for WDM networking II*, volume 5247, pages 1–15. SPIE, 2003. 1

[Ope23] OpenAI. GPT-4 technical report. *arXiv*, abs/2303.08774, 2023. 5.2

[PBMW99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999. 3.2

[PNI+18] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics, 2018. 2

[Pos18] Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers, WMT 2018, Belgium, Brussels, October 31 - November 1, 2018*, pages 186–191. Association for Computational Linguistics, 2018. 1, 4.3

[PRB+20] Amandalynne Paullada, Inioluwa Deborah Raji, Emily M. Bender, Emily Denton, and Alex Hanna. Data and its (dis)contents: A survey of dataset development and use in machine learning research. *arXiv*, abs/2012.05345, 2020. 1

[PRS+17] Rohit Prabhavalkar, Kanishka Rao, Tara N. Sainath, Bo Li, Leif Johnson, and Navdeep Jaitly. A comparison of sequence-to-sequence models for speech recognition. In *Interspeech 2017, 18th Annual Conference of the International SpeechcCommunication Association, Stockholm, Sweden, August 20-24, 2017*, pages 939–943. ISCA, 2017. 2

[PRWZ02] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL, 2002. 1, 4.3, 4.3

[PVU+18] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4052–4061. PMLR, 2018. 4.2

[RHW86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. *Learning internal representations by error propagation*, page 318–362. MIT Press, Cambridge, MA, USA, 1986. 2, 3.1

[RNS+18] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. *OpenAI blog*, 2018. 2

[Ros58] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. 2

[RSR+20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:140:1–140:67, 2020. 1, 2, 3.1, 3.3, 4.2, 4.2, 4.2, 4.3

[RWC+19] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 2

[SHMM99] Craig Silverstein, Monika Rauch Henzinger, Hannes Marais, and Michael Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999. 4.5

[SMAS17] Saeedeh Shekarpour, Edgard Marx, Sören Auer, and Amit P. Sheth. RQUERY: Rewriting natural language queries on knowledge

graphs to alleviate the vocabulary mismatch problem. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 3936–3943. AAAI Press, 2017. 1

[SQXH19] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune BERT for text classification? In *Chinese Computational Linguistics - 18th China National Conference, CCL 2019, Kunming, China, October 18-20, 2019, Proceedings*, volume 11856 of *Lecture Notes in Computer Science*, pages 194–206. Springer, 2019. 4.2

[SS11] Adel A. M. Saleh and Jane M. Simmons. Technology and architecture to enable the explosive growth of the Internet. *IEEE Communications Magazine*, 49(1):126–132, 2011. 1

[SSRB17] Hasim Sak, Matt Shannon, Kanishka Rao, and Françoise Beaufays. Recurrent neural aligner: An encoder-decoder neural network model for sequence-to-sequence mapping. In *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*, pages 1298–1302. ISCA, 2017. 2

[STS11] Falk Scholer, Andrew Turpin, and Mark Sanderson. Quantifying test collection quality based on the consistency of relevance judgements. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*, pages 1063–1072. ACM, 2011. 2

[SUV18] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 464–468. Association for Computational Linguistics, 2018. 4.2

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence-to-sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, volume abs/1409.3215, page 3104–3112, 2014. 2

[TRR+21] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. 2

[TZ11] Edison Tse and Wugang Zhao. Competition in search engine market. *Journal of Business Strategies*, 28(2):123–150, 2011. 1

[VR79] Cornelis J Van Rijsbergen. Information retrieval., 1979. 4.3, 4.3

[VRD+15] Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond J. Mooney, Trevor Darrell, and Kate Saenko. Sequence to sequence - video to text. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 4534–4542. IEEE Computer Society, 2015. 2

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. 2, 3.1, 4.2, 4.2, 4.2, 4.2

[WDS+20] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 38–45. Association for Computational Linguistics, 2020. 2, 3.1, 4.2

[WKH01] Thijs Westerveld, Wessel Kraaij, and Djoerd Hiemstra. Retrieving web pages using content, links, URLs and anchors. In *Proceedings of The Tenth Text REtrieval Conference, TREC 2001, Gaithersburg, Maryland, USA, November 13-16, 2001*, volume

500-250 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2001. 2

[WTRG21] Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. GPL: Generative pseudo labeling for unsupervised domain adaptation of dense retrieval. *arXiv*, abs/2112.07577, 2021. 2

[XLX23] Yiqing Xie, Xiao Liu, and Chenyan Xiong. Unsupervised dense retrieval training with web anchors. *arXiv*, abs/2305.05834, 2023. 2

[YA10] Xing Yi and James Allan. A content based approach for discovering missing anchor text for web search. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010*, pages 427–434. ACM, 2010. 2

[YKYS17] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of CNN and RNN for natural language processing. *arXiv*, abs/1702.01923, 2017. 2

[ZKW+19] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. BERTScore: Evaluating text generation with BERT. *arXiv*, abs/1904.09675, 2019. 1, 4.3, 4.3

[ZYL20] Xinyu Zhang, Andrew Yates, and Jimmy Lin. A little bit is worse than none: Ranking with limited training data. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing, SustaiNLP@EMNLP 2020, Online, November 20, 2020*, pages 107–112. Association for Computational Linguistics, 2020. 1, 2