Martin-Luther-Universität Halle-Wittenberg
Naturwissenschaftliche Fakultät III - Agrar- und Ernährungswissenschaften, Geowissenschaften und Informatik
Institut für Informatik
Studiengang Informatik

Analyse und Lösungsansätze zum Schließen des Curiosity Gaps bei Clickbait-Nachrichten

Masterarbeit

Artur Jurk
geb. am: 16.05.1996 in Nowokusnezk
Matrikelnummer 215203034

1. Gutachter: Prof. Dr. Matthias Hagen
2. Gutachter: Jun.-Prof. Dr. Martin Potthast

Datum der Abgabe: 8. September 2021

Zusammenfassung

In dieser Arbeit untersuchen wir Ansätze zum Schließen des *Curiosity Gaps* bei Clickbait-Nachrichten (also zum ‚Spoilern' von Clickbait). Eine Clickbait-Nachricht ist in unserem Kontext meist ein Satz, der den Leser dazu manipuliert, einem Link zu folgen. Aus dem vom Weblink verlinkten Text soll anhand der Clickbait-Nachricht ein Spoiler extrahiert werden, sodass ein Leser der Clickbait-Nachricht und dessen Spoiler keinen Curiosity Gap mehr verspürt. Dazu erweitern wir den Datensatz Webis-Clickbait-19 auf insgesamt 5,000 Einträge. Zusätzlich annotieren wir, ob ein Spoiler kurz (factoid) oder lang (beschreibend) sein sollte. Auf diesem Datensatz trainieren wir einen Logistic-Regression-Klassifikator, der mit einer Accuracy von 74.9% vorhersagt, ob ein Spoiler zu einer gegebenen Clickbait-Nachricht kurz oder lang sein sollte. Die Unterscheidung dieser Spoilerklassen ermöglicht es uns, ein Modell zum Erstellen kurzer Spoiler und ein weiteres Modell zum Erstellen langer Spoiler zu trainieren. Die jeweils besten Ergebnisse von 78.6 % METEOR-F_{mean} erzielen wir beim Spoilern kurzer Spoiler mit einem RoBERTa$_{large}$-Modell und von 43.7 % METEOR-F_{mean} beim Spoilern langer Spoiler mit einem DeBERTa$_{large}$-Modell.

Inhaltsverzeichnis

Danksagung

Mein Dank gilt zu allererst Herrn Prof. Dr. Matthias Hagen und Jun.-Prof. Dr. Martin Potthast dafür, dass sie mir die Chance gegeben haben, dieses Thema bearbeiten zu dürfen. Ich danke ihnen für die kontinuierlichen Anregungen, auch in Form von Kritik während meiner Bearbeitungszeit.

Weiterhin danke ich der Webis-Group für die Zusammenarbeit. Ohne deren leistungsstarker Hardware und Organisation wäre die Masterarbeit in diesem Rahmen nicht möglich gewesen. Ein besonderer Dank gilt hierbei Maik Fröbe und Michael Völske, die bei technischen Problemen stets zur Verfügung standen.

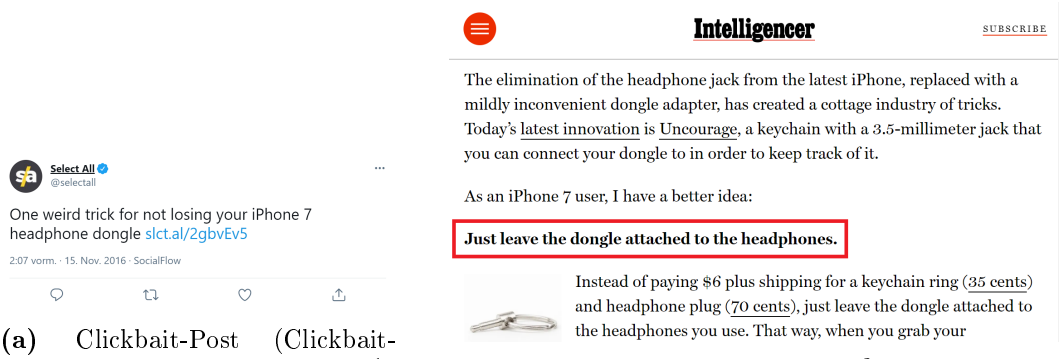
Ich danke auch meiner Freundin und meinen Freunden für das Prüfen meiner Schrift und das Geben aufschlussreicher Ideen und neuer Denkanstöße.

Kapitel 1

Einleitung

Clickbait ist heutzutage überall im Web zu finden. Vor allem auf Social-Media-Plattformen wie Facebook, Twitter oder Reddit sind Clickbait-Posts zu finden, die auf andere Webseiten aufmerksam machen. Gerne nutzen etwa Publisher der *The Washington Post* oder *Business Insider* die Social-Media-Plattformen, um ihre eigenen Webseiten zu bewerben. Das Problem ist: Clickbait manipuliert den Leser, eine Webseite zu besuchen. Clickbait-Nachrichten haben die Eigenschaft, einen *Curiosity Gap* (Wissenslücke) beim Leser entstehen zu lassen. Beim Leser wird also die Neugier auf Information geweckt (Loewenstein 1994, Blom und Hansen 2015) und als Ausweg klickt dieser auf den beistehenden Link des Clickbait-Posts. In Abbildung 1 ist ein Clickbait-Post auf Twitter mit der Clickbait-Nachricht, gefolgt von einem Link und dem verlinkten Text zu sehen. Die Clickbait-Nachricht „**One weird trick for not losing your iPhone 7 headphone dongle**“ (Abbildung 1a) weckt die Neugier zu erfahren, welcher ‚eigenartige Trick‘ gemeint ist. Der Leser klickt dann wahrscheinlich auf den Link im Clickbait-Post und überfliegt den auf der Seite zu findenden Text. Dabei sucht er nach dem eigenartigen Trick, der in der Clickbait-Nachricht erwähnt wurde. Nach dem ersten Absatz, stößt er auf „**Just leave the dongle attached to the headphones**“ (Abbildung 1b). Der Leser verlässt die Webseite wieder, denn seine Neugier ist gestillt, aber er ist potenziell enttäuscht (Chakraborty et al. 2016). Den Dongle einfach an den Kopfhörern zu lassen, scheint kein ‚eigenartiger Trick‘ zu sein. Damit hat die Clickbait-Nachricht ihren Zweck erfüllt, den Leser auf die Webseite zu locken und einen Klick für Werbeeinnahmen zu generieren (Michael 2015).

Um den manipulativen Charakter von Clickbait abzuschwächen, entwerfen wir in dieser Arbeit einen Lösungsansatz zum automatisierten Spoilern von Clickbait. Der Lösungsansatz soll in der Lage sein, mit Hilfe der Clickbait-Nachricht



(a) Clickbait-Post (Clickbait-Nachricht mit Link) auf Twitter[1]

(b) Verlinkter Text[2]

Abbildung 1: (a) Ein Clickbait-Post (Clickbait-Nachricht mit Link) und (b) der verlinkte Text, der unter dem Link des Clickbait-Tweets (blau hervorgehoben) zu finden ist. Der entsprechende Spoiler ist im verlinkten Text rot umrandet.

und des verlinkten Textes, einen Spoiler aus dem verlinkten Text zu extrahieren. Der Spoiler stellt hierbei die Information dar, die den Curiosity Gap der Clickbait-Nachricht schließt (in Abbildung 1b rot umrandet zu sehen). Der Lösungsansatz umfasst einen Logistic-Regression-Klassifikator, der einen Spoiler als kurz (factoid) oder lang (beschreibend) vorhersagt, ein RoBERTa-Modell zum Extrahieren von kurzen und ein DeBERTa-Modell zum Extrahieren von langen Spoilern.

Für den neuen Datensatz Webis-Clickbait-Spoiling-20 erweitern wir den Datensatz Webis-Clickbait-19 aus der Bachelorarbeit von Jana Puschmann auf eine Menge von insgesamt 5000 Clickbait-Posts mit Spoilern (Kapitel 3). Dafür annotieren wir Clickbait-Posts mit Spoilern manuell nach, die im Crowdsourcing-Prozess von Puschmann ungespoilert blieben. Zusätzlich dazu annotieren wir auch Spoilerklassen für die Clickbait-Posts. Die Spoilerklassen geben an, ob der Spoiler für den entsprechenden Clickbait kurz, lang oder mehrteilig ist. Mit Hilfe der drei Spoilerklassen teilen wir den Prozess des Spoilerns in die drei Teilprobleme: Vorhersagen der Spoilerklasse mit einem Klassifikator, Extrahieren eines Spoilers für Clickbait mit kurzen Spoilern und Extrahieren eines Spoilers für Clickbait mit langen Spoilern. Die drei Teilprobleme sollen einzeln gelöst zu einem besseren Gesamtergebnis führen als das direkte Lösen des Gesamtproblems.

[1] https://twitter.com/selectall/status/798331539021033476

[2] https://nymag.com/intelligencer/2016/11/a-good-way-to-not-lose-your-headphone-dongle.html

Clickbait-Nachricht	Cleveland will acquire this star to play alongside LeBron in a blockbuster NBA trade:
Spoiler	Kevin Love

Abbildung 2: Beispiel einer Clickbait-Nachricht mit kurzem Spoiler.

Clickbait-Nachricht	Will you inherit less than you think?
Spoiler	The majority of those who had already inherited got less than $100,000

Abbildung 3: Beispiel einer Clickbait-Nachricht mit langem Spoiler.

Für das Vorhersagen der Spoilerklasse einer Clickbait-Nachricht, bauen wir einen Spoiler-Typ-Klassifikator. Dieser soll uns anhand der Clickbait-Nachricht und des verlinkten Textes vorausberechnen, ob der Spoiler kurz (Abbildung 2) oder lang (Abbildung 3) ist (Kapitel 4). Beim Bau des Spoiler-Typ-Klassifikator konzentrieren wir uns auf eine Repräsentation des Textes durch Häufigkeiten der Clickbait-Nachricht sowie des verlinkten Textes von Wort-N-Grammen und Part-Of-Speech-N-Grammen. Wir stellen fest, dass Textvorverarbeitung wie das Umwandeln in Kleinbuchstaben (Lowercasing) oder das Umwandeln der Wörter auf Grundform bzw. Infinitiv (Lemmatisierung) und Entfernen von aussageschwacher Wörter (Stoppwörtern) eine negative Auswirkung auf die Effektivität des Klassifikators hat. Das heißt, dass das Klassifizieren von Spoilern alle Informationen der Clickbait-Nachricht erfordert, die zur Verfügung stehen. Um zu testen, wie gut der Klassifikator im Endeffekt ist, lassen wir drei Personen mit Metainformationen zu den Spoilerklassen einen Test-Datensatz annotieren. Aus den Annotationen der drei Personen schließen wir, dass der Mensch mit einer Accuracy von 85.5% vorhersagen kann, ob der Spoiler kurz oder lang ist. Unser Klassifikator ist mit einer Accuracy von 74.9 % nicht weit davon entfernt, gilt in Zukunft aber als erstes zu verbessern.

Für das Erstellen von jeweils kurzen und langen Spoilern verfolgen wir die Intuition, dass sich Clickbait-Nachrichten Fragen ähneln. Clickbait-Nachrichten lassen sich als Fragen umformulieren (Abbildung 4), wodurch die Beziehung zwischen Clickbait-Nachricht und Spoiler der Beziehung zwischen Frage und Antwort sehr ähnlich erscheint. Also wollen wir das Spoilern von Clickbait-Nachrichten mit Ansätzen zum Beantworten von Fragen bearbeiten. Das

Clickbait-Nachricht	One weird trick for not losing your iPhone 7 headphone dongle
Frage	What is the one weird trick for not losing your iPhone 7 headphone dongle?

Abbildung 4: Umformulierung einer Clickbait-Nachricht zu einer Frage.

Beantworten von Fragen ist eine bereits bekannte Problemstellung auf dem Gebiet des Natural-Language-Processings und nennt sich *Question Answering*. Dafür wurden Datensätze wie SQuAD (Rajpurkar et al. 2016) und TriviaQA (Joshi et al. 2017) veröffentlicht, um verschiedene Verfahren entwickeln und vergleichen zu können. Die Aufgabe dabei ist es, die Antworten auf Fragen aus Texten, in denen die Antwort enthalten ist, zu extrahieren. In unserem Fall untersuchen wir, wie gut aktuelle Question-Answering-Ansätze Spoiler für Clickbait-Nachrichten aus dem verlinkten Text extrahieren können. Beispiele für Question-Answering-Ansätze sind auf großen Textdatensätzen vortrainierte Neuronale Netzwerke, sogenannte *Sprachverständnismodelle*. Das Nutzen solcher Sprachverständnismodellen wie BERT (Devlin et al. 2019) hat sich zum Erzielen von neuen State-Of-The-Art-Ergebnissen in verschiedenen Bereichen des Natural Language Processings wie *Text Classification* und *Question Answering* durchgesetzt. Aus diesem Grund testen wir neben BERT und ALBERT weitere Sprachverständnismodelle aus der Huggingface-Bibliothek[3] zum Extrahieren von kurzen und langen Spoilern.

Um bei der Evaluierung der Trainingsergebnisse einschätzen zu können, wie ähnlich der von den Modellen extrahierte Spoiler zum annotierten Spoiler ist, berechnen wir die drei Machine Translation Scores METEOR-F_{mean} (Banerjee und Lavie 2005), BLEU-4 (Papineni et al. 2002) und BERTScore-F1 (Zhang et al. 2020a) zwischen extrahiertem und annotiertem Spoiler. Wir binarisieren die Scores zusätzlich mittles eines Schwellwertes. Den Schwellwert wählen wir so, dass möglichst wenig falsch als als korrekt ausgewertete Spoiler einen größeren Score als der Schwellwert haben, wodurch wir relativ sicher sein können, dass diese Spoiler als richtig extrahiert gelten. Das gibt uns eine bessere Vorstellung darüber, wie viele Spoiler von den Modellen richtig extrahiert wurden. Mit Hilfe der drei Scores ermitteln wir schließlich das für uns beste Vorgehen beim Feintunen der Modelle.

[3] https://huggingface.co/transformers/

Kapitel 2

Grundlagen und verwandte Arbeiten

In diesem Kapitel belehren wir uns zu verwandten Arbeiten, bezogen auf unseren Lösungsansatz zum Spoilern von Clickbait. Zuallererst klären wir auf, was genau unter *Clickbait* zu verstehen ist in Abschnitt 2.1. Zum Thema Klassifikation von Clickbait informieren wir uns in Abschnitt 2.2 über das Problem der Text Classification im Allgemeinen. Da wir gezeigt haben, dass Clickbait-Nachrichten auch als Fragen formuliert werden können (Abbildung 4), befassen wir uns auch konkreter mit *Question Classification* in Abschnitt 2.3. Beim Spoilern von Clickbait wollen wir einen Spoiler aus einem entsprechenden Text extrahieren. Weil diese Vorgehensweise der Extraktion eines Textabschnittes ähnelt, schauen wir uns das Problem des *Passage Retrievals* in Abschnitt 2.4 an. Wie auch bei Text Classification existiert zur Extraktion von Textabschnitten bezogen auf Fragen eine konkreteres Problem: Dem sogenannten *Question Answering*. Diesem widmen wir uns in Abschnitt 2.5.

2.1 Clickbait

Unter einem Clickbait (Klickköder) verstehen wir einen Text von rund einem Satz an Länge, der durch das Auslassen von Informationen Aufmerksamkeit erregt und den Leser so dazu verleitet, einen verlinkten Text lesen zu wollen. Dabei wird das kognitive Phänomen des *Curiosity Gaps* ausgenutzt (Loewenstein, 1994). Der Clickbait ist oft so formuliert, dass die ausgelassene Information Neugier im Leser erzeugt, genau diese Information wissen zu wollen. Zum Beispiel ‚One weird trick for not losing your iPhone 7 headphone dongle' (Abbildung 1) gibt es etwa die Problematik, dass einige Leser das Problem ha-

ben, ihren iPhone 7 Kopfhörer-Dongle zu verlieren. Diese Leser sind generell an einer Lösung dieses Problems interessiert. Hinzu kommt, dass das Beispiel leicht übertrieben meint, es gäbe einen ‚weird' (eigenartigen) Trick, um das Problem zu lösen, wodurch falsche Neugier im Leser geweckt wird. So sind auch noch mehr Leser dazu hingeleitet den verlinkten Text lesen zu wollen, weil sie ihrer Neugier nicht widerstehen können und erfahren wollen, welche eigenartige Trick denn gemeint ist. Aber der ‚eigenartige' Trick im verlinkten Text ist, dass man den Kopfhörer-Dongle einfach am iPhone 7 stecken lassen soll, anstatt ihn immer wieder abzunehmen. Genauso wie in diesem Beispiel entspricht die Information, die im verlinkten Text zu finden ist, oft nicht den Erwartungen, die der Clickbait induziert (Chakraborty et al., 2016). Deshalb sprechen wir bei Clickbait vom manipulativen Charakter, der auch in weiteren Publikationen als negativ eingeschätzt wird (Blom und Hansen, 2015; Chen et al., 2015; Zannettou et al., 2019).

Einhergehend mit der fortschreitenden Digitalisierung bietet Clickbait eine große Chance für Publisher wie *The Washington Post*, *Business Insider* und *Spiegel* Umsatz zu generieren und relevant in der Nachrichtenbranche zu bleiben. Immer mehr Meschen halten sich im Web bzw. auf Social-Media-Plattformen, wie Facebook, Reddit und Twitter, auf und beziehen hauptsächlich von dort aktuelle Nachrichten (Newman, 2017). Dadurch ist es für Nachrichtenpublisher notwendig ihre Webpräsenz auszubauen. Umsatz wird im Web größtenteils durch Werbung oder Sponsoren generiert, welche beispielsweise durch festgelegte Summen pro Besucher auf der Webseite zahlen (Michael, 2015; Chakraborty et al., 2016). Somit brauchen Nachrichtenpublisher möglichst viel Aufmerksamkeit und Popularität, um genug oder möglichst viel Umsatz zu machen. An die notwendige Aufmerksamkeit und Popularität kommen die Publisher, indem sie ihre Texte auf Social-Media-Plattformen bewerben und nicht selten dabei Clickbait verwenden. Denn mit Clickbait werden die Nutzer gelockt, sodass sie die entsprechenden verlinkten Webseiten besuchen. Um mit Clickbait Leser auf eine Webseite zu locken, erfordet es keinen qualitativ hochwertigen Text auf der verlinkten Webseite und das ist ein kritischer Punkt. Durch Clickbait entwickelt sich ein Nachrichtenpublisher nicht zu einem journalistisch anerkannten Teil der Nachrichtenbranche, sondern eher zu einem verpönten (Dvorkin, 2016). Nur je sensationeller und übertriebener ein Clickbait formuliert ist desto mehr Einnahmen generiert er, den Text selbst lesen die meisten Leser nicht, weil sie nur den sensationalen Fakt (Spoiler) finden wollen. Deshalb ist es wichtig Clickbait in seinem manipulativen Charakter abzuschwächen, indem wir einen Lösungsansatz zum Schließen des Curiosity Gaps entwickeln.

2.2 Text Classification

Text Classification, auch Text Categorization genannt, als Aufgabengebiet des Natural Language Processings behandelt das Problem, Texte in Kategorien bzw. Klassen einzuordnen. Übliche Beispiele sind das Einordnen von Nachrichtentexten in übergeordnete Themen wie Politik, Gesundheit oder Sport; Unterscheidung, ob eine E-Mail Spam oder kein Spam ist oder die Analyse der Stimmung eines Textes (Yang und Joachims, 2008). Aber auch grundlegendere Aufgaben wie *Part of Speech Tagging* (Workartidentifizierung) gehören zum Aufgabengebiet der Text Classification. Einen Text können wir einerseits mit selbstgebauten Regeln, anhand von Expertenwissen und/oder statistischen Daten zu den Klassen von Texten, die erkannt werden sollen, klassifizieren und andererseits mit Hilfe von statistischen Daten und Algorithmen Machinellen Lernens klassifizieren.

Typische statistische Daten sind Worthäufigkeiten, durchschnittliche Wortlängen, Häufigkeiten von Wortpaaren oder Gleiches für Wortarten (POS). Bei Erhebung statistischer Daten werden häufig sogenannte *Stoppwörter* außer Betracht gelassen, weil sie zum Informationsgehalt des Textes wenig bis gar nichts beitragen und so Rechenaufwand eingespart wird. Zu Stoppwörtern gehören beispielsweise Artikel (der, ein, ...), Konjuktionen (und, doch, ...), häufig benutzte Präpositionen (an, in, von, ...) oder auch Fragewörter (wer, wie, ...). Bei Worthäufigkeiten wird oftmals statt die absolute Häufigkeit des Wortes im Text eher der *TF-IDF-Wert* eines Wortes genutzt. Der TF-IDF-Wert beschreibt, wie wichtig das Wort im entsprechenden Text im Vergleich zu anderen Texten ist. Die Idee dahinter ist, dass wichtige Wörter in einem Text vergleichsweise öfter vorkommen als in Texten, in denen sie weniger relevant sind. Weil Algorithmen des Maschinellen Lernens Texte nicht wie Menschen Wörter lesen können, müssen die Texte als Vektoren $x = (x^{(0)}, \ldots, x^{(p)})$ bestehend aus p numerischen Werten $x^{(i)}$ dargestellt werden. Dabei stellt jedes $x^{(i)}$ ein statistisches Datum des Textes dar, wie den TF-IDF-Wert eines Wortes, die Länge des Textes oder die Häufigkeit einer bestimmten Wortart (POS). Das statistische Datum $x^{(i)}$ wird deshalb *Feature* (Eigenschaft) genannt. Der Begriff *Bag-Of-Words-Repräsentation* wird hierbei genutzt, wenn der Text nur durch die Häufigkeiten von Wörtern repräsentiert wird. Zum Lernen, wie Texte klassifiziert werden, benötigen Algorithmen Maschinellen Lernens eine Sammlung an Texten zusammen mit deren Klassen (Datensatz). Auf einer solchen Sammlung rechnen die Algorithmen die Texte jeweils in einen Vektor aus Features um und versuchen anhand der Klassen Regeln zu lernen (Yang und Joachims, 2008), mit denen sie die Klassen unterscheiden können. Zu Algorithmen Maschinellen Lernens gehören beispielsweise *Naive Bayes*, *Decision Trees*, *Support Vector Machines* (SVM) oder *Neuronale Netz-*

werke. Am Beispiel von Naive Bayes wird zur Klassifikation der Satz von Bayes (Gleichung 1) aus der Wahrscheinlichkeitstheorie verwendet, um mit den $x^{(i)}$ die Wahrscheinlichkeit auszurechnen, mit der ein Text zu einer bestimmten Klasse C gehört.

$$P(A|B) \;=\; \frac{P(B|A) \cdot P(A)}{P(B)} \tag{1}$$

Naive Bayes setzt naiverweise voraus, dass alle $x^{(i)}$ unabhängig von einander sind (was in den meisten Fällen nicht zutrifft), sodass die Gleichung zur Berechnung der Wahrscheinlichkeit zu Gleichung 2 vereinfachbar ist. Trotzdem führt Naive Bayes oft zu guten Ergebnissen in der Klassifikation und wird erfolgreich zur Nachrichtenklassifikation, Sentimentanalyse und Spamerkennung genutzt (Yang und Joachims 2008).

$$P(C|\; x^{(0)}, \ldots, x^{(p)}) \;=\; P(C) \prod_{i=0}^{p} P(x^{(i)}|\; C) \tag{2}$$

2.3 Question Classification

Als Teilproblem von Text Classification befasst sich Question Classification nicht allgemein mit Text, sondern konkret mit Fragen und zählt zu den wesentlichsten Teilaufgaben des Question Answerings. Question Classification bestimmt anhand der Frage, welche Antwortklasse diese Frage erwartet. Antwortklassen sind beispielsweise Eigennamen, Zahlen oder Beschreibungen. Die Antwortklasse bestimmt die darauffolgende Suchstrategie des Algorithmus nach der Antwort im Text. Wenn die Frage eine Zahl als Antwort erwartet, kann gleich nach einer Zahl als Antwort gesucht werden. Das Problem bei Question Classification ist, dass die Menge an Wörtern, die zum Klassifizieren zur Verfügung stehen, oft nicht mehr als zwei Sätze beträgt (Abbildung 4). Dadurch reduziert sich die Menge an Informationen, die aus den Wörtern geschlossen werden kann, deutlich und Stoppwörter werden eher mit einbezogen, um jede mögliche Information zu bewahren. Vor allem Fragewörter (wer, was, wie, ...) sind dabei für die Klassifikation entscheidend und beispielsweise Präpositionen können essenzielle Informationen zu Zeit, Ort oder Grund in einer Frage besitzen (Abbildung 5).

Im Groben können Fragen in die Kategorien *einteilig* und *mehrteilig* und *factoid* und *deskriptiv* eingeordnet werden (Devopedia, 2020). Factoide Fragen können oftmals durch eine einzelne Entität, wie einem Eigennamen, beantwortet werden. Deskriptive Fragen dagegen erfordern eine Antwort von der Länge eines Satzes bis hin zu mehreren Paragraphen, weil die Frage nach einer

Informationsart	Beispiel
Ort	Was ist *unter* der Autobahn?
Zeit	Wer ist *während* des Brandes aufgetaucht?
Grund	Was hat er *infolge* seiner Verletzung nicht getan?

Abbildung 5: Beispiele für essenziele informationstragende Präpositionen.

Kategorien		Beispiel
Einteilig,	Factoid	Wann wurde die erste Autobahn gebaut?
Einteilig,	Deskriptiv	Was ist Schnee?
Mehrteilig		Welche Blumen wachsen höher als Tulpen?
Mehrteilig		Warum ist die Bannane krum und wann darf ich sie essen?

Abbildung 6: Beispiele für verschiedene Kategorien von Fragen.

beschreibenden Antwort sucht, wie einer Anleitung oder einer Definition. Einteilige Fragen suchen nach beispielsweise genau einem Fakt oder genau einer Definition. Mehrteilige Fragen benötigen mindestens zwei Fakten oder Definitionen zum Beantworten. Konkrete Beispiele für die genannten Kategorien sehen wir in Abbildung 6. Da aus den Wörtern einer Frage weniger Informationen geschlossen werden können, um Fragen kategorisieren zu können, steigt die Wichtigkeit, syntaktische und semantische Informationen zwischen den Wörter auszunutzen. So formulierten Zhang und Lee, 2003 einen *baumartigen Kernel* für SVMs, der syntaktische Informationen nutzt. Dafür wird anhand der Wort-N-Gramme eine Frage zu einem Syntaxbaum geparst und in einen Vektorraum abgebildet. Der Syntaxbaum in Abbildung 7 illustriert wie die Frage ‚Was ist ein Atom?' aus linguistischer Sicht aufgebaut ist (Bies et al., 1995). ‚Was ist ein Atom?' ist im Allgemeinen gesehen eine von einer W-Phrase eingeleitete direkte Frage (SBARQ). Die W-Phrase (WHNP) besteht im dem Fall aus einem einzelnen W-Pronomen (WP) ‚Was'. Der zweite Bestandteil der direkten Frage ist der Hauptsatz (SQ), der auf die W-Phrase folgt. In unserem Beispiel ist es eine Verbalphrase (VP), bestehend aus dem Hilfsverb (AUX) ‚ist' und einer Nominalphrase (NP): Die Nominalphrase unseres Beispiels ist aus dem Artikel (DT) ‚ein' und dem Nomen im Singular (NN) ‚Atom' aufgebaut. Abgeschlossen wird die Frage mit der Punktierung (■) ‚?'. Damit schafft die SVM mit baumartigem Kernel eine Accuracy von 90.0 % auf dem Datensatz *TREC-10*. Weitere Froscher nutzten *Transfer-Based-Learning* (TBL), um Support Vector Machines, welche jeweils auf Wörtern, Worteinbettungen, Wort-2-Grammen und der Abhängigkeitsstruktur der Wörter trainiert wurden, zu kombinieren

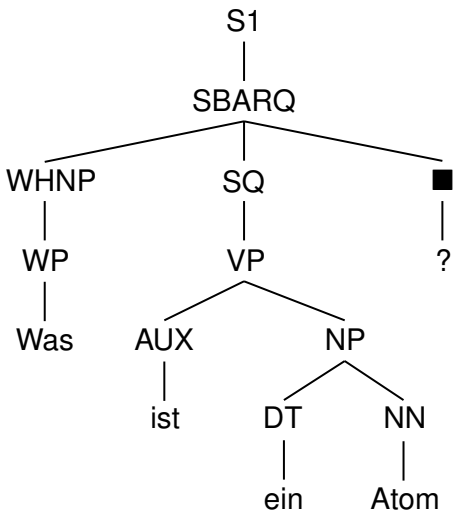


Abbildung 7: Syntaxbaum der Frage ‚Was ist ein Atom?' aus Zhang und Lee [67].

(Li et al., 2005). Damit erreichen Li et al. auf TREC-10 eine Accuracy von 91.6 %. In einer aktuelleren Arbeit stellten Tayyar Madabushi und Lee, 2016 das Konzept *syntaktischer Karten* für Fragen vor. Mit den syntaktischen Karten analysierten Tayyar Madabushi und Lee die Fragen von TREC-10 und bauten damit ein regelbasiertes System zum Klassifizieren der Fragen. Das System erreicht auf TREC-10 eine Accuracy von 97.2 %.

2.4 Passage Retrieval

Passage Retrieval ist eine Aufgabe des Natural Language Processings, einen oder mehrere Abschnitte eines oder mehrerer Texte zu extrahieren, die relevant bezüglich einer Not an Information (Anfrage) sind (Roberts und Gaizauskas, 2004). Passage Retrieval spielt vor allem im Aufgabengebiet des Question Answerings (Abschnitt 2.5) eine große Rolle. Die Anfrage beinhaltet dabei eine Frage, auf die wir eine Antwort suchen. Dazu gilt es einen Textabschnitt zu extrahieren, welcher die Antwort auf die Frage darstellt oder zur Antwort führt. Passage Retrieval bringt aber auch Fortschritt für Information Retrieval im Allgemeinen.

Passage Retrieval entwickelte sich als Verbesserung zu Information-Retrieval-Systemen, die zum Abrufen relevanter Texte zu einer Anfrage entwickelt wurden. Das Ähnlichkeitsmaß, das bei diesen Systemen verwendet wurde, basiert auf Häufigkeiten von Wörtern im Text, die in der Anfrage vorkommen. Dadurch haben längere Texte eine höhere Chance, mehr Wörter der Anfrage abzudecken, als kürzere Texte und somit auch eine höhere Chance relevanter

zu sein. Mit Passage Retrieval wurde erreicht, dass das Ähnlichkeitsmaß von der Textlänge nicht beeinflußt wird (Llopis et al., 2002). Auch in Bezug auf Question Answering stellt Passage Retrieval einen Vorteil dar. Statt auf ganzen Texten agiert Passage Retrieval auf kürzeren Textabschnitten, wodurch sich der Rechenaufwand reduziert und die Präzision beim Abruf relevanterer Texte bzw. Textabschnitte erhöht.

Zu einer der wichtigsten Arbeiten im Bereich Information Retrieval (propagierend zu Passage Retrieval) gehört *BM25* (Robertson und Zaragoza, 2009). BM25, formuliert in den 1970er - 1980er Jahren, ist eine Ranking-Funktion mit der die Relevanz eines Textes bezüglich einer Anfrage berechnet werden kann und bis heute noch benutzt und weiterentwickelt wird. *BM25F* (Zaragoza et al., 2004) ist eine solche Weiterentwicklung. Darin werden Texte in Teilen, wie Titel, Haupttext oder Verweistext mit Gewichtung, statt als Ganzes in die Berechnung einbezogen. Bei Passage Retrieval spielt die Gewichtung von Wörtern als Maß ihrer Wichtigkeit im Text eine wichtige Rolle. Damit wird bestimmt, welche Wörter einer Anfrage relevant sind und somit auch im extrahierten Textabschnitt eine hohe Relevanz haben sollten. Traditionell funktioniert TF-IDF als Gewichtungsmaß bei Texten sehr zuverlässig.

MS MARCO[1] besitzt einen Passage-Retrieval-Task, bei dem die Effektivität von Systemen für Passage Retrieval in MRR (Mean Reciprocal Rank) ausgewertet wird. MRR gibt hier an, wie gut ein System die Relevanz eines Textabschnittes bezüglich einer Anfrage einschätzen kann. Da der Passage-Retrieval-Task von MS MARCO die zehn relevantesten Textabschnitte aus Sicht der Systeme auswertet, wird die Effektivität in MRR@10 angegeben. Der Kehrwert von MRR entspricht dem Rang des relevantesten Textabschnittes aus Sicht des ausgewerteten Systems. Ein Wert von 33.3 % MRR für ein System kann so interpretiert werden, dass das System den relevantesten Textabschnitt durchschnittlich als drittrelevantesten Textabschnitt einschätzt. Das beste System mit BM25 (Han et al., 2020) erreicht auf dem Passage-Retrieval-Task von MS MARCO einen MRR@10 von 40.5 %. Dai und Callan adressieren in ihrer Arbeit das Problem, dass TF-IDF bei Sätzen oder Textabschnitten nicht mehr so zuverlässig ist wie bei Dokumenten. Sätze oder Textabschnitte bestehen häufig aus zu wenig Wörtern, sodass spezifische Wortwiederholungen zu selten vorkommen, um aus ihnen genauso zuverlässig die Relevanz eines Wortes schlussfolgern zu können. So kann beispielsweise das Wort ‚Tinte' in zwei Textabschnitten gleich oft vorkommen. Dabei geht es im ersten Text um Tintenverwendung und im zweiten um Kugelschreiberherstellung. Bei *DeepCT* (Dai und Callan, 2019) werden die Wörter anhand des Kontexts eines Textabschnit-

[1] https://microsoft.github.io/msmarco/

tes neu gewichtet und damit bessere Ergebnisse erzielt als mit Systemen nur mit TF-IDF Gewichtung. Auf dem Passage-Retrieval-Task von MS MARCO ist DeepCT ein Teil eines BERT-RoBERTa-ELECTRA-Ensemble-Ansatzes, das 42.1 % MRR@10 erreicht (Han et al., 2020). Ein anderer aktueller Ansatz namens *COIL* (Gao et al., 2021) bezieht Kontextinformationen in die Repräsentation von Wörtern ein und verarbeitet diese als invertierte Listen. In Verbindung mit RoBERTa erreicht COIL auf dem Passage-Retrieval-Task von MS MARCO ein MRR@10 von 43.6 %. Eine weitere Art in Passage Retrieval nennt sich *Dense Passage Retrieval*. Dabei wird auf TF-IDF verzichtet und durch Worteinbettungen ersetzt. Dadurch können, im Gegensatz zu TF-IDF, verschiedene Synonyme oder ganze Paraphrasen als ähnlich erkannt werden. Der Nachteil dabei ist, dass Worteinbettungen erst trainiert werden müssen, was deutlich mehr Rechenleistung und Daten erfordert. Abhängig vom Training können Dense-Passage-Retrieval-Ansätze besser oder schlechter abschneiden als TF-IDF-Ansätze. Momentan steht ein Trainingsansatz names *RocketQA* (Qu et al., 2021) gepaart mit ERNIE an der Spitze der Rangliste des MS MARCO Passage-Retrieval-Tasks mit 43.9 % MRR@10.

2.5 Question Answering

Unter Question Answering verstehen wir das Beantworten von Fragen in natürlicher Sprache durch Informationssysteme, wie Algorithmen Maschinellen Lernens mit Hilfe von Dokumenten in natürlicher Sprache (Calijorne Soares und Parreiras, 2020). Es wird behauptet, dass Question Answering eine fortgeschrittenere Art von Information Retrieval ist (Cao et al., 2010). Information Retrieval konzentrierte sich im Allgemeinen auf die Darbietung ganzer Texte, die die benötigte Information enthalten. Question Answering hingegen stellt die benötigte Information in überschaubarer Form als beispielsweise Wortgruppe oder längeren Textabschnitt dar. Damit wird die Suche nach der benötigten Information im Text dem Informationssystem überlassen, und bleibt dem Menschen erspart.

Informationssysteme für Question Answering sind üblicherweise aus drei wesentlichen Phasen aufgebaut: Analyse der Frage, Document Retrieval und Analyse der Antwort (Bhoir und Potey, 2014; Neves und Leser, 2015). Zur Analyse der Frage wird unter anderem Question Classification genutzt (Abschnitt 2.3), um die Art der zu erwartenden Antwort zu erkennen. Document Retrieval befasst sich mit dem Abruf relevanter Texte, die die Antwort beinhalten könnten, mittels beispielsweise Passage Retrieval (Abschnitt 2.4). Dabei werden Quellen wie Datenbanken, JSON-Dokumente oder Texte in natürlicher Sprache genutzt, um aus diesen relevante Texte zu aggregieren. Bei der Analyse der Antwort wird die Information zur Art der zu erwartenden Antwort genutzt, um aus den von der Document-Retrieval-Phase ausgewählten Texten eine Antwort auf die Frage zu extrahieren. Als Alternative zum Extrahieren einer Antwort aus einem oder mehreren Texten existiert auch das Generieren von Antworten. In der Arbeit von Yin et al. wird ein System vorgestellt, das an eine *Knowledge Base* geknüpft ist. Die Knowledge Base liefert hier zusätzlich relevante Fakten zur Frage, um daraus eine Antwort auf die Frage generieren zu können.

Datensätze

Algorithmen Maschinellen Lernens erfordern Daten zum Trainieren und Analysieren. Dazu existieren für verschiedenste Aufgaben des Natural Language Processings wie Text Classification, Passage Retrieval und auch Question Answering öffentiche Datensätze. Dabei versuchen die Datensätze auf verschiedene Aspekte ihrer Aufgabe aufmerksam zu machen, die bis dahin nicht bedacht wurden. Zu den populärsten Datensätzen des Question Answerings gehören *SQuADv1.1* (Rajpurkar et al., 2016) und *TriviaQA* (Joshi et al., 2017).

Paragraph	In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail...Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called "showers".
Frage	What causes precipitation to fall?
Antwort	gravity
Frage	What is another form of precipitation besides drizzle, rain, snow, sleet and hail?
Antwort	graupel
Frage	Where do water droplets collide with the ice crystals to form precipitation?
Antwort	within a cloud

Abbildung 8: Beispiel eines Paragraphen eines Wikipedia-Artikels mit zugehörigen Frage-Antwort-Paaren in SQuADv1.1

SQuAD

SQuADv1.1 besteht aus 107,785 Fragen, die zu 536 Wikipedia-Artikeln gestellt wurden. Im Crowdsourcing-Prozess stellten sogenannte Crowdworker zu einem vorgegebenen Paragraphen mindestens drei und maximal fünf Fragen und markierten deren Antwort in diesem Paragraphen. Damit sind mehrere Frage-Antwort-Paare den Paragraphen eines Wikipedia-Artikels zugeordnet, wobei die Antwort einem Textabschnitt in dem zugehörigen Paragraphen entspricht (Abbildung 8). Die Fragen sind durchschnittlich 10 Wörter, die Paragraphen 120 Wörter und die Antworten 3 Wörter lang (Sen und Saffari, 2020). Rajpurkar et al. behaupten zwar, SQuADv1.1 besäße eine diverse Verteilung von Antwortklassen, trotzdem gehen diese nicht weit über die grobe Klasse factoider Fragen hinaus. Mit 93.6 % factoider Antworten (32.6 % Eigennamen, 31.8 % Nominalphrase, 19.8 % Numerisch, 5.5 % Verbalphrase, 3.9 % Adjektivphrase) und 6.4 % deskripiver Antworten (3.7 % Klauseln, 2.7 % Weiteres) ist die Kategorie deskriptiver Fragen kaum vertreten.

Die Effektivitäten von Systemen werden in der SQuAD-Rangliste in F1, angepasst für Textabschnitte, gemessen. Der F1-Score soll ungefähr angeben, zu wie viel Prozent die, von den Systemen berechneten Antworten, den in SQuADv1.1 angegebenen Antworten entsprechen. Der Mensch schafft auf SQuADv1.1 eine Effektivität von 86.8 % F1. Mittlerweile sind viele Question-Answering-Ansätze entwickelt worden, die deutlich besser auf SQuADv1.1 abschneiden als der Mensch. Die besten Question-Answering-Ansätze für SQuADv1.1[2] sind *LUKE* (Yamada et al., 2020) mit 95.4 % F1, *XLNet* (Yang et al., 2019) mit 95.1 F1 % und *SpanBERT* (Joshi et al., 2019) mit 94.6 F1 %. In einem weiteren Crowdsourcing-Prozess sollten Crowdworker bis zu fünf Fragen stellen, die vom Paragraphen, den sie präsentiert bekamen, nicht beantwortet werden können. Damit kamen für SQuADv2.0 weitere 53,775 Fragen hinzu, die unbeantwortbar sind. Um auf SQuADv2.0 erfolgreich zu sein, müssen die Ansätze im Unterschied zu SQuADv1.1 also auch erkennen können, ob eine Frage mit gegebenem Text überhaupt beantwortbar ist. Zu den besten Ansätzen für SQuADv2.0[3,4] gehören *Retro-Reader* mit 93.0 % F1 (Zhang et al., 2020b), *ALBERT* (Lan et al. 2019) mit 92.2 % F1 und *ELECTRA* (Clark et al. 2020) mit 91.4 % F1. Zum Vergleich erreicht XLNet auf SQuADv2.0 einen F1 von 90.7 %, 4.7 Punkte weniger als auf SQuADv1.1.

TriviaQA

Der TriviaQA-Datensatz wurde mit dem Fokus auf schwer zu beantwortende Fragen erstellt. Dementsprechend sind die Fragen größtenteils komplex, besitzen also eine hohe syntaktische und lexikalische Variabilität mit einer durchschnittichen Fragenlänge von 15 Wörtern (Abbildung 9). TriviaQA besteht aus 95,000 Frage-Antwort-Paaren mit durchschnittlich sechs Evidenztexten, auf deren Basis die Fragen beantwortet werden können. Anders als in SQuAD entstammen die Frage-Antwort-Paare Triviawebseiten und Online-Quiz-Wettbewerben. Mit Hilfe automatisierter Suchanfragen wurden die Antwort-Frage-Paare um Texte zu den entsprechenden Themen aus Wikipedia und/oder dem allgemeinen Web erweitert. Durchschnittlich bekommt jedes Frage-Antwort-Paar einen Text aus 746 Wörtern zugeordnet. 93 % der Antworten sind Wikipediatitel, welche zu einem sehr großen Teil Eigennamen sein sollten, denn durchschnittlich sind die Antworten in TriviaQA zwei Wörter lang. Daraus kann geschlossen werden, dass auch TriviaQA die Kategorie deskriptiver Fragen kaum abdeckt. Von Joshi et al. wird aber behauptet, dass das Beantworten der Fragen mehr satzübergreifendes Verständnis benötige,

[2] https://paperswithcode.com/sota/question-answering-on-squad11
[3] https://rajpurkar.github.io/SQuAD-explorer/
[4] https://paperswithcode.com/sota/question-answering-on-squad20

Frage	The Dodecanese Campaign of WWII that was an attempt by the Allied forces to capture islands in the Aegean Sea was the inspiration for which acclaimed 1961 commando film?
Antwort	The Guns of Navarone
Frage	American Callan Pinckney's eponymously named system became a best-selling (1980s-2000s) book/video franchise in what genre?
Antwort	Fitness
Frage	What fragrant essential oil is obtained from Damask Rose?
Antwort	Attar

Abbildung 9: Beispiele für Fragen in TriviaQA

was vorteilhaft für das Lernen von Textverständnis wäre. Zum Vergleich wurde von Joshi et al. *BiDAF* (Seo et al., 2018), der zu der Zeit beste Question-Answering-Ansatz auf SQuADv1.1, mit einem F1 von 68.0 %, auf TriviaQA getestet, und erreichte einen F1 von 40.0 % (F1 bezieht sich hierbei auf den F1-Score von SQuAD). Menschen zum Vergleich schaffen eine Effektivität von 80.0 % F1. Auch auf TriviaQA existieren schon Question-Answering-Ansätze, die effektiver als der Mensch sind[5,6], aber lange nicht so zahlreich, wie für SQuAD. Zu den besten zählen *SpanBERT* (Joshi et al., 2019) mit einem F1 von 83.6 %, *BigBird* (Zaheer et al., 2021) mit einem F1 von 80.9 % und *MemoReader* (Back et al., 2018) mit einem F1 von 72.3 %.

HotPotQA

Ein weiterer potenziell interessanter Datensatz ist HotPotQA (Yang et al., 2018). In der Arbeit zu HotPotQA kritisieren die Autoren Yang et al., dass die Antworten in SQuAD nur mit der Information aus einem einzelnen Paragraphen gegeben werden können. Sie behaupten, dass die Fragen in SQuAD nur über einen Abgleich der Frage mit einzelnen Sätzen aus dem Paragraphen beantwortbar sind. Zu TriviaQA meinen die Autoren von HotPotQA, dass dessen Fragen durch Abgleich der Frage mit wenigstens ein paar wenigen benachbarten Sätzen beantwortbar ist, was aber dennoch nicht erstrebenswert erscheint. Einer Untersuchung zur Frage: ‚Was lernen Verfahren von Question-Answering-Datensätzen?' (Sen und Saffari, 2020) stützt die genannten Kritikpunkte von Yang et al. Hinzu kommt, dass beide Datensätze nur die korrekte Antwort

[5]https://paperswithcode.com/sota/question-answering-on-triviaqa
[6]https://competitions.codalab.org/competitions/17208#results

Single-Hop	What is the name of the member of Mother Love Bone who died just before the release of "Apple"?
Aufgabe	Finde den Namen des Bandmitglieds
Multi-Hop	What was the former band of the member of Mother Love Bone who died just before the release of "Apple"?
Aufgabe	Finde den Namen des Bandmitglieds und damit dessen ursprüngliche Band

Abbildung 10: Beispiel einer Multi-Hop-Frage im Vergleich zu einer Single-Hop-Frage.

zeigen, aber keine antwortstützenden Textabschnitte, die belegen, warum die Antwort korrekt ist. Nach Meinung von Yang et al. hindert das den Fortschritt des Sprachverständnisses von Question-Answering-Ansätzen.

HotPotQA wurde dadurch mit der Intention erstellt, das Lernen von Sprachverständnis bei Question-Answering-Verfahren zu fördern. Die an den anderen Datensätzen kritisierten Punkte sollten mit HotPotQA aufgearbeitet werden. HotPotQA besteht aus 112,779 Frage-Antwort-Paaren mit sogenannten Single- und Multi-Hop-Fragen. Single-Hop bedeutet, dass die Antwort nur nach einer spezifischen Information, wie einem Eigennamen sucht, die meist mit Hilfe eines einzelnen Paragraphen beantwortbar ist. Multi-Hop hingegen erfordert das Verknüpfen von mindestens zwei Informationen von mindestens zwei Paragraphen, um auf die Antwort zu kommen. Rund 84 % der Frage-Antwort-Paare in HotPotQA bestehen aus einer Multi-Hop-Frage. In Abbildung 10 ist der Unterschied zwischen Single- und Multi-Hop illustriert. Im Beispiel einer Multi-Hop Frage aus HotPotQA (Abbildung 11) muss ein Question-Answering-Ansatz zuerst dem Paragraphen A den Namen des Bandmitglieds entnehmen, um danach mit dessen Hilfe in Paragraph B auf die Antwort ‚Malfunkshun' der Frage ‚What was the former band of the member of Mother Love Bone who died just before the release of 'Apple'?' geben zu können. Da HotPotQA auch antwortstützende Sätze aus den Paragraphen angibt, kann ein Question-Answering-Ansatz auch lernen, warum die angegebene Antwort richtig ist. Jedem Frage-Antwort-Paar sind zehn Paragraphen zugeordnet, wovon genau zwei Paragraphen alle antwortstützenden Sätze enthalten, die in HotPotQA angegeben sind. Die restlichen acht Paragraphen dienen der Ablenkung. Durchschnittlich muss ein Question-Answering-Ansatz aus 42 Sätzen (1106 Wörter) die korrekten antwortstützenden Sätze auswählen können, die auf die korrekte Antwort hin-

deuten. Leider beinhaltet HotPotQA genauso wie SQuAD und TriviaQA kaum deskriptive Fragen, sondern ausschließlich factoide Fragen. Um die Effektivität von Question-Answering-Verfahren auf HotPotQA vergleichen zu können, wurde von Yang et al. der Joint-F1-Score vorgestellt. Dieser errechnet sich wie der gewöhnliche F1-Score, wobei die Precision aus dem Produkt der Presicion von Antwort und Presicion der antwortstützenden Sätze berechnet werden und der Recall analog dazu. Zu den besten Question-Answering-Verfahren für HotPotQA zählen *SAE+* (Tu et al. 2020) mit 75.7 % Joint-F1, *HGN* (Fang et al., 2020) mit 74.2 % Joint-F1 und *BigBird* (Zaheer et al., 2021) mit 73.6 % Joint-F1. Der Mensch erreicht im Vergleich dazu eine Effektivität von 82.6 % Joint-F1. Das heißt, anders als bei SQuAD und TriviaQA existiert noch kein Question-Answering-Ansatz, das für HotPotQA ähnlich gut oder besser ist als der Mensch. Daraus schließen wir, dass Question-Answering-Ansätze noch nicht so gut Informationen aus Texten als zusammengehörend erkennen und daraus schlussfolgern können wie Menschen.

Paragraph A ‚Return to Olympus‘	[1] Return to Olympus is the only album by the alternative rock band Malfunkshun. [2] It was released after the band had broken up and after lead singer Andrew Wood (later of Mother Love Bone) had died of a drug overdose in 1990 [3] Stone Gossard, of Pearl Jam, had compiled the songs and released the album on his label, Loosegroove Records.
Paragraph B ‚Mother Love Bone‘	[4] Mother Love Bone was an American rock band that formed in Seattle, Washington in 1987. [5] The band was active from 1987 to 1990. [6] Frontman Andrew Wood's personality and compositions helped to catapult the group to the top of the burgeoning late 1980s/early 1990s Seattle music scene. [7] Wood died only days before the scheduled release of the band's debut album, 'Apple', thus ending the group's hopes of success. [8] The album was finally released a few months later.
Frage	What was the former band of the member of Mother Love Bone who died just before the release of "Apple"?
Antwort	Malfunkshun
antwortstützende Sätze	1, 2, 4, 6, 7

Abbildung 11: Beispiel einer Multi-Hop-Frage in HotPotQA. Die Paragraphen A und B stellen die zwei Texte dar, die alle notwendigen Informationen zum Beantworten der Frage enthalten. Die Antwortstützenden Textabschnitte sind blau hinterlegt.

Kapitel 3

Datensatz

Derzeit existieren keine öffentlichen Datensätze, die sich dem Spoilern von Clickbait widmen. Deshalb ist es notwendig, zur Thematik des Spoilerns von Clickbait einen neuen Datensatz aufzubauen, der zum Analysieren der Daten und zum Trainieren von Algorithmen Maschinellen Lernens verwendet werden kann. Mit Hilfe der Webis-Group[1] können wir Daten aus den Datensätzen *Webis-Clickbait-17*, *Webis-Clickbait-18* und *Webis-Clickbait-19* wiederverwenden und erweitern.

3.1 Webis-Clickbait-17

Der Webis-Clickbait-17-Datensatz wurde zur Problemstellung der Clickbait-Erkennung von Martin Potthast et al. aufgebaut, um bei der Clickbait-Challenge-2017[2] die Forschung auf dem Gebiet der Clickbait-Erkennung voranzutreiben. Der Datensatz mit insgesamt 38,517 Tweets wurde in zwei wesentliche Teile aufgeteilt (Tabelle 1): Der Teil *Train* wurde als Trainingsbasis für Clickbait-Erkennungs-Ansätze veröffentlicht, der Teil *Test* wurde privat von der Webis-Group beim Evaluieren der bei der Clickbait-Challenge eingesendeten Clickbait-Detektoren genutzt (Potthast et al. 2018a). Damit

Tabelle 1: Verteilung der Tweets in Webis-Clickbait-17.

Train	Test	$\sum$
19,538	18,979	38,517

[1] https://webis.de
[2] https://clickbait-challenge.org/

die Voreingenommenheit des Korpus möglichst gering bleibt, wurden über 150 Tage lang Tweets von den 27 meist re-tweeteten Publishern wie *The Washington Post* und *Business Insider* gesammelt und anschließend auf 10 Tweets pro Tag und Publisher herunterskaliert.

Die Tweets wurden von fünf Personen mit Hilfe einer Likert-Skala von „not“ (kein) über ‚slightly‘ (leichter) und ‚considerably‘ (wesentlicher) bis hin zu ‚heavily‘ (starker) Clickbait annotiert und dabei als numerischer Wert (0.0, 0.33, 0.66, 1.0) gespeichert. Im Datensatz sind diese Werte unter dem Schlüssel `truthJudgements`, deren Mittelwert unter dem Schlüssel `truthMean`, zu finden. 9,276 der Tweets aus Webis-Clickbait-17 wurden laut Potthast et al. von der Mehrheit der Annotatoren als Clickbait angenommen (Tabelle 4). Das heißt, mindestens drei der fünf Annotatoren hatten den Tweet als „considerably“ Clickbait annotiert. Der Aufbau des Datensatzes ist in Abbildung 12 und Abbildung 13 zu erkennen.


```
{
  "id": "842024098050002944",
  "postMedia": ["media/photo_842024095105613824.jpg"],
  "postText": ["Will you inherit less than you t..."],
  "targetCaptions": [],
  "targetParagraphs": ["Everyone should be grate..."],
  "targetTitle": "Hoping for an Inheritance? You ...",
  "postTimestamp": "Wed Mar 15 14:46:03 +0000 2017",
  "targetKeywords": "personal-finance",
  "targetDescription": "Keeping inheritance expec..."
}
```


Abbildung 12: Beispiel für ein JSON-Objekt eines Tweets in Webis-Clickbait-17.


```
{
  "id": "842024098050002944",
  "truthJudgements": [1.0, 0.6666666666, 1.0,
                      0.6666666666, 1.0],
  "truthMean": 0.86666666664,
  "truthClass": "clickbait",
  "truthMedian": 1.0,
  "truthMode": 1.0
}
```


Abbildung 13: Beispiel für ein JSON-Objekt der Annotationen eines Tweets in Webis-Clickbait-17.

3.2 Webis-Clickbait-18

Teile des Webis-Clickbait-18-Datensatzes wurden im Rahmen einer Bachelorarbeit innerhalb der Webis-Group aufgebaut und ist eine erste Art eines Clickbait-Spoiling-Datensatzes (Ter-Akopyan, 2017). Der Datensatz besteht aus insgesamt 5,787 Reddit-, Twitter- und Facebook-Posts von Accounts[3], welche regelmäßig Clickbait-Posts spoilern. Die genaue Verteilung der Social-Media-Plattformen ist in Tabelle 2 zu sehen. In Ter-Akopyans Arbeit wurden verschiedene Informationen, wie automatisch extrahierte Informationen aus dem verlinkten Text, die Kategorie des Clickbaits oder Named-Entities des Spoilers in verschiedenen Datensätzen gespeichert. Für eine bessere Übersicht wurden diese Informationen in einer weiteren Bachelorarbeit in einem Datensatz (Webis-Clickbait-18) zusammengefasst (Puschmann, 2019). Alle Schlüssel mit dem Präfix `article_` wurden mit der Python-Bibliothek `newspaper` aus dem verlinkten Text, Link unter Schlüssel `article_url` angegeben, automatisch extrahiert. Die Schlüssel `cb_headline`, `cb_spoiler`, `social_media_platform` und `spoiler_publisher` entstammen der Twitter- bzw. Facebook-API und wurden lediglich umbenannt. Der Schlüssel `answer_type` wurde manuell für 745 Clickbait-Posts mit Hilfe der Named-Entites der Spoiler annotiert und bezieht sich auf die Kategorie des Spoilers des Clickbait-Posts. Der genaue Aufbau des Datensatzes ist in Abbildung 14 zu sehen, genau wie die Verteilung der Werte von `answer_type` in Tabelle 3.

Tabelle 2: Verteilung der Social-Media-Plattformen in Webis-Clickbait-18 von denen die Clickbait-Posts stammen.

Reddit	Twitter	Facebook	$\sum$
2,538	2,108	1,141	5,787

Tabelle 3: Verteilung der Named-Entities von Spoilern in Webis-Clickbait-18 angegeben unter dem Schlüssel `answer_type`.

Spoilerklasse	PERSON	LOCATION	ORGANIZATION	DATE	$\sum$
Anzahl	429	219	47	50	745

[3] r/savedyouaclick, @HuffPoSpoilers, @SavedYouAClick, @UpworthySpoiler, @StopClickBaitOfficial

```
{
  "post_id_str":"497438270801063937",
  "article_description":"The Cleveland Cavali...",
  "article_imgs":["http://slate.me/blogs/the..."],
  "article_keywords":"kevin love trade, kevin...",
  "article_movies":[],
  "article_tags":[],
  "article_text":"Photo by Steve MitchellUSA ...",
  "article_title":"Cleveland Will Acquire Sta...",
  "article_url":"http://slate.me/1ogCSvp",
  "answer_type":"PERSON",
  "categories":["hl_cataphora"],
  "cb_headline":" @Slate: Cleveland will acqu...",
  "cb_spoiler":"Kevin Love ",
  "social_media_platform":"Twitter",
  "spoiler_publisher":"HuffPoSpoilers",
  "spoiler_named_entities":["PERSON","PERSON"],
  "spoiler_word":["Kevin","Love"],
  "modified_spoiler":["Kevin Love"]
}
```


Abbildung 14: Beispiel für ein JSON-Objekt eines Clickbait-Posts in Webis-Clickbait-18.

3.3 Webis-Clickbait-19

Der Webis-Clickbait-19-Datensatz wurde in der Arbeit von Puschmann aus den vorher beschriebenen Datensätzen Webis-Clickbait-17 und Webis-Clickbait-18 zusammengebaut und besteht aus 3,042 Clickbait-Posts mit Spoilern. Aus dem Webis-Clickbait-17 wurden 1,845 Twitter-Posts mit einem `truthMean`-Wert größer als 0.8 übernommen (Tabelle 5), damit Posts relativ sicher als Clickbait einzustufen sind. Webis-Clickbait-18 wurde vollständig (5,787 Einträge) in den Webis-Clickbait-19-Datensatz übernommen. Von den sich ergebenden 7,632 Clickbait-Posts wurden 418 herausgefiltert, die einen leeren `postText` oder `article_text` aufwiesen, sodass der Basis-Datensatz auf insgesamt 7,214 Clickbait-Posts reduzuert wurde (Tabelle 6). Um sicher zu gehen, dass die Clickbait-Posts eindeutig referenzierbar bleiben, wurde von Puschmann auch eine UUID (Universally Unique Identifier) für jeden Clickbait-Post errechnet und beigefügt.

Die Clickbait-Posts des Basis-Datensatzes sollten dann in einem Crowdsourcing-Prozess mit Spoilern annotiert werden. Dafür hat sich *Amazon Mechanical Turk*[4] angeboten. Amazon Mechanical Turk ist eine Crowdsourcing-Webseite, auf der man für Aufgaben, wie das Spoilern von Clickbait-Posts, sogenannte Crowdworker anfordern kann, die diese Aufgaben bearbeiten. Mit Hilfe des *MTurk-Managers*[5] wurde ein Amazon-Mechanical-Turk-Projekt erstellt, welches die Clickbait-Posts auf Crowdworker verteilte. Die Aufgabe der Crowd-

Tabelle 4: Anzahl von `truthMedian`-Werten über einem bestimmten Schwellwert in Webis-Clickbait-17.

Schwellwert	>0.66	=1.0
Anzahl Posts	9,276	3,670

Tabelle 5: Anzahl von `truthMean`-Werten über einem bestimmten Schwellwert in Webis-Clickbait-17.

Schwellwert	>0.7	>0.75	>0.8
Anzahl Posts	4,277	2,987	1,845

Tabelle 6: Verteilung der Ursprungsdatensätze des Basis-Datensatzes und dem daraus resultierenden Webis-Clickbait-19-Datensatz.

	Webis-Clickbait-17	Webis-Clickbait-18	$\sum$
Basis-Datensatz	5,408	1,806	7,214
Webis-Clickbait-19	2,675	367	3,042

[4] https://www.mturk.com/
[5] https://github.com/webis-de/mturk-manager

Tabelle 7: Verteilung der Social-Media-Plattformen in Webis-Clickbait-19 von denen die Clickbait-Posts stammen.

	Twitter	Reddit	Facebook	$\sum$
Basis-Datensatz	3,706	2,427	1,081	7,214
Webis-Clickbait-19	1,428	1,085	529	3,042

worker bestand darin, einen oder mehrere aufeinanderfolgende Sätze aus dem verlinkten Text zu extrahieren, die den Clickbait-Post spoilern. In dem Crowdsourcing-Prozess wurden 7,911 Annotationen für 7,199[6] Clickbait-Posts eingeholt. Die Clickbait-Posts wurden dabei in 15 Teile aufgeteilt: 14 Teile bestanden aus 499 Posts und der 15. Teil aus 213 Posts. Alle Teile wurden jeweils von genau einer Person annotiert, der 1. und der 15. Teil auch von jeweils einer weiteren Person (499+213 extra Annotationen).

Aus den eingeholten Annotationen wurden letztendlich 3,042 in den resultierenden Webis-Clickbait-19-Datensatz übernommen. Die sich ergebende Verteilung der Social-Media-Plattformen in Webis-Clickbait-19 finden wir in Tabelle 7. Die restlichen Annotationen entsprachen entweder nicht den Anforderungen der Aufgabe, wenn z.B. von einem Crowdworker ein selbst zusammengefasster Satz angegeben wurde, anstatt eines aus dem Text extrahierten Satzes. Diese Sätze wurden dementsprechend abgelehnt. Abgelehnt wurden ebenfalls Annotationen, korrekter- oder fälschlischerweise angaben, dass der Clickbait-Post nicht (textuell) spoilerbar wäre. Ein weiterer Grund für das Ablehnen zahlreicher Annotationen war, dass einige Clickbait-Posts eine Auflistung von Fakten als Spoiler erwarteten. Diese Fakten sind meist nicht aufeinanderfolgend im verlinkten Text zu finden und deshalb im Rahmen der Aufgabe des Crowd-Sourcing-Prozesses nicht spoilerbar. – Genauso wie das Erfordernis, ein Video anzuschauen oder an einem Quiz teilzunehmen, im Rahmen der Aufgabe nicht umsetzbar sind. Das Ergebnis des Crowdsourcing-Prozesses ist in Tabelle 8 zu sehen. In Abbildung 15 ist der Aufbau von Webis-Clickbait-19 abgebildet. Dabei sind die Schlüssel `targetUrl` und `humanSpoiler` Webis-Clickbait-18 exklusiv und stellen die URL des verlinkten Textes bzw. den im Post angegebenen Spoiler dar.

[6] 15 Posts beim Aufteilen verloren gegangen, weil bei der Indexierung der CSV-Header nicht beachtet wurde

Tabelle 8: Ergebnis des Crowdsourcing-Prozesses für den Basis-Datensatz von Webis-Clickbait-19. Angegeben sind die Anzahlen der angenommenen und abgelehnten Annotationen mit gefundenen und nicht gefundenen Spoilern.

	Angenommen	Abgelehnt	$\sum$
Spoiler gefunden	3,477	2,387	5,864
Keinen Spoiler gefunden	821	1,226	2,047
$\sum$	4,298	3,613	7,911


```
{
  "uuid": "4e41144f-4985-44be-b69f-d009c1a4a5ed",
  "targetMedia": null,
  "targetParagraphs": ["Everyone should be ..."],
  "targetTitle": "Hoping for an Inheritance?...",
  "targetDescription": "Keeping inheritance ...",
  "targetKeywords": "personal-finance",
  "targetUrl": null,
  "postId": "842024098050002944",
  "postMedia": ["media/photo_84202409510561..."],
  "postText": ["Will you inherit less than ..."],
  "postPlatform": "Twitter",
  "humanSpoiler": null,
  "originalCorpus": "webis-clickbait-17",
  "spoilerSentences": ["When kids do find o..."],
  "spoilerSentencePositions": [[0, 226, 339]]
}
```


Abbildung 15: Beispiel für ein JSON-Objekt eines Clickbait-Posts in Webis-Clickbait-19.

3.4 Webis-Clickbait-Spoiling-20

Aufgrund der relativ geringen Ergebnismenge an gespoilerten Clickbait-Posts (42 %) in Webis-Clickbait-19 im Vergleich zur Menge der Clickbait-Posts in dessen Basis-Datensatz, ändern wir die Aufgabenstellung zum Spoilern und prüfen die Annotationen noch einmal manuell nach. In unserem Datensatz Webis-Clickbait-Spoiling-20 muss der Spoiler kein Satz bzw. mehrere aufeinanderfolgende Sätze aus dem verlinkten Text sein, sondern kann auch einen oder mehrere Abschnitte aus dem verlinkten Text darstellen. Damit lösen wir das Problem von Webis-Clickbait-19, dass Auflistungen von Fakten nicht spoilerbar waren. Beispielsweise ist Clickbait, wie ‚*5 Weißheiten*, die ihr Leben verändern', ‚*Diese U.S.-Stadt* heißt Veteranen mit einem *neuen innovativen Programm* willkommen.', oder ‚Werde auch du reich mit *diesen Expertentipps*!' gespoilert in Webis-Clickbait-Spoiling-20 enthalten. In Anbetracht der neuen Aufgabenstellung annotieren wir selbst Clickbait-Posts, die in der Bachelorarbeit von Jana Puschmann ungespoilert blieben. Damit kommen wir auf insgesamt 5,000 gespoilerte Clickbait-Posts in Webis-Clickbait-Spoiling-20. Die Verteilung der Ursprungsdatensätze sehen wir in Tabelle 9 und die der Social-Media-Plattformen in Tabelle 10.

Beim Annotieren der Spoiler finden wir Clickbait-Posts mit falsch annotierten Texten (Abbildung 16), die nicht den korrekten verlinkten Text wiederspiegeln und somit den Spoiler auch nicht enthalten und unnütz sind. Wir finden weiterhin auch unvollständige Texte (Abbildung 17), die auch nicht den korrekten Spoiler enthalten. Diese korrigieren wir während des Annotierens, soweit eine URL zum verlinkten Text vorhanden ist oder der Clickbait sich im Internet auffinden lässt. Bei letzterem korrigieren wir auch die URL, unter der wir den Text gefunden haben. Genauso kürzen wir die Spoiler auf die notwendigsten Informationen, wie in Abbildung 18 zu sehen ist. Da unser Spoiler auch mehr-

Tabelle 9: Verteilung Ursprungsdatensätze von Webis-Clickbait-Spoiling-20.

Webis-Clickbait-17	Webis-Clickbait-18	$\sum$
4,204	796	5,000

Tabelle 10: Verteilung der Social-Media-Plattformen in Webis-Clickbait-Spoiling-20 von denen die Clickbait-Posts stammen.

Twitter	Reddit	Facebook	$\sum$
2,379	1,797	824	5,000

Clickbait-Nachricht	10 Things About Man Logic That Women Will Never Understand
Link	http://archive.is/QmR7t
verlinkter Text	no other snapshots from this urlno other snapshots from this url

Abbildung 16: Clickbait-Posts aus Webis-Clickbait-19 mit falsch annotiertem verlinkten Text. Diese enthalten keinen Spoiler zur Clickbait-Nachricht.

Clickbait-Nachricht	Social Security's looming $32 trillion shortfall
Link	http://www.cnbc.com/2016/08/08/social-securitys-lo...
verlinkter Text	... The Social Security Administration projects that unfunded obligations will reach $11.4 trillion by 2090. That's up $700 billion from the $10.7 trillion the administration projected for its 2089 shortfall.

Abbildung 17: Clickbait-Post aus Webis-Clickbait-19 mit unvollständigem verlinkten Text. Hierbei handelt es sich bei dem Text nur um die Einleitung zum eigentlichen Text.

teilig sein kann, wodurch dieser mehrere Spoilerabschnitte besitzen kann, speichern wir das erste Vorkommen unter dem Schlüssel `spoilerPositions` jeden Spoilerabschnittes wie folgt: Für jeden annotierten Spoilerabschnitt speichern wir die Position des ersten Zeichens des Spoilers im Text als Paragraphindex in dem das Zeichen sich befindet und Position des Zeichens im Paragraphen und analog für das letzte Zeichen des Spoilerabschnittes. Dabei ist zu beachten, dass Paragraphindex ‚-1' für den Titel des verlinkten Textes steht.

In Bezug auf Lösungsansätze zum Spoilern von Clickbait, annotieren wir auch eigens festgelegte Spoilerklassen. Im Groben teilen wir die Spoiler in drei Klassen ein: kurz (*Phrase*), lang (*Passage*) und mehrteilig (*Multi*). Zu Spoilern der Klasse Phrase gehören beispielsweise Eigennamen, wie ‚Kevin Love' aus Abbildung 2 oder kurze Wortgruppen, bestehend aus maximal fünf Wörtern wie, ‚Den Sitz falsch installiert'. Spoiler der Klasse Passage sind alle Spoiler, die länger sind als die der Klasse Phrase und dabei genau einem Textabschnitt aus dem verlinkten Text entsprechen, wie ‚Mehr als die Hälfte der 2,700 befragten

Clickbait-Nachricht	The refrigerator of the future will do WHAT?
Spoiler vorher	The most notable ideas for innovative products include an entertainment center that also has cooking capabilities and a refrigerator that can take inventory of what's inside and place an order for a food delivery.
Spoiler nachher	can take inventory of what's inside and place an order for a food delivery

Clickbait-Nachricht	Cleveland will acquire this star to play alongside LeBron in a blockbuster NBA trade:
Spoiler vorher	The Cleveland Cavaliers will acquire star forward Kevin Love from the Minnesota Timberwolves in exchange for two prospects - including this year's top draft pick, Andrew Wiggins - and a future draft choice, Yahoo's Adrian Wojnarowski reports.
Spoiler nachher	Kevin Love

Abbildung 18: Beispiele für Clickbait-Nachrichten, deren Spoiler wir im Vergleich zu Webis-Clickbait-19 in Webis-Clickbait-Spoiling-20 kürzen.

Erwachsenen...' aus Abbildung 3. Clickbait-Nachrichten, die ähnlich zu einer Ja-Nein-Frage als Spoiler ein ‚Ja' oder ein ‚Nein' erwarten, behandeln wir gesondert. Meistens spoilert der verlinkte Text nicht direkt als ‚Ja' oder ‚Nein', sondern es müsste erst gelesen werden. Aus diesem Grund annotieren wir den aussagekräftigsten Textabschnitt als Spoiler, aus dem z.B. ein ‚Ja' geschlossen werden kann (Abbildung 3) und annotieren diese als Spoiler der Klasse Passage. Da ein Spoiler in Webis-Clickbait-Spoiling-20 auch aus mehreren Textabschnitten bestehen kann, ordnen wir solche Spoiler der Klasse Multi zu. Beim Annotieren von Spoilern der Klasse Multi nehmen wir nur die ersten fünf chronologisch vorkommenden bzw. die besten fünf aus einer Rangordnung im verlinkten Text. Lesen wir beispielsweise einen verlinkten Text über die Top 50 schönsten Strände Frankreichs, reichen uns meist die schönsten fünf, um den Curiosity Gap zu schließen. Mögliche Beispiele für Spoiler der Klasse Multi sind in Abbildung 19 aufgeführt. Mit annotierten Spoilerklassen eröffnen wir die Möglichkeit, zusammengesetzte Ansätze zum Spoilern von

Clickbait-Nachricht	*9 surprising ways* you're wasting money without even realizing it:
Spoiler	• Expired Food • Not Comparing Prices or Missed Savings • Paying Interest on Credit Card Purchases • Useless Bank Fees • Name Brands
Clickbait-Nachricht	
Spoiler	*The best places* to retire without a car: • Ann Arbor, Michigan • Arlington, Virginia • Berkeley, California • Boston, Massachusetts • Boulder, Colorado
Clickbait-Nachricht	The *former heroin addict* who became a millionaire with a *simple idea*
Spoiler	• Khalil Rafati • juices' unique ingredients

Abbildung 19: Beispiele für Spoiler der Klasse Multi. Kursiv hervorgehoben ist die Information die wir spoilern wollen.

Clickbait-Nachrichten nutzen zu können, indem wir Sprachverständnismodelle zum Spoilern jeweils einer Klasse trainieren und mit Hilfe eines vorgeschalteten Klassifikators zusammenschließen. Der Klassifikator sagt uns dabei voraus, welche Spoilerklasse zu erwarten ist, und ist somit notwendig, um den richtigen Ansatz spoilern zu lassen. Damit das Training der Sprachverständnismodelle möglichst erfolgreich verläuft, reinigen wir die verlinkten Texte der Clickbait-Posts auf mögliche Störfaktoren, die dem inhaltlichen Thema des verlinkten Textes keinen Mehrwert bieten. Beispiele für Störfaktoren wären Referenzen auf weitere Texte des Publishers, Werbetexte zwischen den einzelnen Paragraphen, Quellen von Bildern oder typische Angaben zu Social-Media zum Liken, Teilen, Abonnieren, ... (Abbildung 20). Dafür nutzen wir typische Schreibweisen der Störfaktoren, wie ‚Advertisement', ‚Read more', ‚Subscribe for more' oder ‚Photo by' um Paragraphen mit Störfaktoren ausfindig zu machen und diese, sowie weitere auffällige Paragraphen im selben verlinkten Text zu löschen. Für die Reinigung der Texte benötigen wir etwa 160 Stunden. Der Aufbau von Webis-Clickbait-Spoiling-20 ist in Abbildung 21 zu erkennen.

Aus Zeitgründen beginnen wir den Bau eines Klassifikators und das Training von Sprachverständnismodellen auf einem Teildatensatz (*1kTrain*) von 1,000 Einträgen, der fertig annotiert und gereinigt ist. Damit kommen wir schon zu Erkenntnissen, bevor der gesamte Datensatz vollständig aufbereitet ist, welche dennoch sehr ähnlich zu den Erkenntnissen auf dem gesamten Datensatz sein sollten. Die Verteilung der Ursprungsdatensätze des Teildatensatzes 1kTrain ist in Tabelle 11 zu sehen, sowie die Social-Media-Plattformen in Tabelle 12 und die Klassen von Clickbait in Tabelle 13.

Störfaktor	Text
Referenzen	Also From CNBC; Watch The Profit on Yahoo View, available now...; Don't miss: 11 bad money habits to break before 30; READ ALSO: Duterte gives go signal to Marcos' ...
Werbetext	Advertisement Continue reading the main story below
Quellen	Image Credits: Lucasfilm; Photo by Jeff Haller
Sozial-Media	Subscribe To The Forbes Careers Newsletter; Follow the Community on Facebook and Twitter

Abbildung 20: Beispiele für Teile eines verlinkten Textes, die wir als Störfaktoren im Text sehen.

Tabelle 11: Verteilung Ursprungsdatensätze von 1kTrain.

Webis-Clickbait-17	Webis-Clickbait-18	$\sum$
512	488	1,000

Tabelle 12: Verteilung der Social-Media-Plattformen in 1kTrain von denen die Clickbait-Posts stammen.

Twitter	Facebook	Reddit	$\sum$
885	61	54	1,000

Tabelle 13: Verteilung der Klassen von Spoilern in 1kTrain.

Passage	Phrase	Multi	$\sum$
500	400	100	1,000

```
{
  "uuid": "a4883648-6b98-4f56-b794-97241a022210",
  "postId": "497438270801063937",
  "postText": ["Cleveland will acquire this ..."],
  "postPlatform": "Twitter",
  "targetParagraphs": ["Photo by Steve Mitch..."],
  "targetTitle": "Cleveland Will Acquire Star...",
  "targetDescription": "The Cleveland Cavalie...",
  "targetKeywords": "kevin love trade, kevin ...",
  "targetMedia": ["http://slate.me/blogs/the..."],
  "targetUrl": "http://slate.me/1ogCSvp",
  "provenance": {
    "source": "webis-clickbait-18",
    "humanSpoiler": "Kevin Love",
    "spoilerPublisher": "HuffPoSpoilers"
  },
  "spoiler": ["Kevin Love"],
  "spoilerPositions": [[[0, 50], [0, 60]]],
  "tags": ["phrase"]
}
```


Abbildung 21: Beispiel für ein JSON-Objekt eines Clickbait-Posts in Webis-Clickbait-Spoiling-20.

Kapitel 4

Klassifikation der Spoiler

In diesem Kapitel widmen wir uns dem Teil unseres Lösungsansatzes, der sich mit dem Klassifizieren der Spoiler von Clickbait befasst. Zu jeder Spoilerklasse eines Clickbaits wollen wir in unserem Lösungsansatz ein eigens dafür vorgesehenes Verfahren zum Spoilern benutzen. Dafür bauen wir in diesem Kapitel einen Spoiler-Typ-Klassifikator. Dieser soll uns mit Hilfe der Clickbait-Nachricht und dem verlinkten Text die Spoilerklasse eines Clickbaits vorhersagen können (Abbildung 22). Zuerst betrachten wir die Klassifizierung aller drei Klassen Phrase, Passage und Multi. Im experimentellen Verlauf entscheiden wir uns die Klasse Multi nicht extra zu betrachten und unterscheiden im weiteren Verlauf nur die Klasse Phrase von den Klassen Passage und Multi.

4.1 Baselines

Wie schon in Abschnitt 3.4 erwähnt, unterscheiden wir zwischen Spoilern der Klasse Phrase (Abbildung 2), Passage (Abbildung 3) und Multi (Abbildung 19). Zu Spoilern der Klasse Phrase gehören beispielsweise Eigennamen wie ‚Kevin Love' oder ‚Empire State Building' und kurze Wortgruppen, bestehend aus maximal fünf Wörtern, wie ‚Den Sitz falsch installiert' oder ‚Nicht richtig reingesetzt'. Damit zählen kurze Nominal-, Adjektiv- und Verbialphrasen zu Spoilern der Klasse Phrase. Im Allgemeinen lässt sich also behaupten, dass die Klasse Phrase alle Clickbait-Nachrichten umfasst, deren Spoiler factoid sind. Alle Spoiler, die nicht zur Klasse Phrase dazugehören, aber als ein einzelner ununterbrochener Textabschnitt aus dem verlinkten Text gespoilert werden können, ordnen wir der Klasse Passage zu. Damit sollte der Großteil der Spoilerklasse Passage deskriptiv sein. Dennoch könnten nicht-deskriptive Nominal-, Adjektiv- oder Verbalphrasen dabei sein, die in ihrer Länge über

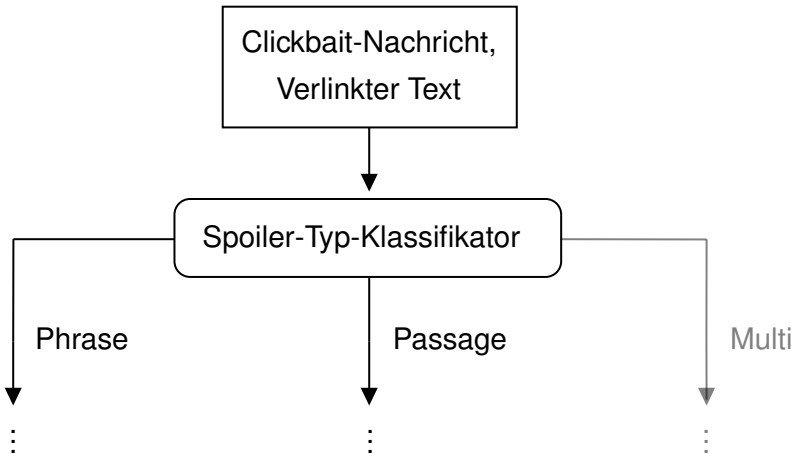


Abbildung 22: Klassifizierungsteil unseres Lösungsansatzes. Hierbei entscheidet ein Spoiler-Typ-Klassifikator mit Hilfe der Clickbait-Nachricht und dem verlinkten Text, ob der Spoiler des Clickbaits zur Klasse Phrase, Passage, oder Multi gehört. Da wir uns nur auf die Klassen Phrase und Passage konzentrieren, ist der Klassifikatorzweig zu Multi ausgegraut.

der Grenze von fünf Wörtern liegen, wie der Buchtitel ‚The Inevitable Death of Mister & Pete' oder der Eigenname ‚Palmer's Cocoa Butter Formula Skin Therapy Oil'. Die Klasse Multi zeichnet aus, dass die Spoiler aus mehreren Teilen bestehen, beispielsweise aus einem Spoiler der Klasse Phrase und einem der Klasse Passage. Dies ist typisch für Auflistungen von Fakten, wie in den ersten beiden Beispielen in Abbildung 19. Zudem kann eine Clickbait-Nachricht auch mehrere Curiosity Gaps aufweisen, wie im dritten Beispiel in Abbildung 19 kursiv hervorgehoben ist. Diese Art von Clickbait-Nachrichten ähneln einer Multi-Hop-Frage aus HotPotQA (Abbildung 10). Ein Spoiler der Klasse Multi kann somit jede Kombinaton der Klasse Phrase, Passage und Multi annehmen.

Um eine Idee davon zu bekommen wie die Klassifikation auf Clickbait-Nachrichten aussieht, testen wir einfachste Ansätze der Klassifikation. Wir beginnen mit Hilfe eines Naive-Bayes-Klassifikators, einer Bag-Of-Words-Repräsentation der Clickbait-Nachricht und einer Stichprobe von 500 Clickbait-Posts aus unserem Webis-Clickbait-Spoiling-20-Datensatz. Wir teilen die Stichprobe in 400 Clickbait-Posts zum Trainieren und 100 Clickbait-Posts zum Validieren der Klassifikationsansätze. Der Teil zum Validieren besitzt 44 Spoiler der Klasse Phrase, 42 der Klasse Passage und 14 der Klasse Multi. Wenn wir einen Klassifikator nehmen, der alle Spoiler als Phrase kategorisiert, erhalten wir also eine Accuracy von 44 % als Baseline für unseren Clickbait-Klassifikator, weil wir 44 % aller Einträge zur korrekten Klasse

zugeordnet haben. Der Naive-Bayes-Klassifikator aus der Python-Bibliothek `sklearn` mit Häufigkeiten von Wörtern als Features, schafft eine Accuracy von 52 %. Berechnen wir die TF-IDF-Werte für Wörter und benutzen diese statt der Häufigkeiten als Features, schafft der Klassifikator genauso eine Accuracy von 52 %. Eine typische vorgehensweise in Text Classification ist es, Stoppwörter, also Wörter, die häufig in Texten vorkommen, aber kaum bis gar keinen Mehrwert zum Inhalt des Textes bieten, sowie Punktierungen aus dem Text zu entfernen. Dazu nutzen wir die englische Stoppwortliste der Python-Bibliothek `nltk` und passen sie an die Aufgabe der Spoiler-Typ-Klassifikation an: Aus der Stoppwortliste entfernen wir die Wörter, die wir zur Klassifikation als wichtig erachten bzw. die typische Erkennungsmerkmale von Clickbait-Nachrichten sind. Dazu zählen Fragewörter (what, why, where, when, who, how), Personalpronomen (he, she, you) und das Wort ‚this'. Weiterführend werden alle Buchstaben der Wörter in Kleinbuchstaben umgewandelt und mit dem WordNet-Lemmatisierer aus `nltk` in Lemmata transformiert. Mit lemmatisierten Wörtern und entfernten Stoppwörtern kommt der Naive-Bayes-Klassifikator mit den Häufigkeiten der Wörtern genauso wie mit TF-IDF-Werten der Wörter auf eine Accuracy von 55 %. Auf dem 500er Stichprobendatensatz scheint die Klassifizierung der drei Klassen unsensibel gegenüber der TF-IDF-Repräsentation zu sein (0 % Verbesserung bzw. Verschlechterung), aber sensibel gegenüber der Entfernung von Stoppwörtern (3 % Verbesserung). In dieser Arbeit wollen wir uns aber nur auf das Spoiling der Spoilerklassen Phrase und Passage konzentrieren. Die Spoilerklasse Multi schient schwieriger umsetzbar zu sein, weil diese im Vergleich zu den Klassen Phrase und Passage die Extraktion mehrerer relevanter und potenziell thematisch unterschiedlicher Textabschnitte wie im dritten Beispiel in Abbildung 19 erfordert. Zeitlich gesehen passt sie somit auch nicht in den Rahmen dieser Arbeit. Wir beschließen dadurch die Klassen Passage und Multi als eine Klasse zu betrachten und führen ab jetzt eine Klassifizierung durch, die nur die Klasse Phrase von den anderen beiden Klassen unterscheiden soll.

Unter diesen neuen Umständen nutzen wir für unseren Naive-Bayes-Klassifikator TF-IDF-Werte von Wörtern der Clickbait-Nachrichten als Features. Die Wörter werden vorher lowercased, lemmatisiert und Stoppwörter entfernt. Der Stoppwörterliste haben wir Punktierungen beigefügt, das Fragezeichen und das Ausrufezeichen allerdings außen vorgelassen. Außerdem nehmen wir nur die TF-IDF-Werte der 160 meist vorkommenden Wörter zum Klassifizieren, wodurch unser Naive-Bayes-Klassifikator eine Accuracy von 72 % erreicht. Eine gängige Art von Features bei Klassifikationsaufgaben, sind Häufigkeiten von Wortarten (POS) statt Wörtern an sich. Wir wählen intuitiv POS-3-Gramme (Triplets) als Features, die wir uns anschauen wollen. Dabei erreichen wir die

Tabelle 14: Klassifizierungseffektivität vom Naive-Bayes-Klassifikator (NB) mit verschiedenen Features für Spoilerklassen Phrase, Passage und Multi gemessen in Accuracy (Acc): Baseline, die alle Spoiler in Klasse Phrase einordnet (Phrase); Häufigkeiten von Wörtern (#words); TF-IDF-Werte von Wörtern (tf-idf); TF-IDF-Werte von lowercased Wörtern nach Entfernung von Stoppwörtern (tf-idf_lem_stop).

Ansatz	Acc (%)
Phrase	44.0
$NB_{\#words}$	52.0
$NB_{tf\text{-}idf}$	52.0
$NB_{tf\text{-}idf\_lem\_stop}$	55.0

Tabelle 15: Klassifizierungseffektivität vom Naive-Bayes-Klassifikator (NB) mit verschiedenen Features für Unterscheidung der Klasse Phrase von Passage und Multi gemessen in Accuracy (Acc): Baseline, die die Spoiler nicht in die Klasse Phrase einordnet (Not-Phrase); POS-3-Gramme der 525 am häufigsten vorkommenden Wörter (POS-3_T525); TF-IDF-Werte der 160 am häufigsten vorkommenden lowercased Wörter nach Entfernung von Stoppwörtern (tf-idf_lem_stop_T160).

Ansatz	Acc (%)
Not-Phrase	56.0
$NB_{POS\text{-}3\_T525}$	65.0
$NB_{tf\text{-}idf\_lem\_stop\_T160}$	72.0

höchste Accuracy von 65 % mit den POS-3-Grammen der 525 meist vorkommenden Wörter ohne jegliche Textvorverarbeitung wie Lowercasing oder Entfernung von Stoppwörtern. Alle vorherigen Experimente sehen wir in Tabelle 14 und Tabelle 15 mit 72 % als höchste erreichte Effektivität beim Unterscheiden von Spoilern der Klasse Phrase von den Klassen Passage und Multi.

4.2 Vorgehen

Nach der Pilotstudie zum Klassifizieren experimentieren wir in diesem Abschnitt systematisch mit verschiedenen Klassifikationsalgorithmen und Wort- und POS-N-Grammen auf dem 1kTrain-Datensatz. Der 1kTrain-Datensatz besitzt 1000 Clickbait-Posts und ist balanciert auf 40 % Spoiler der Klasse Phrase und 60 % Spoiler der Klasse Passage bzw. Multi (Tabelle 13). Damit steigt die Baseline alle Spoiler in die Klasse Passage einzuordnen auf 60 % Accuracy. Wir nutzen auch weiterhin eine 80-20 Aufteilung der Stichprobe in Trainingssatz und Validierungssatz. Durchschnittliche Anzahl, genauso wie den Minimal- und Maximalwert, an Wörtern der Clickbait-Nachricht, des Spoilers und des verlinkten Textes im 1kTrain-Datensatz sehen wir in Tabelle 16 für jeweils Spoiler der Klasse Phrase und Passage. Für alle Klassifizierungsexperimente dieses Kapitels nutzen wir Algorithmen aus der Python-Bibliothek `sklearn`. Wir entscheiden uns für die Klassifizierungsalgorith-

Tabelle 16: Minimale (min.), Maximale (max.) und Durchschnittliche (ave.) Anzahl an Wörter in jeweils der Clickbait-Nachricht, Spoiler und dem verlinkten Text im 1kTrain-Datensatz für Spoiler der Klasse Phrase und Passage.

Spoilerklasse	Clickbait-Nachricht			Spoiler			verlinkter Text		
	min.	max.	ave.	min.	max.	ave.	min.	max.	ave.
Phrase	2	24	12	1	8	3	52	9,048	543
Passage	2	33	11	1	106	29	18	7876	689

men *Multinomial-Naive-Bayes*[1] (MultinomNB), *Linear-SVM*[2] (LinearSVM), *Radial-Basis-SVM*[2] (RbfSVM) und *Logistic-Regression*[3] (LogisticRegr). Der Multinomial-Naive-Bayes-Klassifikator ist ein Klassifikator, der gerne zu Beginn von Klassifikationsaufgaben getestet wird, weil er einfach zu verstehen und sehr kosteneffizient zu berechnen ist. Multinomial-Naive-Bayes berechnet, wie in Abschnitt 2.2 beschrieben, anhand des Satz von Bayes (Gleichung 1), zu welcher Wahrscheinlichkeit der Spoiler einer Clickbait-Nachricht zur Klasse Phrase und zur Klasse Passage gehört. Dazu werden die Wahrscheinlichkeiten, zu der die Wörter in der Clickbait-Nachricht auf einen Spoiler der Klasse Phrase und der Klasse Passage deuten, multipliziert. Beim Lernen werden die Wahrscheinlichkeiten zu jedem vorkommenden Wort in den Clickbait-Nachrichten des Datensatzes berechnet, mit der das Wort auf die jeweiligen Spoilerklassen Phrase und Passage deutet. Multinomial-Naive-Bayes setzt dabei voraus, dass die Wörter eine multinomiale Verteilung aufweisen. Die multinomiale Verteilung stellen wir uns als die Verteilung der Anzahlen der gewürfelten Seiten eines k-seitigen Würfels bei n-maligem Würfeln vor. Setzten wir $k{=}2$ und $n{>}1$, stellt die multinomiale Verteilung die binomiale Verteilung dar (mehrmaliges Werfen einer Münze).

Radial-Basis- und Linear-SVM basieren beide auf der Idee der Support Vector Machines (SVM). Im Falle der Linear-SVM wird beim Lernen auf einem Datensatz versucht, eine lineare Funktion (eine Gerade) zu finden, die zwei Klassen bestmöglich trennt, sodass auf beiden Seiten der Funktion so wenig wie möglich Instanzen der falschen Klasse existieren. Um gegen Ausnahmen bzw. Ausreißer in den Trainingsdaten vorzugehen, existiert bei SVMs ein Parameter C, dessen Wert für jeden Datensatz und jede Kombination an Features durch Testen gegen einen Validierungsteil des Datensatzes angepasst werden muss, ein sogenannter *Hyperparameter*. Die optimale Festlegung dessen Wertes kann

[1] https://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes
[2] https://scikit-learn.org/stable/modules/svm.html
[3] https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

die Klassifizierungseffektivität der SVM deutlich verbessern. Der Hyperparameter C, hat einen regularisierenden, also vereinheitlichenden Effekt auf die Trainingsdaten, wodurch Ausreißer beim Lernen, je nach Festlegung des Wertes, mehr oder weniger toleriert werden. Tolerieren von Ausreißern kann je nach Datensatz das Risiko von *Overfitting* mindern. Wenn beispielsweise ein Klassifikator den Trainingsdatensatz nahezu perfekt klassifizieren kann (gut gelernt), aber auf dem Validierungsdatensatz vergleichsweise sehr schlecht abschneidet (schlecht generalisierend), gilt der Klassifikator als overfittet. Je geringer C gewählt wird, desto weniger strebt die SVM danach, alle Instanzen (inklusive Ausreißer) der Trainingsdaten zu trennen und desto geringer wird das Risiko von Overfitting. Je höher C gewählt wird, desto mehr strebt die SVM danach, alle Instanzen in den Trainingsdaten zu trennen und desto höher wird das Risiko von Overfitting. Im Unterschied zur Linear-SVM nutzt die Radial-Basis-SVM statt einer linearen Funktion eine radiale Basisfunktion (Kurve), um die Trennlinie zwischen Klassen zu bestimmen. Deshalb existiert bei der Radial-Basis-SVM ein weiterer Hyperparameter γ. Je geringer γ gewählt wird, desto sensibler reagiert die SVM auf Ausreißer in den Trainingsdaten und desto höher wird das Risiko von *Overfitting* und umgekehrt. Der Logistic-Regression-Klassifikator ist der komplexeste der von uns genutzten Klassifikatoren, weil dieser die meisten Optionen zur Optimierung der Effektivität mitbringt.

Logistic Regression ist auch unter den Bezeichnungen Logit Regression, Maximum-Entropy Classification oder Log-Linear Classifier bekannt. Wie Linear-SVM ist der Logistic-Regression-Klassifikator ein linearer Klassifikator, aber beruht, statt auf einer linearen Funktion, auf der logistischen Funktion, die auch *Sigmoid-Funktion* genannt wird. Weiterhin existiert wie bei Linear-SVM der Hyperparameter C mit ähnlicher Wirkung. Mit dem *SAGA-Solver* (Defazio et al. 2014) besteht die Möglichkeit der *Elastic-Net-Regularisierung*. Elastic-Net-Regularisierung reguliert durch einen Parameter ρ die Stärke von ℓ_1- und ℓ_2-Regularisierung im gegenseitigen Austausch ($\rho \cdot \ell_1 + (1-\rho) \cdot \ell_2$). Dieser Parameter ρ darf nur Werte zwischen 0 und 1 annehmen. Dabei bedeutet ρ=1 nur ℓ_1-Regularisierung und ρ=0 nur ℓ_2-Regularisierung. Die ℓ_1-Regularisierung hilft irrelevante Features herauszufiltern, die ℓ_2-Regularisierung hilft korrelierenden Features entgegenzuwirken. Aufgrund von Scheinkorrelationen, die in den Trainingsdaten enthalten sind, kann ein Klassifikator falsche Regeln lernen, die außerhalb der Trainingsdaten eher zu schlechterer Klassifizierungeffektivität führt. Deshalb können korrelierende Features zu Overfitting führen und sind damit eher vermieden unerwünscht. (Hadi und Chatterjee, 2013).

Tabelle 17: Effektivität der Klassifikatoren für *Häufigkeiten von jeweils Wort- und POS-N-Grammen* als Features auf Clickbait-Nachrichten: *Keine Vorverarbeitung.* Alle Werte als Ergebnisse einer 10-Fold-Cross-Validation in Accuracy als Prozentwert angegeben. In der letzten Zeile ist die Anzahl an Features (#Features) für jedes N-Gramm zu sehen.

	Wort-N-Gramme				POS-N-Gramme			
Algorithmus	1	2	3	4	1	2	3	4
RbfSVM	69.8	64.8	61.7	60.6	66.3	68.8	67.8	61.4
LinearSVM	70.0	66.1	64.3	61.1	69.1	68.9	**69.3**	64.1
MultinomNB	**71.0**	66.1	64.3	60.8	64.8	69.3	67.9	63.6
LogisticRegr	70.1	**67.0**	63.8	61.0	**69.4**	**70.5**	68.9	64.0
#Features	3.1k	7.0k	7.4k	6.8k	42	694	2.7k	4.8k

Für die Hyperparameteroptimierung der Algorithmen nutzen wir die Klasse `GridSearchCV`. Damit führen wir eine erschöpfende Suche über die Hyperparameter für eigens gewählte Werte durch, um die beste Kombination an Hyperparametern für die genannten Klassifikatoren zu finden, die die Accuracy maximiert. `GridSearchCV` führt beim Evaluieren eine *10-Fold-Cross-Validation* durch. Dabei wird der Trainingsdatensatz in zehn Teile geteilt. Für jeden Teil werden die restlichen neun Teile zum Trainieren des Klassifikators verwendet und auf dem ausgewählten Teil evaluiert. Die Effektivität für jede Kombination an Hyperparametern wird als Durchschnitt der zehn Evaluierungen angegeben. Die Hyperparameterkombination mit der höchsten durchschnittlichen Effektivität gilt als die beste. Damit testen wir die Effektivität der Klassifikatoren für Häufigkeiten von jeweils Wort- und POS-N-Grammen als Features auf Clickbait-Nachrichten ohne Vorverarbeitung (Verteilungsverhältnisse in Anhang A erkennbar). In Tabelle 17 sehen wir die Effektivitäten von Accuracy in Prozentwerten angegeben. Die Zahlen lassen vermuten, dass die Klassifizierung von Clickbait am besten mit Wort-1-Grammen, Wort-2-Grammen, POS-1-Grammen, POS-2-Grammen und POS-3-Grammen funktioniert. Im nächsten Experiment schließen wir aus, dass Lowercasing, Lemmatisierung und Stoppwörterentfernung die Effektivität der Klassifikatoren verbessern, weil die Accuracy um 3-4 % abfällt (Tabelle 18). Das heißt, wir haben durch die Vorverarbeitung des Textes einige zum Klassifizieren relevante Features entfernt. Es heben sich nur noch Wort-1-Grammen, POS-1-Gramme und POS-2-Gramme als Features ab. Da wir mit Häufigkeiten arbeiten ist es wichtig, maximale Häufigkeiten von einzelnen Wörtern zu betrachten. Die maximalen Häufigkeiten können sich von Wort zu Wort stark unterscheiden und so den Anschein

Tabelle 18: Effektivität der Klassifikatoren für Häufigkeiten von jeweils Wort- und POS-N-Grammen als Features auf Clickbait-Nachrichten: mit *Lowercasing*, *Lemmatisierung* und *Stoppwortentfernung*. Alle Werte als Ergebnisse einer 5-Fold-Cross-Validation in Accuracy als Prozentwert angegeben. In der letzten Zeile ist die Anzahl an Features (#Features) für jedes N-Gramm zu sehen.

	Wort-N-Gramme				POS-N-Gramme			
Algorithmus	1	2	3	4	1	2	3	4
RbfSVM	68.0	63.1	61.0	60.4	**66.4**	65.3	63.5	62.3
LinearSVM	67.4	64.8	61.6	60.3	66.3	64.4	62.5	63.5
MultinomialNB	66.5	63.9	61.5	60.5	65.6	65.8	64.4	61.4
LogisticRegr	**68.6**	**64.9**	61.8	60.4	65.4	**66.8**	**64.5**	62.9
#Features	2.2k	4.0k	3.5k	2.7k	30	306	1.2k	2k

erwecken, dass ein Wort, das häufiger vorkommt, relevanter sein könnte. Von der Repräsentation der Clickbait-Nachrichten in TF-IDF-Werten sehen wir ab, weil diese die Absicht haben, seltener vorkommenden Wörtern mehr Relevanz zuzuschreiben. Aber gerade die oft vorkommenden Signalwörter in Clickbait-Nachrichten (this, why, where, how) sind entscheident für die Klassifikation derer Spoiler. Zudem wird in der Regel geraten mit normalisierten Werten zu arbeiten, da sich SVMs variabel gegenüber nicht normalisierten Daten verhalten[4]. Aus diesem Grund normalisieren wir die Häufigkeiten auf das Intervall $[0, 1]$ mit `QuantileTransformer`. Dadurch transformieren wir die Features auf eine uniforme Verteilung, sodass Ausreißer nicht mehr von den anderen Werten unterscheidbar sind. Den Einfluss der normalisierten Features sehen wir für alle N-Gramme in Tabelle 19. Eine Normalisierung führt zu keiner Verbesserungen und bei den POS-1-Grammen sogar zu einer Verschlechterung von rund 3 %. Da es Clickbait-Nachrichten gibt, deren Signalwörter, beispielsweise ‚how to', nicht eindeutig auf die Klasse des Spoilers schließen, müsste der Mensch in den verlinkten Text schauen, um auf die Klasse des Spoilers zu kommen. Für den Clickbait: ‚Here's how to cheat at Pokémon Go', ist in unserem Datensatz der Spoiler: ‚FakeLocation App', annotiert. Der Spoiler gehört damit zur Klasse Phrase, doch wenn wir ‚how to' lesen, denken wir meist an eine Anleitung als Spoiler. Um dieses Problem anzugehen, erweitern wir unsere Features um die TF-IDF-Werte der Wort-N-Gramme und POS-N-Gramme des verlinkten Textes, thematisch relevante Wörter des Textes hervorgehoben werden. Die IDF-Werte berechnen wir auf dem OpenWebText-Datensatz (Gokaslan und Cohen, 2019), einer Reddit-Post-Textsammlung, um

[4] https://scikit-learn.org/stable/modules/svm.html#tips-on-practical-use

Tabelle 19: Effektivität der Klassifikatoren für Häufigkeiten von jeweils Wort- und POS-N-Grammen alleinstehend als Features auf Clickbait-Nachrichten: Keine Vorverarbeitung, *normalisierte Häufigkeiten auf Intervall [0,1]*. Alle Werte als Ergebnisse einer 5-Fold-Cross-Validation in Accuracy als Prozentwert angegeben. In der letzten Zeile ist die Anzahl an Features (#Features) für jedes N-Gramm zu sehen.

	Wort-N-Gramme				POS-N-Gramme			
Algorithmus	1	2	3	4	1	2	3	4
RbfSVM	69.9	63.6	61.1	60.6	66.3	68.1	67.4	62.1
LinearSVM	69.0	66.3	63.9	61.0	66.3	67.8	67.3	64.3
MultinomialNB	**71.1**	**67.5**	64.5	60.6	64.3	68.9	68.4	63.5
LogisticRegr	70.4	67.1	63.9	62.6	**66.8**	**70.1**	**68.5**	63.5
#Features	3.1k	7.0k	7.4k	6.8k	42	694	2.7k	4.8k

eine möglichst gute Abschätzung zu den Vorkommen von Wörter in englischen Texten des Internets zu bekommen. Die Clickbait-Nachricht bestimmt hauptsächlich die Klasse des Spoilers. Den Text nutzen wir nur als Hilfe, um beispielsweise zu entscheiden, ob ‚how to' eine Anleitung oder einen Eigennamen als Spoiler erwartet. Dazu haben wir experimentell eine Gewichtung der Features von 4:1 für Clickbait-Nachricht zu verlinktem Text anhand der Accuracy-Scores bestimmt. Wir sehen in Tabelle 20 leichte Verbesserungen bei den Wort-1-Grammen und POS-1-Grammen als Features und leichte Verschlechterung bei Wort-2-Grammen, POS-2-Grammen und POS-3-Grammen. Da Wort-1-Gramme, POS-1-Gramme, POS-2-Gramme und POS-3-Gramme in den bisherigen Experimenten am besten einzeln abschneiden, nehmen wir diese jetzt zusammen als Features zum Klassifizieren. Anstelle der POS-1-Gramme nehmen wir Wort-2-Gramme als weitere Features, weil POS-1-Gramme durch die Wort-1-Gramme schon abgedeckt sein sollten und Wort-2-Gramme, wie ‚how many', zur Unterscheidung von Phrase und Passage sehr hilfreich sein dürften. Genauso fügen wir die Features: Länge der Clickbait-Nachricht und Länge des verlinkten Textes, hinzu. Damit beobachten wir die Effektivitäten der Klassifikatoren in Tabelle 21. Der Logistic-Regression-Klassifikator erreicht mit 74.9 % die beste Accuracy. Multinomial-Naive-Bayes schneidet, als einziger Klassifikator schlechter ab als nur mit normalisierten Worthäufigkeiten als Features.

Anhand dieses Ergebnisses untersuchen den Logistic-Regression-Klassifikator weiter. Weil der Klassifikator an erster Stelle unseres Lösungsansatzes steht, propagiert sich dessen Fehler zu den Spoiling-Modellen, wodurch das Gesamtsystem unzuverlässiger wird. Damit das nicht passiert opfern wir die An-

Tabelle 20: Effektivität der Klassifikatoren für Häufigkeiten von jeweils Wort- und POS-N-Grammen als Features auf Clickbait-Nachrichten + *TF-IDF-Werte auf verlinkten Texten*: Keine Vorverarbeitung, normalisierte Häufigkeiten auf Intervall [0,1], *4:1 Gewichtung* zwischen Clickbait-Nachricht und verlinktem Text. Alle Werte als Ergebnisse einer 5-Fold-Cross-Validation in Accuracy als Prozentwert angegeben. In der letzten Zeile ist die Anzahl an Features (#Features) für jedes N-Gramm zu sehen.

	Wort-N-Gramme				POS-N-Gramme			
Algorithmus	1	2	3	4	1	2	3	4
RbfSVM	**72.0**	60.9	60.4	60.4	67.1	**69.1**	68.1	61.2
LinearSVM	69.0	66.1	64.1	61.0	**68.1**	66.8	67.8	64.1
MultinomialNB	68.0	65.4	60.0	60.0	63.6	69.0	68.6	65.8
LogisticRegr	71.3	**66.1**	64.4	61.9	67.6	69.0	**69.4**	64.5
#Features	45k	277k	482k	553k	84	2.1k	20k	92k

Tabelle 21: Effektivität der Klassifikatoren für Features: *Wort-1-, Wort-2-, POS-2- und POS-3-Gramm-Häufigkeiten zusammen* für Clickbait-Nachrichten und TF-IDF-Werte für den verlinkten Text und *Länge der Clickbait-Nachricht und des verlinkten Textes*. Es findet keine Vorverarbeitung statt, Features sind auf Intervall [0,1] normalisiert, Clickbait-Nachricht und verlinkter Text sind 4-1 gewichtet. Alle Werte sind Ergebnisse einer 5-Fold-Cross-Validation.

Algorithmus	RbfSVM	LinearSVM	MultinomialNB	LogisticRegr
Accuracy (%)	73.9	73.3	69.6	**74.9**

zahl klassifizierter Clickbait-Nachrichten (Recall), um die Zuverlässigkeit des Klassifikators zu maximieren (Precision). Dafür betrachten wir nur Clickbait-Nachrichten deren Spoilerklasse mit hoher Sicherheit vorhergesagt wird. Der Logistic-Regression-Klassifikator errechnet für jede Clickbait-Nachricht eine Wahrscheinlichkeit, zu der dessen Spoiler in die Klasse Phrase oder Passage einzuordnen ist. Für diese Wahrscheinlichkeit legen wir einen Schwellwert fest, den es zu erreichen gilt. Sobald diese errechnete Wahrscheinlichkeit für eine Klasse den Schwellwert übersteigt, akzeptieren wir die vorhergesagte Klasse. Wie genau der Logistic-Regression-Klassifikator Spoilerklassen abhängig vom Schwellwert vorhersagt, sehen wir in Tabelle 22. Um die Zuverlässigkeit zu maximieren wählen wir für die Spoilerklasse Phrase den Schwellwert 80 %. Damit sind die vorhergesagten Spoiler zu 95 % korrekt, welche dafür aber nur 23.5 % aller Clickbait-Nachrichten ausmachen. Bei der Klasse Passage sind 21 % der vorhergesagten zu 92.6 % zuverlässig bei einem Schwellwert von 95 %.

Tabelle 22: Zuverlässigkeit (Precision) und Anzahl klassifizierter Spoiler (Recall) des besten Logistic-Regression-Klassifikators aus Tabelle 21. Abhängig vom Schwellwert, der die Wahrscheinlichkeit angibt, mit der der Klassifikator die Spoilerklasse vorhersagt. Alle Angaben in Prozent.

Schwellwert		60	70	80	90	95
Phrase	Precision	75.0	80.0	**95.0**	90.0	–
	Recall	44.4	29.6	**23.5**	11.1	0
Passage	Precision	78.4	80.6	83.8	86.7	**92.6**
	Recall	76.5	66.4	52.1	32.8	**21.0**
Accuracy		77.4	80.5	86.2	87.3	92.6

4.3 Menschliche Effektivität

Um einschätzen zu können, wie gut der Klassifikator ist, hilft es die Effektivität des Menschen beim Klassifizieren des Validierungssatzes zu betrachten. Dafür lassen wir drei Personen, die 200 Clickbait-Nachrichten des Validierungssatzes ‚ohne dass dies den verlinkten Text gesehen haben, in die Klassen Phrase und Passage einordnen. Jeder Person erklären wir vorher, wie die Einordnung in die Klassen Phrase und Passage gedacht ist (Abschnitt 3.4). In Tabelle 23 sehen wir die Effektivität der Klassifikationen durch die drei Personen (P_1, P_2 und P_3), wobei Person P_1 der Ersteller des Datensatzes ist, und einen Majority-Vote-Klassifikator (Major), der die Klasse wählt, die mindestens zwei der drei Personen auch gewählt haben. Das Korrelationsmaß *Fleiss'* κ gibt uns Aufschluss über die Ähnlichkeit der durch mehrere Annotatoren angegebenen Spoilerklassen. Mit dem Wert von Fleiss' κ können wir einschätzen, ob die Spoilerklassen in der Form akzeptierbar sind oder überdacht werden sollten.

Tabelle 23: Effektivität von drei Personen (P_1, P_2 und P_3) und ein Majority-Vote der Personen (Major) beim Klassifizieren auf dem Validierungssatz (200 Clickbait-Nachrichten). Der Majority-Vote bedeutet: Es wird die Klasse gewählt, die mindestens zwei der drei Personen auch gewählt haben. Cohen's d als Korrelationsmaß zwischen der jeweiligen Person und den annotierten Klassen im Validierungssatz angegeben.

Person	P_1	P_2	P_3	Major
Accuracy (%)	87.5	81.0	76.5	85.5
Cohen's d (%)	78.7	63.1	49.1	69.7

Fleiss' κ der drei Personen und den annotierten Klassen im Validierungssatz entspricht 62.7 %. Nach der Arbeit von Landis und Koch (1977) liegt unser Wert für Fleiss' κ an der unteren Schwelle von ‚wesentlicher Übereinstimmung' der Klassen zwischen den drei Personen und den annotierten Klassen im Validierungssatz. Damit nehmen wir an, dass unsere Annotationen der Klassen Phrase und Passage in Ordnung sind, aber dennoch einen Raum für Verbesserungen haben. Je nach Clickbait-Nachricht und verlinktem Text, können Personen, aufgrund ihres subjektiven Empfindens bezüglich des angesprochenen Themas und der Wortwahl der Clickbait-Nachricht verschiedene Spoiler erwarten. Aus Tabelle 23 entnehmen wir den Majority-Vote: unsere menschliche Effektivität beim Klassifizieren der Spoilerklassen Phrase und Passage. Diese beträgt 85.5 % Accuracy, womit noch genug Raum für Verbesserung des Klassifikators existiert.

Kapitel 5

Clickbait-Spoiling

In diesem Kapitel befassen wir uns mit dem Spoiling-Teil unseres Lösungsansatzes (Abbildung 23), der dem Spoilern von Clickbait jeweiligs der Klassen Phrase und Passage dienen soll. Wir testen verschiedene vortrainierte Sprachverständnismodelle aus der Huggingface-Bibliothek, wie ALBERT (Lan et al., 2019) und RoBERTa (Liu et al., 2019), auf ihre Effektivität beim Extrahieren von Spoilern jeweils für Clickbait der Klasse Phrase und Passage. Dafür nutzen wir drei Feintuning-Ansätze und unterschiedliche Modellgrößen zum Erzielen der besten Ergebnisse. Zum Vergleich testen wir auch wie effektiv diese Sprachverständnismodelle bei Feintuning auf beiden Klassen von Clickbait sind und ermitteln, wie gut ein Klassifikator Clickbait der Klasse Phrase und Passage vorhersagen müsste, um mit einer Kombination aus zwei Modellen besser zu sein.

5.1 Pilotstudie

Wir beginnen mit einer Pilotstudie, indem wir den AllenAI-Document-QA-Ansatz (Clark und Gardner, 2018) auf Clickbait-Nachrichten aus 1kTrain spoilern lassen. Der AllenAI-Document-QA-Ansatz als erster uns aufgefallener Question-Answering-Ansatz mit gut dokumentiertem Code war 2017 ein kleiner Druchbruch in Effektivität auf dem Datensatz TriviaQA. Mit einer paragraphenübergreifenden Normalisierung (Shared-Norm) zur Berechnung von Antworten gelang damit den Autoren Clark und Gardner ein großer Sprung im State-Of-The-Art-Ergebnis für TriviaQA – Mit einem F1 von 71.3 % erzielten sie eine Verbesserung von rund 16 F1-Punkten. Auf SQuADv1.1 gelangt der AllenAI-Document-QA-Ansatz auf einen F1 von 80.2 %. Bei der manuellen Auswertung der Spoiler fällt auf, dass die richtig vorhergesagten Spoiler

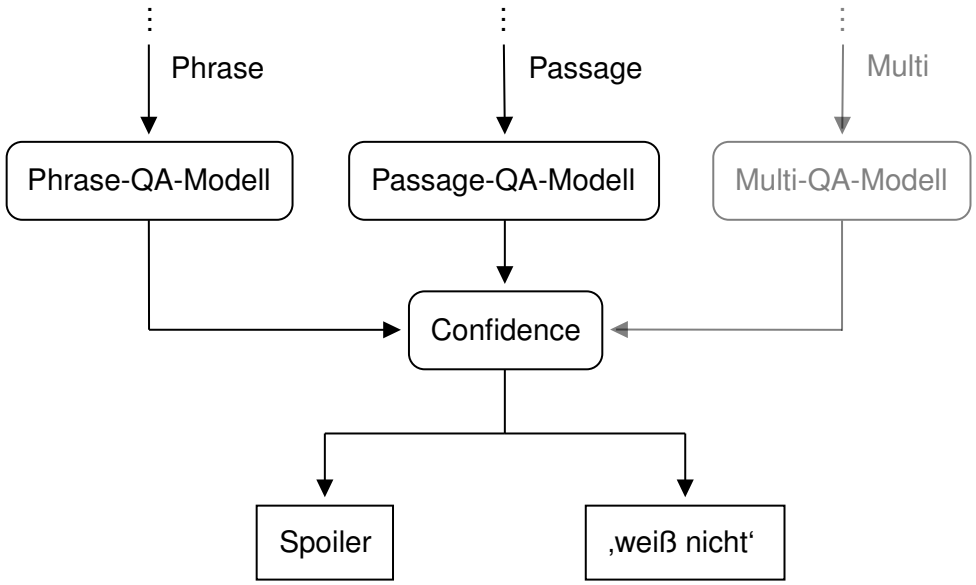


Abbildung 23: Der Spoiling-Teil unseres Lösungsansatzes. Zum Spoilern gehen wir davon aus, dass wir die Spoilerklasse des Clickbait-Posts kennen und behandeln die Klassen Phrase und Passage unabhängig von einander. Die Klasse Multi betrachten wir nicht und ist deshalb ausgegraut. Für die jeweiligen Klassen feintunen wir ein Modell im Question-Answering-Stil (QA-Modell).

des AllenAI-Document-QA-Ansatzes meist sehr kurz sind (zwischen ein bis vier Wörtern). Um diese Behauptung zu verifizieren, müssen wir die Spoiler des AllenAI-Document-QA-Ansatzes für den gesamten 1kTrain-Datensatz auswerten. Da manuelle Auswertung sehr zeitintensiv ist, informieren wir uns zu automatisierten Methoden bzw. *Evaluationsmetriken* zum Vergleichen zweier Textabschnitte (unseres Spoilers mit dem vom AllenAI-Document-QA-Ansatz extrahierten Spoiler) und wählen passende Metriken für das Vergleichen zweier Spoiler.

5.2 Evaluationsmetriken

Um die Effektivität des Spoilerns von Clickbait zu messen, benötigen wir eine Metrik, die die von beispielsweise dem AllenAI-Document-QA-Verfahren extrahierten Spoiler mit unseren annotierten Spoilern in Webis-Clickbait-Spoiling-20 vergleicht. Die Metrik soll ein Maß dafür sein, wie ähnlich sich zwei Textabschnitte sind. Bei Texten unterscheiden wir zwischen lexikalischer (gleiche oder ähnliche Wörter) und semantischer Ähnlichkeit (gleiche oder ähnliche Bedeutung). Für die Ähnlichkeit von Spoilern ist es vor allem wichtig, deren

semantische Ähnlichkeit zu beachten. In den verlinkten Texten von Clickbait-Nachrichten ist es nicht unüblich, dass mehrere Textabschnitte existieren, die die Clickbait-Nachricht ähnlich gut spoilern. Dadurch ist der annotierte Spoiler nicht als einzig richtig anzusehen, sondern als Vorgabe für die Bedeutung, die ein extrahierter Spoiler aufweisen muss. Trotzdem haben wir beim Annotieren der Spoiler versucht, den bestmöglichen Spoiler zu finden, wodurch ein exakt gleich extrahierter Spoiler (lexikalisch identisch) auch die beste Bewertung (Score) bekommen sollte. Für die semantische Ähnlichkeit muss z.B. auf Wörter mit gleicher oder ähnlicher Bedeutung (Synonyme), die Reihenfolge von Wörtern und Paraphrasierungen geachtet werden. Das Erkennung von Paraphrasierungen gehört hierbei zu den schwerer umzusetzenden Eigenschaften eines Ähnlichkeitsmaßes, weil die Zusammengehörigkeit von Wörtern richtig erkannt und auch richtig gedeutet werden muss. Ein weiterer wichtiger Aspekt ist die Beachtung von Verneinungen, da diese die Bedeutung eines Spoilers negieren und somit von richtig auf falsch kippen. In den folgenden Unterabschnitten schauen wir uns dafür die Evaluationsmetriken *BLEU-4* (Unterabschnitt 5.2.1), *METEOR* (Unterabschnitt 5.2.2) und *BERTScore* (Unterabschnitt 5.2.3) an.

5.2.1 BLEU-4

BLEU steht für *Bilingual Evaluation Understudy* und wurde mit der Intention entwickelt, maschinell übersetzte Sätze mit professionell übersetzten Sätzen vom Menschen zu vergleichen (Papineni et al., 2002). Dabei sollte die Metrik so funktionieren, dass je ähnlicher der maschinelle Satz dem professionell übersetzten menschlichen Satz ist, desto höher der Wert der Metrik (Score). BLEU ist eine lang bewährte Metrik und zählt heutzutage zu den populärsten Metriken zur Evaluation von maschinell übersetzten Texten, da es eine der ersten Metriken mit hoher Korrelation zur Auswertung maschineller Übersetzungen durch den Menschen war und einfach und schnell zu berechnen und verstehen ist. Zum Vergleichen eines Kandidatensatzes C mit Referenzsätzen nutzt BLEU eine *modifizierte N-Gram-Precision* p_n. Hierbei werden alle Vorkommen von unterschiedlichen N-Grammen des Kandidatensatzes so oft gezählt, wie sie maximal in einem der Referenzsätze vorkommen ($Count_{clip}$) und durch die Anzahl der N-Gramm-Vorkommen im Kandidatensatz geteilt (Gleichung 3).

$$p_n = \frac{\sum_{ngram \in C} Count_{clip}(ngram)}{\sum_{ngram \in C} Count(ngram)} \tag{3}$$

Wenn ein N-Gramm drei Mal in einem Kandidatensatz vorkommt, aber nur maximal zwei Mal ein einem der Referenzsätze, so wird für dieses N-Gramm im Zähler die Ziffer zwei aufsummiert und im Nenner die Ziffer drei. Der Kandidatensatz ‚Die Maus die Maus die Maus.' enthält nach Lowercasing und Entfernen von Punktierung zwei unterschiedliche 2-Gramme (‚die maus' und ‚maus die'), wobei das 2-Gramm ‚die maus' ganze drei mal vorkommt und ‚maus die' zwei mal. Bezogen auf den Referenzsatz ‚Die Maus weiß, wo die Maus ist.' (auch lowercased und ohne Punktierung betrachtet) zählen wir für p_2 das 2-Gramm ‚die maus' nur zwei mal, weil es auch nur zwei mal im Referenzsatz enthalten ist. Wenn wir das für die beiden 2-Gramme vom Kandidatensatz machen, ergibt sich für dieses Beispiel $p_2 = \frac{2+0}{3+2} = 0.4$ und analog $p_1 = \frac{2+2}{3+3} = 0.67$. Für den BLEU-Score werden die modifizierten N-Gram-Precisions als geometrisches Mittel mit uniformen Gewichten w_n (Summe aller Gewichte ergibt 1) kombiniert. BLEU wird dadurch oft auch als BLEU-N bezeichnet, wobei N für die N-Gramme steht, auf welchen BLEU berechnet wird. Demnach steht BLEU-2 für den Score auf 1- bis 2-Grammen bzw. Wortpaaren und BLEU-4 für den Score auf 1- bis 4-Grammen bzw. Wort-4-Tupeln. Dadurch werden bei BLEU soweit zu lange Kandidatensätze bestraft und auf Wortwahl und -reihenfolge (für N $\geq$ 2) geachtet. Da zu kurze Kandidatensätze auch beachtet werden müssen, wurde von Papineni et al. eine exponentielle Knappheitsstrafe BP (Gleichung 4) eingeführt, die sich aus der Länge des Kandidatensatzes c und der Länge des Referenzsatzes r, der der Länge des Kandidatensatzes am nächsten kommt, errechnet. Die Knappheitsstrafe BP geht multiplikativ in den Score ein, woraus sich final die Berechnungsvorschrift für BLEU in Gleichung 5.

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} \tag{4}$$

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{5}$$

Unser Kandidatensatz: ‚Die Maus die Maus die Maus.', erreicht damit einen BLEU-2 von 0.437 und einen BLEU-4 von $9 \cdot 10^{-155}$. BLEU-4 ist, wie wir sehen können, deutlich bestrafender, wenn es sich beim Kandidatensatz um ‚sinnlose' Wiederholungen von Wörtern des Referenzsatzes handelt.

Kritisiert wird an BLEU dessen Striktheit zu den im Kandidatensatz und in den Referenzsätzen benutzen Wörtern (Callison-Burch et al., 2006; Kané et al., 2019). Zum Kompensieren der Variationen an möglichen Wörtern (Synonyme, ...), Wortstellungen und Bedeutungen von Sätzen, berechnen Papineni et al. den Score mit Hilfe mehrerer Referenzsätze. Somit werden gültige Variationen, die nicht durch die Referenzsätze abgedeckt sind, bestraft.

extrahierter Spoiler	annotierter Spoiler	BLEU-2 (%)	korrekt?
16	September 16	36.8	×
igloo	colorful igloo	36.8	✓

Abbildung 24: Beispiele für starke Bestrafung zu kurzer Spoiler, bei Spoilern der Klasse Phrase. In der Spalte ‚korrekt?', ob wir den extrahierten Spoiler als noch richtig (✓) oder als falsch (×) ansehen.

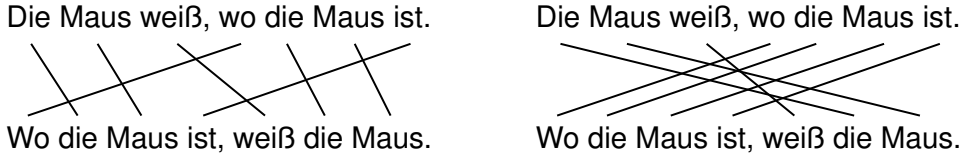
Vor allem mit nur einem Referenzsatz, wie bei uns mit nur einem annotierten Spoiler zum Vergleichen, werden Variationen kaum beachtet. Ein gutes Beispiel dafür ist der Vergleich vom annotierten Spoiler ‚boy' mit dem extrahierten Spoiler ‚little brother'. Wenn wir dafür BLEU-1 berechnen, erhalten wir einen Score von 0.0, obwohl es sich hierbei eher um einen korrekten Spoiler handelt. Hinzu kommt, dass eine Menge an ungültigen Variationen existiert, die einen ähnlichen BLEU-Score erzeugen. Außerdem kann BLEU keine Negation erkennen (Kané et al., 2019), da dazu nur die Änderung oder Addition eines Wortes zum Satz notwendig ist. Da die restlichen Wörter identisch bleiben, bleibt auch die Anzahl an N-Gramm-Vorkommen ähnlich und der BLEU-Score bleibt somit ähnlich hoch oder niedrig, obwohl die Bedeutung gekippt ist. In der Arbeit von Callison-Burch et al. zeigen die Autoren, dass aufgrund der fehlenden expliziten Bedingungen an die vorkommenden N-Gramme, die N-Gramme, die im Kandidaten- und Referenzsatz vorkommen, im Kandidatensatz beliebig permutiert werden können, solange deren Anzahl unverändert bleibt. Dadurch sind die BLEU-Scores dieser Permutationen sehr ähnlich, obwohl die meisten davon als grammatikalisch falsch gelten würden. Bezüglich des Vergleichens von Spoilern sollte das weniger ein Problem darstellen, da wir die Spoiler aus überwiegend grammatikalisch korrekten Texten extrahieren lassen. Ein weiterer Kritikpunkt an BLEU ist, dass dieser ohne genaueste Angaben über die Umstände der Berechnung des BLEU-Scores schlecht vergleichbar ist (Post, 2018). Einen Einfluss auf den BLEU-Score haben zuallererst die N-Gramme, dann die verschiedenen Vorverarbeitungsschritte, wie Lowercasing, Lemmatisierung und vor allem Tokenisierungsverfahren. Ob wir Zeichen oder Wörter als Tokens nutzen, bestimmt, welche N-Gramme bei der Berechnung beachtet werden und hat dadurch eine Auswirkung auf den Score. Lowercasing und Lemmatisierung kann genutzt werden, um potenziell mehr N-Gramm-Vorkommen zu erzielen, durch deren kanonisierende Wirkung auf den Text. Beobachtbar bei uns ist auch die starke Bestrafung von kürzeren Spoilern bei Spoilern der Klasse Phrase, aufgrund des höheren Anteil eines Wortes am Spoiler Abbildung 24.

Zum Berechnen unseres BLEU-4-Scores zwischen zwei Spoilern nutzen wir die Funktion `sentence_bleu` und den Tokenizer aus der Python-Bibliothek `nltk`. Weiterhin führen Lowercasing, Lemmatisierung (WordNet-Lemmatizer) und Entfernung von Punktierung als Vorverarbeitungsschritte vor der Berechnung des Scores durch. Für Spoiler, die kürzer als vier Wörter sind, berechnen wir BLEU-N, bei dem N für die Anzahl an Wörtern im Spoiler steht.

5.2.2 METEOR

METEOR (*Metric for Evaluation of Translation with explicit Ordering*) (Banerjee und Lavie, 2005) wurde genau wie BLEU für die Evaluation von maschinell übersetztem Text entwickelt, aber mit der Absicht, einige Probleme von BLEU zu beheben und vor allem die Zuverlässigkeit auf Satzebene (Vergleich nur mit einem Referenzsatz) zu verbessern. Die Hauptidee bei METEOR ist es, eine *Wort-zu-Wort-Angleichung* zwischen Kandidaten- und Referenzsatz zu erstellen, bei der jedem Wort vom Kandidatensatz maximal ein Wort des Referenzsatzes zugeordnet wird. Die Zuordnung erfolgt, wenn zwei Wörter gleich sind, deren Wortstamm gleich ist oder die Wörter Synonyme sind. Aus den Zuordnungen der Wörter werden alle möglichen Wort-zu-Wort-Angleichungen erstellt. Als beste wird die Angleichung gewählt, die die wenigsten *Überkreuzungen* zwischen Zuordnungen der Wörter besitzt (siehe Abbildung 25). Der METEOR-Score berechnet sich folglich aus der besten Angleichung. Aus der Anzahl von zugeordneten Wortpaaren m in der besten Angleichung, der Anzahl an Wörtern im Kandidatensatz t und der Anzahl an Wörtern im Referenzsatz r wird die Precision $P = \frac{m}{t}$ und der Recall $R = \frac{m}{r}$ des Kandidatensatzes berechnet. Daraus wird ein parametrisiertes harmonisches Mittel F_{mean} mit Werten von 0 bis 1 berechnet:

$$F_{mean} = \frac{P \cdot R}{\alpha \cdot P + (1 - \alpha) \cdot R} \tag{6}$$



(a) Besser **(b)** Schlechter

Abbildung 25: Beispiel für eine Wort-zu-Wort-Angleichung bei METEOR mit (a) geringsten Überkreuzungen und (b) meisten Überkreuzungen.

Im Gegensatz zur Knappheitsstrafe BP in BLEU wird der Recall bei METEOR direkt abgehandelt. Zum Beachten von Wortreihenfolgen wird bei METEOR das Bestrafungsmaß Pen eingeführt. Dieses errechnet sich aus der besten Angleichung, indem der Kandidatensatz in ch-viele längste zusammengehörige Abschnitte, die auch so an der gleichen Stelle im Referenzsatz vorkommen, geteilt wird. Im Beispiel der Abbildung 25a existieren keine zusammengehörigen Abschnitte im Kandidatensatz, länger als ein Wort sind und an der gleichen Stelle wie im Referenzsatz stehen. Unser Beispiel wird also in sieben Abschnitte geteilt. Die Berechnung von Pen nutzt die Anzahl der zusammengehörigen Abschnitte ch und die Anzahl von zugeordneten Wortpaaren m (Gleichung 7). Damit wird der METEOR-Score aus dem Produkt von Pen und F_{mean} berechnet (Gleichung 8).

$$Pen = \gamma \cdot \left(\frac{ch}{m}\right)^{\beta} \tag{7}$$

$$\text{METEOR-Score} = (1 - Pen) \cdot F_{mean} \tag{8}$$

Wenn wir α=0.75, β=1.4 und γ=0.45 wählen, ergibt sich für unser Beispiel aus Abbildung 25a F_{mean}=1 und METEOR-Score=0.55, weil die Anordnung der Wörter gar nicht übereinstimmt. Da METEOR wie BLEU die Angabe mehrerer Referenzsätze unterstützt, wird für den finalen METEOR-Score für jeden Referenzsatz ein Score berechnet und der beste daraus ausgewählt. Dadurch, dass wir unsere annotierten Spoiler eins zu eins extrahiert dem verlinkten Text extrahiert haben, können wir davon ausgehen, dass die annotierten Spoiler auch grammatikalisch korrekt sind. Die Bestrafung für die andere Anordnung der Wörter in den zu vergleichenden Spoilern ist für uns damit irrelevant, solange die Bedeutung derer des annotierten Spoilers ähnelt. Bei unserer Auswertung von extrahierten Spoilern lassen wir deshalb das Bestrafungsmaß Pen weg (Das Setzen von γ=0 hat den gleichen Effekt) und nutzen ausschließlich F_{mean}. Ab METEOR-Version 1.3 (Denkowski und Lavie, 2011) unterstützt METEOR, mit Hilfe einer Tabelle zum Nachschlagen, das Angleichen von Paraphrasen und es findet eine Unterscheidung von funktionalen und inhaltlichen Wörtern statt. Letzteres hilft dabei die Abwesenheit weniger relevanter Wörter im Kandidatensatz weniger zu bestrafen und sich so mehr auf die Bedeutung zu konzentrieren. Die Parameter α, β, γ wurden von Denkowski und Lavie mit einer Reihe an Datensätzen experimentell so bestimmt, dass diese maximal mit der menschlichen Auswertung der Datensätze korreliert.

Bisher scheint METEOR die deutlich bessere Wahl für die Evaluation der Ähnlichkeit von annotiertem und extrahiertem Spoiler. METEOR ist zudem auch an die jeweilige Aufgabe, die evaluiert werden soll, durch justieren der Gewichte und Parameter α, β, γ, anpassbar. Aber auch METEOR besitzt

Nachteile, die beachtet werden sollten. Ein Beispiel wäre der Vergleich von ‚Die Drogenhändler' und ‚Zwei Männer' (Chen et al., 2019). Da es sich bei den Drogenhändlern um zwei Männer handelt, sollten diese zwei Texte als sehr ähnlich ausgewertet werden. METEOR gibt den beiden Texten aber einen Score von 0, weil es weder Synonyme noch Paraphrasen sind. Weiterhin kann METEOR genauso wie BLEU Negationen nicht korrekt verarbeiten, da ein Kandidatensatz (‚Die Maus weiß, wo die Maus ist.') und dessen Negation (‚Die Maus weiß nicht, wo die Maus ist.') als sehr ähnlich ausgewertet werden.

Für unsere Auswertung von extrahierten Spoilern nutzen wir METEOR 1.5 (Denkowski und Lavie, 2014)[1] mit Paraphrasierungs-, Wortstamm- und Synonymabgleich und die dafür vorgegebenen Gewichte und Parameter für *Adequacy*[2], weil wir besonders auf die Bedeutung zweier Spoiler achten wollen.

5.2.3 BERTScore

BERTScore (Zhang et al. 2020a) wurde als Evaluationsmetrik für die Aufgabe der automatisierten Textgenerierung entwickelt. Zhang et al. haben dabei in der Arbeit zum BERTScore ihr Hauptaugenmerk auf die Evaluation semantischer Ähnlichkeiten gesetzt. Seit der Vorstellung von BERT (Devlin et al. 2019) und dessen Erzielen neuer State-Of-The-Art-Ergebnisse wird viel mit dem Modell und dessen Bestandteilen experimentiert. Vor allem die kontextualisierten Worteinbettungen haben maßgeglich zum Erfolg von BERT beigetragen. Diese dienen der Repräsentation von Wörtern abhängig von deren Kontext, wodurch die Wortbedeutung in gewissen Zügen auch abgebildet wird. Damit können insbesondere identische Wörter mit unterschiedlichen Bedeutungen wie die Bank zum Sitzen und die Bank als Kreditinstitut je nach Kontext unterschieden werden. Somit kamen Zhang et al. auf die Idee, die kontextualisierten Worteinbettungen von BERT als Basis für ihre Evaluationsmetrik BERTScore zu nutzen. Anders als bei den vorher behandelten Metriken wird bei BERTScore ein Ähnlichkeitsmaß zwischen den einzelnen Wörtern eines Kandidatensatzes $\hat{x}$ und eines Referenzsatzes x mit Hilfe vortrainierter Worteinbettungen von BERT berechnet. Der Kandidatensatz wird von BERT auf die Worteinbettungen $\hat{\mathbf{x}}$ und der Referenzsatz auf die Worteinbettungen $\mathbf{x}$ umgerechnet. Das Ähnlichkeitsmaß ist die Kosinusähnlichkeit auf den bereits normalisierten kontextualisierten Worteinbettungen ($\mathbf{x}^\top\hat{\mathbf{x}}$). Über die Ähnlichkeiten der Wörter werden die Precision P_{BERT} und der Recall R_{BERT} vom BERTScore berechnet. Für die Precision P_{BERT} wird jedes Wort $\hat{x}_i$ aus dem Kandidatensatz

[1] http://www.cs.cmu.edu/ãlavie/METEOR/

[2] meteor-1.5-wo-en-norm-0.75_1.4_0.45_0.7-ex_st_sy_py-1.0_1.0_0.6_0.8

mit dem Wort x_j aus dem Referenzsatz höchster Kosinusähnlichkeit gepaart und die resultierenden Kosinusähnlichkeiten aufsummiert (Gleichung 9). Der Recall R_{BERT} wird analog berechnet nur, dass dabei jedes Wort x_j aus dem Referenzsatz mit einem Wort $\hat{x}_i$ aus dem Kandidatensatz gepaart wird (Gleichung 10). Mit Presicion P_{BERT} und Recall R_{BERT} wird folglich ein F1-maß errechnet (Gleichung 11).

$$P_{\text{BERT}} = \frac{1}{|x|} \sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} \mathbf{x}_i^\top \hat{\mathbf{x}}_j \tag{9}$$

$$R_{\text{BERT}} = \frac{1}{|\hat{x}|} \sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} \mathbf{x}_i^\top \hat{\mathbf{x}}_j \tag{10}$$

$$F_{\text{BERT}} = 2\frac{P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}} \tag{11}$$

Weiterhin zitieren Zhang et al., dass seltene Wörter eher auf die Ähnlichkeit zweier Sätze hindeuten können, wodurch sie eine Skalierung der einzelnen Wörter über deren IDF-Werte für P_{BERT} und R_{BERT} vornehmen. Die IDF-Werte werden dabei auf den Referenzsätzen des Test-Datensatzes (bei uns wäre das 1kTrain-Valid) berechnet und sind damit für alle Ansätze eines Test-Datensatzes gleich, aber für verschiedene Test-Datensätze unterschiedlich. Insgesamt zeigt BERTScore eine höhere Korrelation mit menschlicher Auswertung bei Paraphrasierungsaufgaben und kann somit beispielsweise mit dem Vergleich von ‚Die Drogenhändler' mit ‚Zwei Männer' umgehen als METEOR. Ein weiterer Vorteil ist, dass sich der BERTScore aufgrund des Aufbaus auf BERT auch mit anderen BERT-ähnlichen Modellen wie z.B. RoBERTa berechnen lässt. Genauso lassen sich diese Modelle auf eine spezifische Aufgabenstellung feintunen, sodass noch höhere Korrelation erreichbar sind, da Wörter auch Fachgebietsspezifisch verschiedene Bedeutungen haben können.

Empirisch scheint BERTScore eine robustere und flexiblere Version von METEOR zu sein (Zhang et al., 2020a), vor allem bezüglich Paraphrasierung, aber das Problem der Negation kann der BERTScore ähnlich schlecht konfrontieren wie METEOR. Nach Chen et al. scheint METEOR zurzeit trotzdem die beste Wahl zur Evaluation auf Question-Answering-Datensätzen, hat aber weniger Potenzial als der BERTScore. Ein weiterer Nachteil betrifft den Aufwand zur Berechnung des BERTScore. Aufgrund der Nutzung von BERT-ähnlichen Sprachverständnismodellen, benötigt der BERTScore für die zuverlässigste Evaluation auch sehr viel Rechenleistung bzw. Speicherplatz für das genutze Modell. Unter anderem deshalb kann der BERTScore mit den meisten Modellen auch keine Sätze länger als 510 Wörtern verarbeiten. Zum

Vergleichen längerer Sätze kann, zum Preis höheren Speicherverbrauchs, auf das XLNet-Modell zurückgegriffen werden. Sollte jedoch nicht genug Speicherplatz vorhanden sein, kann auf kleinere Modelle ausgewichen werden. Die Konsequenz dabei ist aber, potenziell unzuverlässige Evaluationsergebnisse zu erhalten.

Zur Evaluation auf unserem Datensatz nutzen wir den BERTScore aus dem originalen GitHub-Repository der Autoren.[3] Dabei verwenden wir zur Berechnung das Modell ALBERT-XXLarge-V2[4] ohne IDF-Skalierung, weil unser Validierungsdatensatz nicht groß genug ist. Zudem verzichten wir auf die Skalierung für bessere Lesbarkeit der Scores (breitere Verteilung der Scores auf dem Interval [0,1]), weil die Scores ausreichend lesbar sind und die Skalierung in einigen Vergleichen von Spoilern negative Scores verursacht.

5.2.4 Fazit zu Evaluationsmetriken

Wir geben bei unseren folgenden Experimenten immer alle drei Metriken BLEU-4, METEOR-F_{mean} (METEOR-F) und BERTScore-F1 an. BLEU-4 nehmen wir, aufgrund dessen Striktheit zu den verwendeten Wörtern im annotierten Spoiler, als Indiz auf die Fähigkeit eines Modells die exakt gleichen Spoiler zu extrahieren wie die annotierten Spoiler. Bei der Berechnung von BLEU-4 für Spoiler, die kürzer sind als vier Wörter, berechnen wir BLEU-N mit N als die Länge des kürzeren Spoilers der beiden zu vergleichenden Spoiler. Die Effektivität eines Ansatzes errechnen wir aus dem Mittelwert der BLEU-4-Scores für jedes Paar des extrahierten und annotierten Spoilers. METEOR-F geben wir an, weil es robust gegenüber Synonymen und Wortbeugungen reagiert und die beste Korrelation mit menschlicher Auswertung für Question-Answering-Datensätze aufweist (Chen et al., 2019). Dabei betrachten wir den F_{mean}, weil wir andere Reihenfolgen von Wörtern, aufgrund der größtenteils grammatikalisch korrekten extrahierten Spoiler nicht bestrafen wollen und damit robuster gegenüber Wortreihenfolgen auswerten können. Den BERTScore nutzen wir, um die Bedeutung eines extrahierten Spoilers mit der Bedeutung des annotierten Spoilers zu vergleichen. Ein höherer BERTScore lässt uns vermuten, dass ein Modell die Bedeutung der annotierten Spoiler besser getroffen hat, weil BERTScore deutlich besser bei Paraphrasierungsaufgaben wie Bildbetitelung korreliert als METEOR (Zhang et al., 2020a).

[3] https://github.com/Tiiiger/bert_score

[4] albert-xxlarge-v2_L8_no-idf_version=0.3.9(hug_trans=4.7.0.dev0)

5.3 Vorgehen

Als Baseline für Clickbait der Klasse Phrase nehmen wir den AllenAI-Document-QA-Ansatz (Clark und Gardner, 2018) mit dessen Modell getuned auf TriviaQA mit Shared-Norm. Die Effektivität des Verfahrens auf Clickbait der Klasse Phrase sehen wir in Tabelle 24. Weil der AllenAI-Document-QA-Ansatz beim Spoilern von Clickbait der Klasse Passage außerordentlich schlecht abschneidet (5.2 % METEOR-F, Tabelle 24) entscheiden wir uns gegen diesen Ansatz als Baseline für Clickbait der Klasse Passage (Ergebnisse sind wahrscheinlich dem Mangel an deskriptiven Antworten im TriviaQA-Datensatz geschuldet). Somit eignet sich der Allenai-Document-QA-Ansatz speziell für factoide Antworten wahrscheinlich besonders gut. Stattdessen nehmen wir ein ALBERT-Modell (Lan et al., 2019) getuned auf SQuADv1.1 mit einer Effektivität von 89.9 % F1 auf SQuADv1.1 als Baseline für das Spoilern von Clickbait der Klasse Passage. Wie in Tabelle 24 erkennbar ist, spoilert ALBERT getuned auf SQuADv1.1 mit einem METEOR-F von 20.5 % deutlich besser als der AllenAI-Document-QA-Ansatz getuned auf TriviaQA. Daher nutzen wir das ALBERT-Modell als Baseline. Mit Hilfe der beiden Baselines AllenAI-Document-QA und ALBERT vergleichen wir die drei Evaluationsmetriken BLEU-4, METEOR-F und BERTScore-F1 mit der manuellen Auswertung in Tabelle 25. METEOR-F scheint der manuellen Auswertung am nahesten zu sein. In einer Untersuchung zu Evaluationsmetriken für Question Answering ergab sich METEOR auch als beste Metrik zum Auswerten von Question-Answering-Ergebnissen (Chen et al., 2019). Um die Scores greifbarer und besser interpretierbar zu machen, nehmen wir eine Binarisierung der Scores vor, damit wir abschätzen können, wie viele Spoiler ein Modell korrekt beantworten kann. Da das Spoilern der Spoilerklasse Passage deutlich schlechtere Ergebnisse aufweist, beginnen wir danach zuerst dieses zu

Tabelle 24: Vergleich der Effektivität von AllenAI-Document-QA feingetuned auf TriviaQA ($\text{AllenAI-QA}_{\text{TriviaQA}}$) und ALBERT feingetuned auf SQuADv1.1 ($\text{ALBERT}_{\text{SQuADv1.1}}$) beim Spoilern von Clickbait jeweils der Klasse Phrase und Passage: Die Ergebnisse sind auf dem Phrase-Teil (400 Einträge) bzw. dem Passage-Teil (500 Einträge) des 1kTrain-Datensatzes berechnet und in Prozent angegeben.

Klasse	Modell	BLEU-4	METEOR-F	BERTScore-F1
Phrase	$\text{AllenAI-QA}_{\text{TriviaQA}}$	35.3	38.0	57.4
	$\text{ALBERT}_{\text{SQuADv1.1}}$	17.6	13.7	37.0
Passage	$\text{AllenAI-QA}_{\text{TriviaQA}}$	0.8	5.2	18.6
	$\text{ALBERT}_{\text{SQuADv1.1}}$	8.5	20.5	33.1

Tabelle 25: Vergleich der Evaluationsmetriken BLEU-4, METEOR-F und BERTScore-F1 mit der manuellen Auswertung der Spoiler von AllenAI-Document-QA feingetunt auf TriviaQA (AllenAI-$QA_{TriviaQA}$) für Clickbait der Klasse Phrase und ALBERT feingetunt auf SQuADv1.1 ($ALBERT_{SQuADv1.1}$) für Clickbait der Klasse Passage.

	Manuell	BLEU-4	METEOR-F	BERTScore-F1
AllenAI-$QA_{TriviaQA}$	41.0	35.3	38.0	57.4
$ALBERT_{SQuADv1.1}$	21.5	8.5	20.5	33.1

verbessern und wenden dadurch gewonnene Erkenntnisse gegebenfalls auf die Spoilerklasse Phrase an. Danach bauen wir unseren Lösungsanatz aus Logistic-Regression-Klassifikator für unsere Spoilerklassen und den jeweiligen Modellen zum Spoilern der Spoilerklassen Phrase und Passage zusammen und vergleichen diesen zu einem Modell, dass wir auf Clickbait der Klasse Phrase und Passage gleichzeitig feintunen. Also der direkten Kokurrenz unseres Lösungsansatzes ohne Klassifikator.

5.3.1 Binarisierung der Scores

Anhand der Scores selbst ist nicht genau erkennbar, wie gut ein Modell wirklich spoilert, da wir nicht wissen, wie viele Spoiler das Modell wirklich korrekt extrahiert hat. Dafür legen wir für jede Metrik einen Schwellwert fest, der uns angibt, ab welchem Score ein Spoiler zuverlässig als korrekt angesehen werden kann. Scores über dem Schwellwert erhalten den Wert Eins, Scores darunter den Wert Null. Somit binarisieren wir die Scores und können so zum Zählen der korrekten Spoiler einfach alle Werte aufsummieren. Um den Schwellwert experimentell bestimmen zu können, werten wir manuell die extrahierten Spoiler von AllenAI-Document-QA für Spoiler der Klasse Phrase und von ALBERT für Spoiler der Klasse Passage aus und notieren, welche Spoiler davon korrekt und welche falsch sind. Dabei achten wir darauf, dass die Spoiler ähnlich gut zum annotierten Spoiler den Curiosity Gap der Clickbait-Nachricht schließen, da es, wie in Abschnitt 5.2 erwähnt, alternative aber inhaltlich ähnliche Spoiler im verlinkten Text geben kann. Danach untersuchen wir die Anzahl der extrahierten *falsch positiven* Spoiler (FP) und *falsch negativen* Spoiler (FN), indem wir Schwellwerte von 10 % bis 80 % in 10er Schritten wählen. Falsch positiv sind diejenigen Spoiler, die über dem jeweiligen Schwellwert und damit als korrekt angenommen werden, tatsächlich aber falsch sind. Falsch negativ hingegen sind diejenigen Spoiler, die unter dem jeweiligen Schwellwert und damit als falsch angenommen werden, tatsächlich aber korrekt sind. Unser An-

Tabelle 26: Verteilung der Anzahl an falsch positiv (FP) und falsch negativ extrahierten Spoilern abhängig vom Schwellwert für Spoiler der Klasse Phrase. Erhoben auf den 400 Clickbait-Posts mit Spoilern der Klasse Phrase aus 1kTrain.

Schwellwert	BLEU-4		METEOR-F		BERTScore-F1	
	FP	FN	FP	FN	FP	FN
10 %	11	11	18	7	238	0
20 %	7	14	16	7	234	0
30 %	7	14	14	9	165	1
40 %	2	27	8	13	59	6
50 %	**2**	**27**	2	28	24	14
60 %	2	30	3	31	11	25
70 %	1	33	**2**	**31**	6	36
80 %	1	34	0	37	**1**	**40**

spruch ist es den Schwellwert für BLEU-4, METEOR-F und BERTScore-F1 so zu wählen, dass möglichst wenig falsch positive Spoiler über dem Schwellwert zu finden sind. Damit können wir zukünftig mit hoher Sicherheit sagen, dass Spoiler mit Scores über den gewählten Schwellwerten korrekt sind, zusätzliche korrekt extrahierte Spoiler allerdings unentdeckt bleiben. Deshalb reden wir dabei von ‚Mindestanzahl' korrekter Spoiler. Wir achten darauf, dass auch die Summe der falsch positiven und falsch negativen Spoiler vertretbar klein, also die Korrelation mit manueller Auswertung möglichst hoch bleibt. In Tabelle 26 sehen wir die Verteilung der falsch positiven und falsch negativen Spoiler in Abhängigkeit vom Schwellwert für die drei Metriken und Clickbait der Klasse Phrase und in Tabelle 27 für Clickbait der Klasse Passage. Wir wägen ab, wie viele falsch negative Spoiler wir akzeptieren, um die Zahl der falsch positiven Spoiler zu verringern. So wählen wir für Clickbait der Klasse Phrase für BLEU-4 einen Schwellwert von 50 %, da wir keine sechs zusätzliche falsch negativen Spoiler in Kauf nehmen wollen, nur um die falsch positiven Spoiler um einen weiteren zu reduzieren. Für METEOR-F wählen wir einen Schwellwert von 70 % und für BERTScore-F1 einen von 80 %. Für Clickbait der Klasse Passage entscheiden wir uns entsprechend für die Schwellwerte 30 % für BLEU-4 70 %, für METEOR-F und 60 % für BERTScore-F1.

Tabelle 27: Verteilung der Anzahl an falsch positiv (FP) und falsch negativ extrahierten Spoilern abhängig vom Schwellwert für Spoiler der Klasse Passage. Erhoben auf den 500 Clickbait-Posts mit Spoilern der Klasse Passage aus 1kTrain.

Schwellwert	BLEU-4		METEOR-F		BERTScore-F1	
	FP	FN	FP	FN	FP	FN
10 %	5	44	168	15	399	0
20 %	3	48	67	27	325	3
30 %	**1**	**51**	31	35	134	21
40 %	0	55	15	39	18	38
50 %	0	60	9	42	5	51
60 %	0	64	4	47	**1**	**59**
70 %	0	66	**1**	**54**	0	66
80 %	0	66	0	61	0	73

5.3.2 Spoiling von Clickbait der Klasse Passage

Unsere erste Idee, um Verbesserung zu erzielen, ist es, aktuellere Ansätze für Question Answering zu testen. Wie in Kapitel 1 erwähnt, liegt der Trend im Nutzen vortrainierter Sprachverständnismodelle wie BERT (Devlin et al., 2019), die durch Feintuning ihrer Parameter auf verschiedenste Aufgaben des Natural Language Processings angepasst werden. Die Huggingface-Bibliothek bietet viele aktuelle Sprachverständnismodelle und auch Python-Skripte zum Feintunen dieser Modelle vorimplementiert an. Wir nutzen hierbei speziell die Python-Skripte zu Question-Answering[5] und folgende sieben Modelle zum Testen: *BERT-cased*, *BERT-uncased*, *ALBERT* (Lan et al. 2019), *ELECTRA* (Clark et al., 2020), *FunnelTransformer* (Dai et al., 2020), *MPNet* (Song et al., 2020) und *RoBERTa* (Liu et al., 2019).

Um die Ausgangslage einzuschätzen, feintunen wir die kleinste verfügbare Version (‚base' bzw. ‚small') der genannten Modelle auf SQuADv1.1 mit Hilfe des `run_qa.py` Skripts der Huggingface-Bibliothek. Das `run_qa.py` Skript bereitet die Daten von SQuADv1.1 vor und fügt einen sogenannten Question-Answering-Header an die Modelle an, damit diese auf die Aufgabenstellung des Question Answerings feingetunt werden können. Das Skript bietet zudem die Möglichkeit, diverse Hyperparameter für das Feintuning der Sprachverständnismodelle festzulegen. Wir feintunen dabei alle Modelle mit identischen Hyperparametern (Batchgröße: 16, Sequenzlänge: 384, Lernrate: 3e-5, Rest: Standardwert des `run_qa.py` Skripts) für drei Epochen bzw.

[5] https://github.com/huggingface/transformers/tree/master/examples/pytorch/question-answering

Tabelle 28: Effektivität der sieben ausgewählten Sprachverständnismodelle auf SQuADv1.1 gemessen in Prozent von SQuAD-F1: Jedes Modell ist trainiert für 3 Epochen (∼16,500 Trainingsschritte) mit Batchgröße 16, Sequenzlänge 384 und Lernrate 3e-5. Bei den Modellen handelt es sich bei jedem um die ‚base'- bzw. ‚small'-Versionen.

ALBERT	BERT-uncased	BERT-cased	ELECTRA
89.9	88.2	88.5	92.1

FunnelTransformer	MPNet	RoBERTa
92.0	92.8	92.1

Tabelle 29: 80 %-20 % Aufteilung des 1kTrain-Datensatzes in Trainings- (1kTrain-Train) und Validierungsdatensatz (1kTrain-Valid) und deren Verteilung der Spoilerklassen.

	Phrase	Passage	$\sum$
1kTrain-Train	320	398	718
1kTrain-Valid	80	102	182
$\sum$	400	500	900

∼16,500 Trainingsschritte (siehe Tabelle 28). Den 1kTrain-Datensatz ohne Spoiler der Klasse Multi teilen wir in 80 %-20 % Manier in einen Trainings- (1kTrain-Train) und einen Validierungsdatensatz (1kTrain-Valid) auf (Tabelle 29), um die Effektivität der feingetunten Modelle auswerten zu können. Wir nehmen diese feingetunten Modelle und betrachten, wie gut diese durch das Feintuning auf SQuADv1.1 die Spoilerklasse Passage spoilern können. Dafür evaluieren wir die feingetunten Modelle auf 102 Clickbait-Posts mit Spoilern der Klasse Passage aus 1kTrain-Valid (Passage-1kTrain-Valid) (Tabelle 30). Wir erkennen, dass RoBERTa hierbei am besten spoilert und drei bis vier BLEU-4 bzw. BERTScore-F1 Punkte mehr als ALBERT erzielt bei ähnlichem METEOR-F.

Um zu erfahren, wie effektiv ein Modell bezüglich einer bestimmten Problemstellung ist, feintunen Forscher üblicherweise dieses Modell auf dem Trainingteil eines Datensatzes speziell für diese Problemstellung. Für uns heißt das, wir feintunen die Modelle auf 398 Clickbait-Posts mit Spoilern der Klasse Passage aus 1kTrain-Train (Passage-1kTrain-Train). Zuerst passen wir jedoch die Daten des 1kTrain-Datensatzes an den Aufbau des SQuADv1.1-Datensatzes an, um den Dateneinleseprozess des `run_qa.py` Skriptes für SQuADv1.1 zu nut-

Tabelle 30: Effektivität der auf SQuADv1.1 feingetunten Modelle beim Spoilern der Spoilerklasse Passage. Gemessen auf Passage-1kTrain-Valid (102 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem im Tabellenkopf auch in Klammern angegebenen Schwellwert der Evaluationsmetrik.

Modell	BLEU-4 (>30)	METEOR-F (>70)	BERTScore-F1 (>60)
ALBERT	10.60 (13)	26.47 (13)	34.34 (16)
$\text{BERT}_{\text{uncased}}$	8.12 (11)	19.73 (10)	31.43 (12)
$\text{BERT}_{\text{cased}}$	9.96 (13)	21.47 (10)	32.52 (13)
ELECTRA	11.90 (15)	**28.36 (15)**	34.53 (19)
FunnelTrans	12.82 (16)	27.24 (16)	36.50 (18)
MPNet	12.50 (16)	26.39 (15)	34.23 (16)
RoBERTa	**14.47 (18)**	26.44 (18)	**37.28 (17)**

zen. Dazu schreiben wir das Python-Skript `sq_transformer.py`, das ein dem SQuADv1.1-Datensatz nachempfundenes JSON-Objekt erzeugt und dessen Schlüssel wie folgt mit Werten aus dem 1kTrain-Datensatz versieht. Die Frage ist für uns die Clickbait-Nachricht und die Antwort der Spoiler. In SQuADv1.1 wird die Frage mit Hilfe eines Paragraphen beantwortet, bei uns hingegen mit Hilfe des verlinkten Textes, der aus mehreren Paragraphen zuzüglich des Titels besteht. Das heißt, dem Schlüssel `context` ordnen wir alle Paragraphen und den Titel des verlinkten Textes konkateniert zu. Die Startposition des Spoilers im verlinkten Text errechnen wir, indem wir das erste Vorkommen des Spoilers im Text finden und dessen und die Position dessen ersten Zeichens im Text ermitteln. Als ID nehmen wir die UUID des entsprechenden Clickbait-Posts im 1kTrain-Datensatz. Nach der Umformung des 1kTrain-Datensatzes können wir das `run_qa.py` Skript zum Feintunen der Modelle auf 1kTrain nutzen. Genau wie beim Feintuning auf SQuADv1.1 nutzen wir auch hier Batchgröße 16, Sequenzlänge 384, Lernrate 3e-5 und Rest: Standardwert des `run_qa.py` Skripts der Huggingface-Bibliothek. Diesmal feintunen wir für 15 Epochen bzw. rund 1,200 Schritte. Jedes Modell erzielt dadurch generell bessere Ergebnisse als die auf SQuADv1.1 feingetunten Modelle (siehe Tabelle 31), auch wenn 398 Clickbait-Posts zum Trainieren von Sprachverständnismodellen sehr wenig sind. Auch erkennen wir, dass sich das FunnelTransformer-Modell hierbei als das beste Modell leicht abhebt gefolgt von MPNet, RoBERTa und ALBERT. Eine mögliche Erklärung zur besseren Effektivität der auf Passage-1kTrain-Train feingetunten Modelle ist, dass Sprachverständnismodelle, aufgrund der Neuronalen-Netzwerk-Basis, in der Regel zum Overfitting neigen.

Tabelle 31: Effektivität der auf Passage-1kTrain-Train (398 Einträge) feingetunten Modelle beim Spoilern der Spoilerklasse Passage. Gemessen auf Passage-1kTrain-Valid (102 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem im Tabellenkopf auch in Klammern angegebenen Schwellwert der Evaluationsmetrik.

Modell	Schritte	BLEU-4 (>30)	METEOR-F (>70)	BERTScore-F1 (>60)
ALBERT	300	20.41 (27)	33.22 (24)	42.49 (20)
$\text{BERT}_{\text{uncased}}$	200	12.10 (16)	24.95 (16)	37.12 (14)
$\text{BERT}_{\text{cased}}$	300	15.41 (19)	25.71 (17)	38.94 (18)
ELECTRA	500	16.81 (24)	33.22 (21)	37.90 (18)
FunnelTrans	200	**21.90 (27)**	35.53 (26)	**44.49 (24)**
MPNet	500	21.06 (29)	**35.93 (25)**	41.57 (22)
RoBERTa	500	21.24 (27)	31.46 (24)	42.47 (20)

Die auf SQuADv1.1 feingetunten Modelle lernen Regeln für Daten aus SQuADv1.1, welche nur für diesen Datensatz vorteilhaft und somit schlechter auf andere Datensätze übertragbar sind. Als weiteres Problem ist SQuADv1.1 ein Question-Answering-Datensatz und demnach spezialisiert auf Fragen. Clickbait lässt sich zwar in Fragen umformulieren, aber Clickbait-Spoiling ist scheinbar sich dennoch stärker vom Question Answering zu unterscheiden. Dadurch ist denkbar, dass die Modelle, nach einer bestimmten Anzahl an Trainingsschritten, unterschiedlich gut Clickbait spoilern können, weil sich das Modell mit steigender Anzahl an Trainingsschritten mehr und mehr dem SQuADv1.1-Datensatz anpasst. Dafür speichern wir beim Feintuning der Modelle auf SQuADv1.1 alle 1,000 Schritte das Modell und evaluieren dieses auf Passage-1kTrain-Valid. In Tabelle 32 erkennen wir die besten Ergebnisse auf Passage-1kTrain-Valid mit dem jeweiligen Trainingsschritt, nach welchem ein Modell das beste Ergebnis erzielt. Die Behauptung des Overfittings scheint ohne Ausnahme zu stimmen, da alle Modelle spätestens nach 15,000 Trainingsschritten, aber teilweise auch schon nach 7,000 (FunnelTransformer), am besten Spoiler der Klasse Passage spoilern. Wahrscheinlich lernen die Modelle in den ersten Schritten allgemeingültigere Regeln zu Fragen, die sich auch auf Clickbait übertragen lassen. Erst in den letzten Schritten passen sich die von den Modellen erlernten Regeln dem SQuADv1.1-Datensatz genauer an. Dennoch sind die Ergebnisse um mindestens sechs korrekte Spoiler schlechter als die von den auf Passage-1kTrain-Train feingetunten Modelle.

Tabelle 32: Effektivität der auf SQuADv1.1 trainierten Modelle beim Spoilern der Spoilerklasse Passage mit der Anzahl an Trainingsschritte, die zum besten Ergebnis führen. Gemessen auf Passage-1kTrain-Valid (102 Einträge) und angegeben in Prozent. In Klammern erkenen wir die Anzahl an Spoilern über dem im Tabellenkopf auch in Klammern angegebenen Schwellwert der Evaluationsmetrik.

Modell	Schritte	BLEU-4 (>30)	METEOR-F (>70)	BERTScore-F1 (>60)
ALBERT	14000	11.85 (15)	27.86 (15)	34.59 (18)
$\text{BERT}_{\text{uncased}}$	13000	8.13 (11)	19.87 (10)	31.88 (12)
$\text{BERT}_{\text{cased}}$	15000	10.40 (13)	22.21 (10)	33.30 (13)
ELECTRA	9000	13.64 (16)	29.48 (15)	35.35 (18)
FunnelTrans	7000	15.73 (22)	**32.14 (20)**	**39.53 (20)**
MPNet	11000	**16.44 (19)**	31.06 (19)	37.94 (18)
RoBERTa	15000	15.47 (19)	28.78 (20)	38.63 (19)

Wie schon in Abschnitt 2.2 erwähnt, ist der Mangel an Daten eine mögliche Ursache für schlechtere Effektivität von feingetunten Modellen. Da wir davon ausgehen, dass Question Answering und Clickbait-Spoiling in gewissen Zügen verwandt sind, nutzen wir die Datenmenge von SQuADv1.1, um unseren Mangel an Daten zu kompensieren. Dazu wählen wir die auf SQuADv1.1 feingetunten Modelle nach jeweils dem Trainingsschritt, bei dem diese das beste Ergebnis auf Passage-1kTrain-Valid erzielen (Tabelle 32). Diese feintunen wir mit dem `run_qa.py` Skript genauso wie auf SQuADv1.1 weiter auf Passage-1kTrain-Train. Die Werte für Hyperparameter nehmen wir analog zum Feintuning auf SQuADv1.1. Mit dem Feintuningansatz, zuerst auf SQuADv1.1 und dann fortführend auf Passage-1kTrain-Train feinzutunen, erzielen wir die besten Ergebnisse zum Spoilern der Spoilerklasse Passage (Tabelle 33) MPNet spoilert damit am besten Clickbait der Klasse Passage, feingetunt auf SQuADv1.1, mit fortführendem Feintuning auf Passage-1kTrain-Train und schafft auf Passage-1kTrain-Valid rund neun mehr korrekte Spoiler als beim Feintuning nur auf Passage-1kTrain-Train. Insgesamt haben wir das Spoiling von Clickbait der Klasse Passage im Vergleich zur ALBERT-Baseline um rund 20 korrekte Spoiler verbessert und sind damit in der Lage, mindestens 34 von 102 (34 %) Clickbait-Nachrichten der Klasse Passage korrekt zu spoilern.

Weiterhin wollen wir den Effekt der Modellgröße zeigen. Die bisherigen Ergebnisse basieren auf Training mit zwei 8 GB VRAM Grafikkarten und den kleinsten Versionen der aufgeführten Modelle, weil größere Modelle mit dem `run_qa.py` Skript von Huggingface 8 GB VRAM überschreiten. Dazu feintunen wir die drei weiteren Modelle: $\textit{RoBERTa}_{\textit{large}}$ (Liu et al., 2019), $\textit{DeBERTa}_{\textit{large}}$

Tabelle 33: Effektivität der zuerst auf SQuADv1.1 feingetunten und danach weiter auf Passage-1kTrain-Train (398 Einträge) feingetunten Modelle beim Spoilern der Spoilerklasse Passage mit der Anzahl an Trainingsschritten, die zum besten Ergebnis führen. Gemessen auf Passage-1kTrain-Valid (102 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem im Tabellenkopf auch in Klammern angegebenen Schwellwert der Evaluationsmetrik.

Modell	Schritte	BLEU-4 (>30)	METEOR-F (>70)	BERTScore-F1 (>60)
ALBERT	400	24.51 (33)	38.42 (27)	44.61 (24)
$\text{BERT}_{\text{uncased}}$	500	18.05 (22)	32.50 (20)	39.86 (18)
$\text{BERT}_{\text{cased}}$	700	17.65 (22)	28.09 (20)	40.30 (16)
ELECTRA	100	25.78 (32)	39.87 (30)	46.64 (27)
FunnelTrans	300	28.59 (36)	40.95 (32)	47.93 (29)
MPNet	100	**30.16 (36)**	40.68 (35)	**50.07 (32)**
RoBERTa	100	27.61 (35)	**41.55 (35)**	48.76 (30)

Tabelle 34: Effektivität der drei ausgewählten Sprachverständnismodelle in der ‚large'-Version auf SQuADv1.1 angegeben in Prozent von SQuAD-F1. Jedes Modell ist trainiert für 3 Epochen (∼33,000 Trainingsschritte) mit Batchgröße 8, Sequenzlänge 384 und Lernrate 3e-5.

BigBird	$\text{DeBERTa}_{\text{large}}$	$\text{RoBERTa}_{\text{large}}$
93.2	94.0	93.8

(He et al., 2020) und *BigBird* (Zaheer et al., 2021) zuerst auf SQuADv1.1 und danach fortführend auf Passage-1kTrain-Train. Dabei benötigen die Modelle bis zu 40 GB VRAM. Wir wählen für Hyperparameter Batchgröße 8, aufgrund des Speicherlimits von 40 GB, wodurch sich die Anzahl der Trainingsschritte auf rund 33,000 verdoppelt. Für den Rest der Hyperparameter verwenden wir weiterhin die gleichen Werte wie in allen bisherigen Feintuning-Ansätzen. In Tabelle 34 sehen wir die Effektivität der drei Modelle auf SQuADv1.1. Wir erkennen, dass diese Modelle nochmal näher an den State-Of-The-Art-Ergebnissen auf SQuADv1.1 (95.4 % SQuAD-F1) liegen, als die kleineren sieben vorher vorgestellten Modelle. Wir wählen die auf SQuADv1.1 feingetunten Modelle jeweils nach dem Trainingsschritt, nach welchem diese das beste Ergebnis auf Passage-1kTrain-Valid erzielen, aus und feintunen diese weiter auf Passage-1kTrain-Train Damit untersuchen wir, ob sich wie bei SQuADv1.1 ein positiver Effekt auf die Effektivität durch das Nutzen größerer Modelle beobachten lässt. Tabelle 35 zeigt uns einen leichten positiven Effekt. Die Ergebnisse

Tabelle 35: Effektivität der auf zuerst auf SQuADv1.1 feingetunten und danach weiter auf Passage-1kTrain-Train (398 Einträge) feingetunten Modelle auf Passage-1kTrain-Valid (102 Einträge) mit Anzahl an Trainingsschritten, die zum besten Ergebnis führen. Gemessen auf Passage-1kTrain-Valid (102 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem im Tabellenkopf auch in Klammern angegebenen Schwellwert der Evaluationsmetrik.

Modell	Schritte	BLEU-4 (>30)	METEOR-F (>70)	BERTScore-F1 (>60)
BigBird_{384}	600	23.89 (30)	36.2 (28)	44.55 (27)
$\text{DeBERTa}_{\text{large}}$	100	**29.52 (38)**	**43.72 (36)**	**49.63 (37)**
$\text{RoBERTa}_{\text{large}}$	600	**29.58 (35)**	43.49 (32)	48.65 (32)

von $\text{RoBERTa}_{\text{large}}$ und $\text{DeBERTa}_{\text{large}}$ sind bezüglich METEOR-F besser als die Ergebnisse von MPNet in Tabelle 33, während BLEU-4 und BERTScore-F1 annähernd gleich hoch sind. Das heißt, die zwei Modelle tendieren ähnliche Spoiler zu extrahieren, aber mit anders gebeugten Wörtern oder Synonymen. BigBird fällt weit zurück auf eine ähnliche Effektivität wie ALBERT. Die Mindestanzahl korrekter Spoiler steigt mit $\text{DeBERTa}_{\text{large}}$ damit auf 37 von 102 (37 %), wenn eine 40 GB VRAM Grafikkarte zur Verfügung steht.

Das BigBird-Modell unterstützt im Gegensatz zu den anderen in diesem Kapitel aufgeführten Modellen Sequenzlängen über 512 Wörter. Diese Möglichkeit nutzen wir, um den Einfluss der Sequenzlänge auf die Effektivität des Spoilerns von Clickbait der Klasse Passage zu betrachten. Die Sequenzlänge gibt uns an, wie viele Wörter das Modell verarbeiten kann. Bei Textabschnitten länger als die angegebene Sequenzlänge alle restlichen Wörter abgeschnitten und damit auch nicht einbezogen. Eine höhere Sequenzlänge beim Feintuningprozess eines Modells bedeutet, mehr Wörter zum Ziehen von Schlüßen. Bei SQuADv1.1 mag eine Sequenzlänge höher als 384 keine Verbesserungen mehr bringen, da der Text, aus dem die Antwort extrahiert werden soll, nur einem Paragraphen entspricht, der durchschnittlich 120 Wörter besitzt. Bei Passage-1kTrain-Train hingegen haben die Texte mehrere Paragraphen mit durchschnittlich 689 Wörtern (Tabelle 16). Dadurch vermuten wir, dass höhere Sequenzlängen durchaus Verbesserungen mit sich bringen können. Dafür nehmen wir das auf SQuADv1.1 feingetunte BigBird-Modell nach dem Trainingsschritt, nach welchem dieses das beste Ergebnis auf Passage-1kTrain-Valid erzielt, und feintunen es weiter auf Passage-1kTrain-Train mit Sequenzlängen von 384 bis 1280. In Tabelle 36 erkennen wir keine eindeutige Abhängigkeit der Effektivität von BigBird-Modellen mit der Sequenzlänge. Bei einer Sequenzlänge von 512 findet

Tabelle 36: Effektivität des auf zuerst auf SQuADv1.1 feingetunten BigBird-Modells mit Sequenzlänge 384 und danach weiter auf Passage-1kTrain-Train (398 Einträge) feingetunten Modelle mit unterschiedlichen Sequenzlängen (Fußnote) mit Anzahl an Trainingsschritten, die zum besten Ergebnis führen. Gemessen auf Passage-1kTrain-Valid (102 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem im Tabellenkopf auch in Klammern angegebenen Schwellwert der Evaluationsmetrik.

Modell	Schritte	BLEU-4 (>30)	METEOR-F (>70)	BERTScore-F1 (>60)
BigBird_{384}	600	23.89 (30)	36.2 (28)	44.55 (27)
BigBird_{512}	400	**27.09 (34)**	39.07 (29)	45.64 (31)
BigBird_{768}	600	23.13 (32)	38.24 (27)	40.84 (26)
BigBird_{1024}	400	24.41 (33)	**39.96 (29)**	42.41 (28)
BigBird_{1280}	100	25.86 (32)	39.31 (33)	**46.39 (34)**

ein deutlicher Sprung der Effektivität statt. Bei einer Sequenzlänge von 768 sinkt die Effektivität wieder rapide um rund 4 % BLEU-4 und BERTScore-F1. Ab da ist mit der steigenden Sequenzlänge auch eine kontinuierliche Erhöhung des BLEU-4-Scores und des BERTScore-F1-Scores zu verzeichnen. Entweder gibt es einen Zusammenhang mit der Sequenzlänge oder bei 512 wird ein lokales Maximum der Scores erreicht. Beim METEOR-F-Score schwanken die Ergebnisse unregeläßig. Damit gibt es keine eindeutigen Zusammenhang zwischen der Sequenzlänge und der Effektivität der Modelle beim Spoilern der Spoilerklasse Passage. Da diese Ergebnisse nicht besser sind als die schon beobachteten, belassen wir dieses Experiment als Indiz oder Anregung für weitere Forschung.

Tabelle 37: Effektivität der auf SQuADv1.1 trainierten Modelle beim Spoilern der Spoilerklasse Phrase. Gemessen auf Phrase-1kTrain-Valid (80 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem im Tabellenkopf auch in Klammern angegebenen Schwellwert der Evaluationsmetrik.

Modell	BLEU-4 (>50)	METEOR-F (>70)	BERTScore-F1 (>80)
AllenAI-QA	35.30 (24)	**38.00 (25)**	57.40 (22)
ALBERT	13.21 (10)	9.41 (10)	37.16 (9)
$\text{BERT}_{\text{uncased}}$	24.81 (21)	13.04 (18)	46.01 (19)
$\text{BERT}_{\text{cased}}$	28.46 (23)	16.99 (21)	49.86 (21)
ELECTRA	30.64 (25)	17.65 (22)	48.32 (21)
FunnelTrans	26.29 (21)	14.29 (22)	45.29 (18)
MPNet	42.42 (35)	22.85 (31)	57.04 (30)
RoBERTa	**45.2 (36)**	25.10 (34)	**59.31 (31)**

5.3.3 Spoiling von Clickbait der Klasse Phrase

Für die Verbesserung des Spoilings der Spoilerklasse Phrase werten wir die gleichen Feintuning-Ansätze aus, wie zuvor bei der Spoilerklasse Passage (Unterabschnitt 5.3.2). Wir erwarten ein ähnliches Verhalten der verschiedenen Modelle, wie bei Spoilern der Klasse Passage zu beobachten. In Tabelle 37 sehen wir zuerst, wie gut die auf SQuADv1.1 feingetunten Modelle auf den 80 Clickbait-Posts mit Spoilern der Klasse Phrase von 1kTrain-Valid (Phrase-1kTrain-Valid) im Vergleich zum AllenAI-Document-QA-Ansatz, feingetunt auf TriviaQA, abschneiden. Auf den 80 Einträgen von Clickbait der Klasse Phrase in Phrase-1kTrain-Valid sehen wir, dass das AllenAI-Document-QA-Ansatz, trotz seines Alters, sehr gut spoilert. Der deutlich höhere METEOR-F-Score weist für AllenAI-Document-QA-Ansatz ein hohes Potenzial zum Spoilern der Spoilerklasse Phrase auf, obwohl die Mindestanzahl korrekter Spoiler um neun Spoiler geringer ausfällt als bei RoBERTa. Dadurch, dass RoBERTa bei BLEU-4 aber viel besser und bei BERTScore ein wenig besser ist, werten wir das als höhere Effektivität. Analog zum Unterabschnitt 5.3.2 zu Spoilern der Klasse Passage, feintunen wir jetzt die Modelle auf den 320 Clickbait-Posts mit Spoilern der Klasse Phrase des 1kTrain-Train-Datensatz (Phrase-1kTrain-Train). Wir nutzen die gleichen Werte für die Hyperparameter, nur entsprechen diesmal 15 Epochen rund 800 Schritten. Wir erkennen wieder, dass jedes Modell feingetunt auf den Clickbait-Posts selbst, generell bessere Ergebnisse als die auf SQuADv1.1 feingetunten Modelle (siehe Tabelle 38) erzielen. Die Spoilerklasse Phrase spoilert MPNet feingetunt auf Phrase-1kTrain-Train am besten und RoBERTa am zweitbesten. Als nächstes untersuchen wir,

Tabelle 38: Effektivität der auf Phrase-1kTrain-Train (320 Einträge) trainierten Modelle beim Spoilern der Spoilerklasse Phrase mit Anzahl an Trainingsschritten, die zum besten Ergebnis führen. Gemessen auf Phrase-1kTrain-Valid (80 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem im Tabellenkopf auch in Klammern angegebenen Schwellwert der Evaluationsmetrik.

Modell	Schritte	BLEU-4 (>50)	METEOR-F (>70)	BERTScore-F1 (>80)
ALBERT	600	56.36 (45)	**54.25 (43)**	69.83 (42)
$\text{BERT}_{\text{uncased}}$	300	41.85 (32)	35.39 (33)	60.42 (31)
$\text{BERT}_{\text{cased}}$	500	45.40 (36)	35.81 (36)	60.76 (31)
ELECTRA	500	44.72 (34)	37.45 (35)	61.13 (33)
FunnelTrans	200	51.35 (42)	50.46 (38)	64.49 (33)
MPNet	400	**60.03 (48)**	51.15 (48)	**71.73 (44)**
RoBERTa	500	58.82 (47)	46.60 (46)	69.91 (42)

nach wie vielen Trainingsschritten die auf SQuADv1.1 feingetunten Modelle am besten auf Phrase-1kTrain-Valid Clickbait spoilern. Dazu betrachten wir, wie zuvor bei Spoilern der Klasse Passage, die Effektivität der Modelle nach jedem 1000. Trainingsschritt. Die besten Ergebnisse mit der entsprechenden Anzahl an notwendigen Trainingsschritten, listen wir in Tabelle 39 auf. Mit Ausnahme von $\text{BERT}_{\text{uncased}}$ erreichen alle Modelle spätestens nach 16,000 Trainingsschritten, aber teilweise auch schon nach 6,000 ($\text{BERT}_{\text{cased}}$) die besten Ergebnisse auf Phrase-1kTrain-Valid. Die Ergebnisse sind trotzdem, ähnlich wie bei der Spoilerklasse Passage beobachtet, um rund sieben korrekte Spoiler schlechter als die von den auf Phrase-1kTrain-Train feingetunten Modellen. Als letzten Feintuning-Ansatz feintunen wir die Modelle aus Tabelle 39 weiter auf Phrase-1kTrain-Train, um wie bei Spoilern der Klasse Passage die soweit beste Effektivität zu erreichen. Die Hyperparameter bleiben wieder unverändert. Tabelle 40 illustriert die damit erzielten Ergebnisse. Wieder erkennen wir, dass wir mit diesem letzten Feintuning-Ansatz die besten Modelle zum Spoilern von Clickbait erzeugen. Das MPNet-Modell spoilert die Spoilerklasse Phrase am besten, aber sehr dicht gefolgt von RoBERTa mit nahezu gleichen Ergebnissen. Anhand der beiden Modellen $\text{BERT}_{\text{uncased}}$ und $\text{BERT}_{\text{cased}}$, können wir eine interessante Beobachtung feststellen. $\text{BERT}_{\text{uncased}}$ verarbeitet einen Text im Gegensatz zu $\text{BERT}_{\text{cased}}$ nur in Kleinbuchstaben. Während $\text{BERT}_{\text{uncased}}$ in allen Feintuningansätzen schlechter abschneidet als $\text{BERT}_{\text{cased}}$, ist $\text{BERT}_{\text{uncased}}$ im letzten Feintuningansatz mit fortführendem Feintuning mit $\text{BERT}_{\text{cased}}$ gleichauf. Wir hätten vermuten können, dass die Verarbeitung nur in Kleinbuchstaben nachteilig für das Feintuning ist, aber das ist möglicher-

Tabelle 39: Effektivität der auf SQuADv1.1 trainierten Modelle beim Spoilern der Spoilerklasse Phrase mit Anzahl an Trainingsschritten, die zum besten Ergebnis führen. Gemessen auf Phrase-1kTrain-Valid (80 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem im Tabellenkopf auch in Klammern angegebenen Schwellwert der Evaluationsmetrik.

Modell	Schritte	BLEU-4 (>50)	METEOR-F1 (>70)	BERTScore-F1 (>80)
ALBERT	8000	18.48 (14)	8.71 (14)	39.70 (13)
$\text{BERT}_{\text{uncased}}$	16000	24.81 (21)	12.77 (18)	45.93 (19)
$\text{BERT}_{\text{cased}}$	6000	34.89 (29)	23.92 (26)	53.86 (25)
ELECTRA	16000	34.39 (28)	19.30 (25)	49.78 (23)
FunnelTrans	10000	28.77 (24)	16.39 (22)	47.70 (20)
MPNet	9000	50.64 (40)	31.85 (38)	**65.06 (38)**
RoBERTa	12000	**52.06 (42)**	**32.62 (40)**	64.61 (34)

weise nicht der Fall, oder es bestehen Zusammenhänge mit dem fortführenden Feintuning. Im Endeffekt schaffen wir mit dem MPNet-Modell feingetunt auf SQuADv1.1 mit fortführendem Feintuning auf Phrase-1kTrain-Train auf Phrase-1kTrain-Valid rund 31 mehr korrekte Spoiler als mit unserer Baseline vom AllenAI-Document-QA-Verfahren. Damit können wir mindestens 56 von 80 (70 %) Clickbait-Nachrichten der Klasse Phrase korrekt spoilern.

Zum Effekt größerer Modelle beim Spoilern von Clickbait der Klasse Phrase betrachten wir wie auch zu Clickbait der Klasse Passage die drei Modelle $\text{RoBERTa}_{\text{large}}$, $\text{DeBERTa}_{\text{large}}$ und BigBird. Analog zum Vorgehen in Unterabschnitt 5.3.2 feintunen wir die drei Modelle erst auf SQuADv1.1 und führen anschließend ein fortführendes Feintuning auf Phrase-1kTrain-Train durch. Anhand der Ergebnisse in Tabelle 41 können wir eine weitere Verbesserung in der Effektivität beobachten (zum Vergleich siehe Tabelle 40). Das $\text{RoBERTa}_{\text{large}}$-Modell schafft hier die besten Ergebnisse mit rund fünf korrekten Spoilern mehr als MPNet zuvor. Das heißt, wir beobachten einen positiven Effekt auf die Effektivität beim Spoilern von Clickbait der Klasse Phrase durch das Nutzen größerer Modelle. Wenn wir also eine Grafikkarte mit 40 GB VRAM zur Verfügung haben, können wir noch mehr Clickbait-Nachrichten korrekt spoilern: 61 von 80 (76 %).

Tabelle 40: Effektivität der zuerst auf SQuADv1.1 feingetunten und danach weiter auf Phrase-1kTrain-Train (320 Einträge) feingetunten Modelle beim Spoilern der Spoilerklasse Phrase mit Anzahl an Trainingsschritten, die zum besten Ergebnis führen. Gemessen auf Phrase-1kTrain-Valid (80 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem im Tabellenkopf auch in Klammern angegebenen Schwellwert der Evaluationsmetrik.

Modell	Schritte	BLEU-4 (>50)	METEOR-F (>70)	BERTScore-F1 (>80)
ALBERT	300	63.82 (50)	55.97 (49)	74.07 (46)
$\text{BERT}_{\text{uncased}}$	100	62.36 (49)	53.17 (47)	75.87 (47)
$\text{BERT}_{\text{cased}}$	300	60.27 (49)	58.87 (47)	73.55 (44)
ELECTRA	200	69.10 (55)	**65.97 (53)**	79.26 (51)
FunnelTrans	600	68.31 (54)	63.89 (53)	78.78 (51)
MPNet	700	**72.92 (58)**	**65.90 (57)**	**80.26 (55)**
RoBERTa	200	**73.02 (59)**	65.56 (57)	**80.39 (54)**

Tabelle 41: Effektivität der zuerst auf SQuADv1.1 feingetunten und danach weiter auf Phrase-1kTrain-Train (320 Einträge) feingetunten Modelle beim Spoilern der Spoilerklasse Phrase mit Anzahl an Trainingsschritten, die zum besten Ergebnis führen. Gemessen auf Phrase-1kTrain-Valid (80 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem im Tabellenkopf auch in Klammern angegebenen Schwellwert der Evaluationsmetrik.

Modell	Schritte	BLEU-4 (>50)	METEOR-F (>70)	BERTScore-F1 (>80)
BigBird_{384}	300	69.21 (55)	64.80 (54)	77.39 (49)
$\text{DeBERTa}_{\text{large}}$	300	70.19 (57)	65.08 (56)	78.02 (50)
$\text{RoBERTa}_{\text{large}}$	200	**79.47 (66)**	**78.61 (61)**	**84.04 (58)**

5.3.4 Kombinieren der Ansätze: Klassifikator und Spoiling der beiden Klassen

Um die Behauptung zu stützen oder zu falsifizieren, dass die Modelle, feingetunt auf Clickbait der Klasse Phrase und Passage getrennt, besser spoilern können, als Modelle feingetunt auf beiden Klassen von Clickbait, wenden wir den letzten Feintuning-Ansatz an. Hier betrachten wir nur die ‚base'-Modelle von MPNet und RoBERTa, weil diese bei beiden Klassen von Clickbait die besten Ergebnisse erzielen. Wir nehmen also MPNet feingetunt für 9,000 Schritte und RoBERTa feingetunt für 12,000 Schritte auf SQuADv1.1 und feintunen diese für acht Epochen bzw. rund 1,000 Schritte auf 1kTrain-Valid. Dazu nutzen wir Batchgröße: 16, Sequenzlänge: 384, Lernrate: 3e-5, Rest: Standardwert des `run_qa.py` Skripts und zwei Grafikkarten mit 8 GB VRAM. MPNet zeigt nach dem Feintuning bessere Ergebnisse und gilt für uns damit als Maßstab, der überboten werden muss, um mit getrenntem Feintuning besser zu spoilern (Tabelle 42). Vergleichen wir die Summe der mindestens korrekt gespoilerten Clickbait-Nachrichten aus dem getrennten Feintuning ($\text{MPNet}_{\text{combi}}$) zuvor in Tabelle 33 und Tabelle 40, sehen wir klar, dass das getrennte Feintuning jeweils zu besser spoilernden Modellen führt (siehe Tabelle 43). Um die besseren Modelle nutzen zu können, müssen wir für die Clickbait-Nachrichten wissen, um welche Klasse von Clickbait es sich handelt, bevor wir sie von den Modellen spoilern lassen. Ansonsten würde das Modell feingetunt auf Clickbait der Klasse Phrase potenziell Clickbait der Klasse Passage und umgekehrt spoilern, was zu deutlich schlechteren Ergebnissen führen dürfte, als das auf beiden Clickbaitklassen feingetunte MPNet-Modell ($\text{MPNet}_{\text{combi}}$). Dafür haben wir in Kapitel 4 einen Klassifikator gebaut, der mit einer Accuracy von 74.9 % zwischen Clickbait der Klasse Phrase und Passage unterscheiden kann. Aus der Tabelle 22 können wir anhand des Recalls abschätzen, wie viele Spoiler wir mit unserem Klassifikator bei maximaler Zuverlässigkeit spoilern können. Da

Tabelle 42: Effektivität der zuerst auf SQuADv1.1 feingetunten und danach weiter auf 1kTrain-Train (718 Einträge) feingetunten Modelle auf 1kTrain-Valid (182 Einträge) mit Anzahl an Trainingsschritten, die zum besten Ergebnis führen. In Klammern hinter dem Score steht die Anzahl an Spoilern mit einem Score über dem Schwellwert für korrekte Spoiler für Clickbait jeweils der Klasse Phrase und Passage aufsummiert.

Modell	Schritte	BLEU-4	METEOR-F	BERTScore-F1
RoBERTa	300	39.24 (79)	36.58 (75)	54.24 (67)
MPNet	400	**41.22 (82)**	**39.44 (79)**	**56.2 (76)**

Tabelle 43: Vergleich zwischen getrennt feingetunten Modellen gepaart mit einem Klassifikator zur Unterscheidung von Clickbait der Klasse Phrase und Passage ($\text{MPNet}_{\text{combi}}$) und einem auf beiden Clickbaitklassen feingetunten Modell ($\text{MPNet}_{\text{single}}$). Bei $\text{MPNet}_{\text{combi}}$ unterscheiden wir zwischen Paarungen mit unterschiedlich effektiven Klassifikatoren zur Unterscheidung von Clickbait der Klasse Phrase und Passage (75%-Klass, 87%-Klass, 100%-Klass). Dabei stellt 100%-Klass ein Oracel, das perfekt zwischen den Clickbaitklassen unterscheiden kann, dar und 75%-Klass unseren Klassifikator.

	BLEU-4	METEOR-F	BERTScore-F1
75%-Klass+$\text{MPNet}_{\text{combi}}$	71	69	65
$\text{MPNet}_{\text{single}}$	82	79	76
87%-Klass+$\text{MPNet}_{\text{combi}}$	82	80	75
100%-Klass+$\text{MPNet}_{\text{combi}}$	94	92	86

wir 23 % der Clickbait-Nachrichten mit Spoilern der Klasse Phrase zuverlässig Spoilern können, heißt es, dass wir rund 13.2 % aller Spoiler der Klasse Phrase erfolgreich extrahieren. Für Spoiler der Klasse Passage sind es mit einem Recall von 21 %, 7 % aller Clickbait-Nachrichten. Damit schafft unser Lösungsansatz insgesamt schätzungsweise 20 % aller Clickbait-Nachrichten erfolgreich zu spoilern, wenn zwei 8 GB VRAM Grafikkarten zum Feintuning verwendet werden. Wenn eine Grafikkarte mit 40 GB VRAM zur Verfügung steht, steigt die Anzahl auf 22 %. Die Summe der mindestens korrekt gespoilerten Clickbait-Nachrichten aus den getrennt feingetunten Modellen würde einer Art ‚Oracel' entsprechen, welches einen Klassifikator mit 100 % Accuracy darstellt. Unter der Annahme, dass ein falsch klassifizierter Clickbait auch automatisch zu einem falsch gespoilerten Clickbait führt, vollziehen wir folgendes Gedankenexperiment: Wenn wir die Summe der mindestens korrekt gespoilerten Clickbait-Nachrichten aus den getrennt feingetunten Modellen mit der Accuracy multiplizieren, schätzen wir ab, wie gut unser Clickbait-Spoiling-System aus Klassifikator und den Modellen für jeweils Clickbait der Klasse Phrase und Passage spoilert. Das dabei errechnete Resultat mindestens korrekter Spoiler zeigt, dass die Effektivität unseres Klassifikators und den besser spoilenden Modellen zusammen nicht ausreicht (68 von 182 bzw. 37 %), um besser als $\text{MPNet}_{\text{single}}$ zu spoilern. Der Klassifikator braucht mehr als 87 % Accuracy beim Vorhersagen der Spoilerklassen Phrase und Passage, damit unser Lösungsansatz besser als $\text{MPNet}_{\text{single}}$ spoilert (Tabelle 43). Damit können wir, wenn wir zwei Grafikkarten mit 8 GB VRAM zur Verfügung haben, mit $\text{MPNet}_{\text{single}}$ mehr Clickbait korrekt spoilern als mit unserem Clickbait-Spoiling-System und zwar mindestens 79 von 182 (43 %) Clickbait-Nachrichten.

Kapitel 6

Fazit und Ausblick

In dieser Arbeit setzten wir uns mit dem Thema Clickbait und dem automatisierten Schließen dessen Curiosity Gaps (also dem Spoilern von Clickbait) auseinander. Bisher wurden nur Ansätze veröffentlicht, die das Anzeigen von Clickbait-Nachrichten durch Clickbait-Erkennung eindämmen sollen. Angewandt wird ein solcher Ansatz im Algorithmus des Facebook-Newsfeeds[1]. Unser Lösungansatz hingegen setzt beim konkreten Spoilern von Clickbait an (Abbildung 26).

Die Präsenz von Clickbait ist in den letzten zwei Jahren deutlich gestiegen. So sahen wir anfangs Clickbait überwiegend auf dubiosen Webseiten, die einen Leser zum Klicken animieren sollten. Heute ist Clickbait bis in die Nachrichtenportale vorgedrungen und gefährdet den qualitativen Journalismus (Dvorkin, 2016). Deshalb ist es vonnöten, den manipulativen Charakter von Clickbait abzuschwächen.

Da es keine öffentlichen Datensätze für Trainingszwecke gab, in denen Clickbait mit deren expliziten Spoilern zu finden sind, konstruierten wir dazu einen eigenen Datensatz namens Webis-Clickbait-Spoiling-20. Der Datensatz besteht aus 5,000 Clickbait-Posts von Social-Media-Accounts auf Twitter, Reddit und Facebook. Diese annotierten wir mit Spoilern, die den Curiosity Gap schließen. Hinzu kamen auch Annotationen von Spoilerklassen zur Unterscheidung zwischen kurzen, langen und mehrteiligen Spoilern (Phrase, Passage und Multi). Um die Qualität des Datensatzes zu erhöhen, bereinigten wir die vom Clickbait-Post verlinkten Texte, indem wir Werbetexte und ablenkende Referenzen auf andere Texte und Social-Media entfernten. Zur Erweiterung des

[1] https://about.fb.com/news/2016/08/news-feed-fyi-further-reducing-clickbait-in-feed/

Datensatzes kann das Archiv der Webseite Upworthy[2], das für seine Massenproduktion an Clickbait-Titeln für alltägliche Geschichten bekannt ist, genutzt werden. Sollten diese Texte nicht ausreichen, können Ansätze zur automatisierten Clickbait-Generierung (Xu et al., 2019) in Erwägung gezogen werden.

Als Lösungsansatz zum Spoilern von Clickbait überlegten wir uns, die Spoilerklassen Phrase und Passage getrennt voneinander zu betrachten. Dazu benötigten wir einen Spoiler-Typ-Klassifikator, der uns die Spoilerklasse anhand der Clickbait-Nachricht vorhersagt. In unseren Experimenten zu Häufigkeiten von Wort- und POS-N-Grammen als Features erkannten wir, dass eine Kombination aus Wort-1-, Wort-2-, POS-2- und POS-3-Grammen die besten Vorhersagen treffen konnten. Als bester Algorithmus kristallisierte sich der Logistic-Regression-Klassifikator heraus, der mit 74.9 % Accuracy die Klassen Phrase und Passage vorhersagt. Der Mensch hingegen kommt auf eine Accuracy von 85 %. In Zukunft könnte es sich lohnen, mit kontextualisierten Worteinbettungen und *Coreference Resolution* als Features zu experimentieren, sowie durch Sprachverständnismodelle zu klassifizieren. Damit der verlinkte Text besser in die Klassifikation einfließen kann, sollten *Topic Models* und andere zusammenfassende Ansätze für Texte beurteilt werden. Sollten die genannten Ansätze nicht zum erwünschten Erfolg führen, können die Spoilerklassen, angelehnt an die Klassen von Fragen aus dem TREC-10-Datensatz, überdacht werden.

Aufgrund der Trennung der Spoiler in die Klassen Phrase und Passage war es möglich, zwei Sprachverständnismodelle zu nutzen, wovon wir jeweils ein Sprachverständnismodell zum Spoilern einer Klasse feintunten. Mit der Erkenntnis, dass Clickbait-Nachrichten sich in Fragen umformulieren lassen, verwendeten wir dabei Methoden für Question Answering. Da die Evaluation der Spoiler eine wichtige Rolle spielt, um zu entscheiden, welche Ansätze besser oder schlechter funktionieren, wählten wir die drei Metriken BLEU-4, METEOR-F_{mean} und BERTScore-F1 zum Vergleichen der extrahierten Spoiler mit den annotierten Spoilern aus. Mit den drei Metriken konnten wir gut syntaktische und semantische Ähnlichkeiten von Spoilern einschätzen. METEOR-F_{mean} und BERTScore-F1 bieten dabei noch Anpassungspotenzial. Dafür können die Parameter von METEOR-F_{mean} experimentell angepasst und jene von BERTScore-F1 auf den Spoilern feingetunt werden, um so eine höhere Korrelation mit unserer manuellen Auswertung zu erzielen. Dadurch lassen sich bessere Ansätze noch deutlicher von schlechteren Ansätzen unterscheiden. Zum BERTScore-F1 existiert auch eine Variation namens *Conditional BERTScore-F1* (Chen et al., 2019), welche die Verbindung des Spoilers zur Clickbait-Nachricht in der Evaluation einbezieht. Zudem kann auch das Annotieren wei-

[2]https://www.upworthy.com/

terer möglicher Spoiler für unseren Datensatz Vorteile bei der Auswertung von extrahierten Spoilern bringen und zu einem besseren Verständnis der Optimierungsmöglichkeiten und Probleme beim Clickbait-Spoiling führen.

Unser Trainingsansatz des fortgeführten Feintunings, um Datenmangel zu kompensieren, erzielte die besten Resultate, indem wir die Sprachverständnismodelle zuerst auf SQuADv1.1 bis zu einem Maximum an Effektivität feintunten und danach das Tuning auf unserem Trainingsdatensatz 1kTrain fortführten. Am besten spoilerte das Modell $\text{RoBERTa}_{\text{large}}$ mit 78.6 % METEOR-F_{mean} Spoiler der Klasse Phrase und $\text{DeBERTa}_{\text{large}}$ mit 43.7 % METEOR-F_{mean} Spoiler der Klasse Passage. Der enorme Unterschied zwischen den Effektivitäten der Modelle der beiden Spoilerklassen deutet darauf hin, dass Modelle die Bedeutung von kurzen beispielsweise Nominalphrasen besser verstehen lernen als die Bedeutung längerer Sinnesabschnitte. Für die Zukunft können unsere jetzigen Modelle potenziell durch Feintuning mit höheren Sequenzlängen oder Modellgrößen noch ein weiteres Stück verbessert werden. Es gilt, weitere Trainingsmethoden und Datensätze als Feintuningbasis wie in *CorefQA* (Wu et al., 2020), *SMART-Task*[3], *Span Selection Pretraining* (Glass et al., 2020) oder *Soft Prompts* (Qin und Eisner, 2021), zu erforschen.

In dieser Arbeit haben wir die Spoilerklasse Multi nicht weiter betrachtet. Zukünftig sollten Lösungsansätze auch für diese Klasse entwickelt werden, um Clickbait-Spoiling voranzubringen. Eine Möglichkeit dafür wäre ein *Ranking-Ansatz*, der mehrere Spoiler nach Relevanz geordnet extrahiert. Ein anderer Ansatz für mehrteilige Spoiler wäre es, einen Teil der Spoilers zu extrahieren, daraufhin aus dem verlinkten Text zu entfernen und eine oder mehrere erneute Extraktionen durchzuführen, bis der Spoiler vollständig ist.

[3] https://smart-task.github.io/2021/

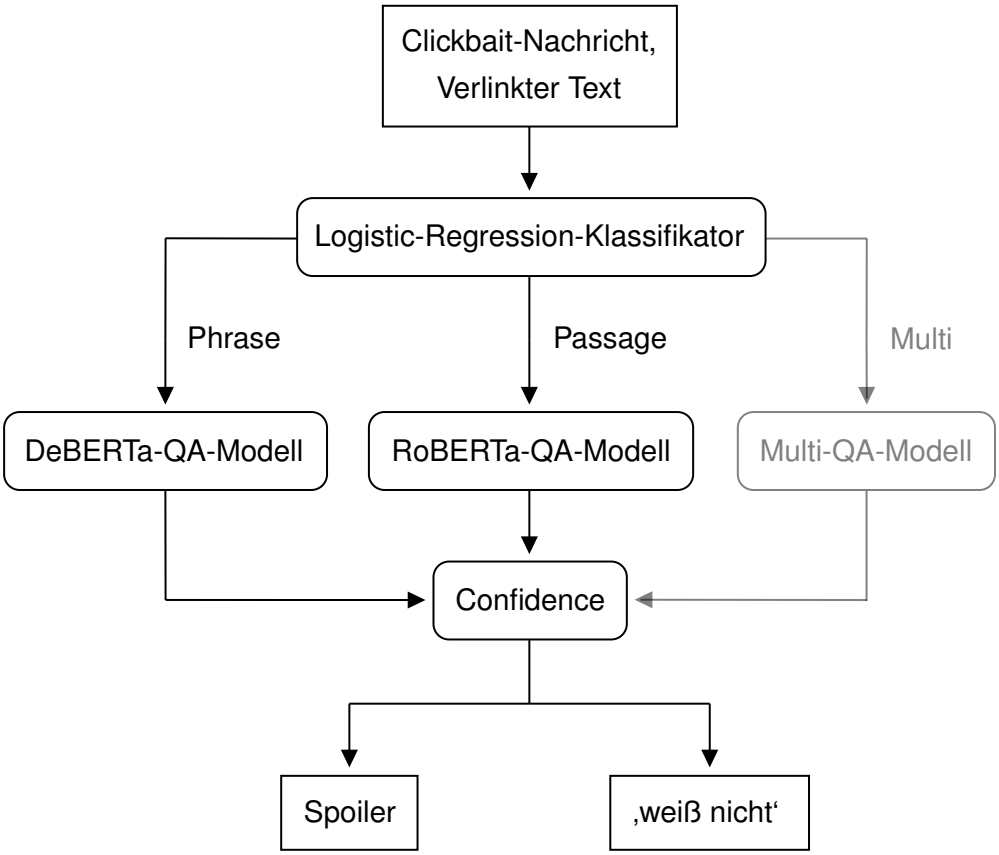


Abbildung 26: Clickbait-Spoiling-Ansatz in einem Diagramm dargestellt. Anhand der Clickbait-Nachricht und dem verlinkten Text sagt uns ein Logistic-Regression-Klassifikator mit 74.9 % Accuracy voraus, ob die Spoilerklasse Phrase oder Passage zu erwarten ist. Wird die Klasse Phrase vorhergesagt, lassen wir ein $\text{DeBERTa}_{\text{large}}$-Modell und für die Klasse Passage ein $\text{RoBERTa}_{\text{large}}$-Modell spoilern. Anhand eines Schwellwertes (Confidence) filtern wir unzuverlässige Vorhersagen des Klassifikators aus, damit dessen Fehler sich nicht in den Spoiling-Teil propagiert. Zu den übrig bleibenden Clickbait-Posts geben wir einen Spoiler zurück, womit wir insgesamt rund 22 % aller Clickbait-Posts erfolgreich spoilern.

Literaturverzeichnis

[1] Back, S., Yu, S., Indurthi, S. R., Kim, J., und Choo, J. (2018). MemoReader: Large-Scale Reading Comprehension through Neural Memory Controller. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Seiten 2131–2140, Brussels, Belgium. Association for Computational Linguistics. DOI: https://doi.org/10.18653/v1/D18-1237.

[2] Banerjee, S. und Lavie, A. (2005). METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, Seiten 65–72. Association for Computational Linguistics. URL: https://aclanthology.org/W05-0909.

[3] Bhoir, V. und Potey, M. A. (2014). Question answering system: A heuristic approach. In *The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014)*, Seiten 165–170. DOI: https://doi.org/10.1109/ICADIWT.2014.6814704.

[4] Bies, A., Ferguson, M., Katz, K., und MacIntyre, R. (1995). Bracketing Guidelines For Treebank II Style Penn Treebank Project.

[5] Blom, J. und Hansen, K. (2015). Click bait: Forward-reference as lure in online news headlines. *Journal of Pragmatics*, 76:87–100. DOI: https://doi.org/10.1016/j.pragma.2014.11.010.

[6] Calijorne Soares, M. A. und Parreiras, F. S. (2020). A literature review on question answering techniques, paradigms and systems. *Journal of King Saud University - Computer and Information Sciences*, 32(6):635–646. DOI: https://doi.org/10.1016/j.jksuci.2018.08.005.

[7] Callison-Burch, C., Osborne, M., und Koehn, P. (2006). Re-evaluating the Role of Bleu in Machine Translation Research. In *11th Conference of the*

European Chapter of the Association for Computational Linguistics. Association for Computational Linguistics. URL: https://aclanthology.org/E06-1032.

[8] Cao, Y., Cimino, J. J., Ely, J., und Yu, H. (2010). Automatically extracting information needs from complex clinical questions. *Journal of Biomedical Informatics*, 43(6):962–971. DOI: https://doi.org/10.1016/j.jbi.2010.07.007.

[9] Chakraborty, A., Paranjape, B., Kakarla, S., und Ganguly, N. (2016). Stop Clickbait: Detecting and preventing clickbaits in online news media. Seiten 9–16. DOI: https://doi.org/10.1109/ASONAM.2016.7752207.

[10] Chen, A., Stanovsky, G., Singh, S., und Gardner, M. (2019). Evaluating Question Answering Evaluation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, Hong Kong, China. Association for Computational Linguistics. DOI: https://doi.org/10.18653/v1/D19-5817.

[11] Chen, Y., Conroy, N. J., und Rubin, V. L. (2015). Misleading Online Content: Recognizing Clickbait as "False News". In *Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection*, Seite 15–19. Association for Computing Machinery. DOI: https://doi.org/10.1145/2823465.2823467.

[12] Clark, C. und Gardner, M. (2018). Simple and Effective Multi-Paragraph Reading Comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics. DOI: https://doi.org/10.18653/v1/P18-1078.

[13] Clark, K., Luong, M.-T., Le, Q. V., und Manning, C. D. (2020). ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. *arXiv preprint arXiv:2003.10555*.

[14] Dai, Z. und Callan, J. (2019). Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval. *arXiv preprint arXiv:1910.10687*.

[15] Dai, Z., Lai, G., Yang, Y., und Le, Q. V. (2020). Funnel-Transformer: Filtering out Sequential Redundancy for Efficient Language Processing. *arXiv preprint arXiv:2006.03236*.

[16] Defazio, A., Bach, F., und Lacoste-Julien, S. (2014). SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives. *arXiv preprint arXiv:1407.0202*.

[17] Denkowski, M. und Lavie, A. (2011). Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, Seiten 85–91. Association for Computational Linguistics. URL: https://aclanthology.org/W11-2107.

[18] Denkowski, M. und Lavie, A. (2014). Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Seiten 376–380. Association for Computational Linguistics. DOI: https://doi.org/10.3115/v1/W14-3348.

[19] Devlin, J., Chang, M.-W., Lee, K., und Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Seiten 4171–4186. Association for Computational Linguistics. DOI: https://doi.org/10.18653/v1/N19-1423.

[20] Devopedia (2020). Question Answering. URL: https://devopedia.org/question-answering. Besucht am 28.07.2021.

[21] Dvorkin, J. (2016). Column: Why click-bait will be the death of journalism. URL: https://www.pbs.org/newshour/economy/what-you-dont-know-about-click-bait-journalism-could-kill-you. Besucht am 11.07.2021.

[22] Fang, Y., Sun, S., Gan, Z., Pillai, R., Wang, S., und Liu, J. (2020). Hierarchical Graph Network for Multi-hop Question Answering. Seiten 8823–8838. DOI: https://doi.org/10.18653/v1/2020.emnlp-main.710.

[23] Gao, L., Dai, Z., und Callan, J. (2021). COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List. *arXiv preprint arXiv:2104.07186*.

[24] Glass, M., Gliozzo, A., Chakravarti, R., Ferritto, A., Pan, L., Bhargav, G., Garg, D., und Sil, A. (2020). Span Selection Pre-training for Question Answering. Seiten 2773–2782. DOI: https://doi.org/10.18653/v1/2020.acl-main.247.

[25] Gokaslan, A. und Cohen, V. (2019). OpenWebText Corpus. URL: `http://Skylion007.github.io/OpenWebTextCorpus`.

[26] Hadi, A. und Chatterjee, S. (2013). *Regression Analysis by Example*. Wiley Series in Probability and Statistics. Wiley. URL: `https://books.google.de/books?id=86MCAZaY1noC`.

[27] Han, S., Wang, X., Bendersky, M., und Najork, M. (2020). Learning-to-Rank with BERT in TF-Ranking. *arXiv preprint arXiv:2004.08476*.

[28] He, P., Liu, X., Gao, J., und Chen, W. (2020). DeBERTa: Decoding-enhanced BERT with Disentangled Attention. *arXiv preprint arXiv:2006.03654*.

[29] Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L., und Levy, O. (2019). SpanBERT: Improving Pre-training by Representing and Predicting Spans. *arXiv preprint arXiv:1907.10529*.

[30] Joshi, M., Choi, E., Weld, D., und Zettlemoyer, L. (2017). TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics. DOI: `https://doi.org/10.18653/v1/P17-1147`.

[31] Kané, H., Kocyigit, Y., Ajanoh, P., Abdalla, A., und Coulibali, M. (2019). Towards Neural Language Evaluators. *arXiv preprint arXiv:1909.09268*.

[32] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., und Soricut, R. (2019). ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *arXiv preprint arXiv:1909.11942*.

[33] Landis, J. R. und Koch, G. G. (1977). The Measurement of Observer Agreement for Categorical Data. *Biometrics*. URL: `http://www.jstor.org/stable/2529310`.

[34] Li, X., Huang, X.-J., und Wu, L.-d. (2005). Question Classification using Multiple Classifiers. In *Proceedings of the Fifth Workshop on Asian Language Resources (ALR-05) and First Symposium on Asian Language Resources Network (ALRN)*. URL: `https://aclanthology.org/I05-4009`.

[35] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., und Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.

[36] Llopis, F., González, J., und Ferrández, A. (2002). Using a Passage Retrieval System to Support Question Answering Process. Seiten 61–69. DOI: https://doi.org/10.1007/3-540-46043-8_5.

[37] Loewenstein, G. (1994). The Psychology of Curiosity: A Review and Reinterpretation. *Psychological Bulletin*, 116:75–98. DOI: https://doi.org/10.1037/0033-2909.116.1.75.

[38] Michael, B. (2015). Newspapers Fact Sheet. URL: https://assets.pewresearch.org/wp-content/uploads/sites/13/2017/05/30142603/state-of-the-news-media-report-2015-final.pdf. Besucht am 28.08.2021.

[39] Neves, M. und Leser, U. (2015). Question answering for Biology. *Methods*, 74:36–46. URL: 10.1016/j.ymeth.2014.10.023.

[40] Newman, N. (2017). Overview and Key Findings of the 2017 Report. URL: https://www.digitalnewsreport.org/survey/2017/overview-key-findings-2017/. Besucht am 11.07.2021.

[41] Papineni, K., Roukos, S., Ward, T., und Zhu, W.-J. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Seiten 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics. DOI: https://doi.org/10.3115/1073083.1073135.

[42] Post, M. (2018). A Call for Clarity in Reporting BLEU Scores. *arXiv preprint arXiv:1804.08771*.

[43] Potthast, M., Gollub, T., Komlossy, K., Schuster, S., Wiegmann, M., Garces Fernandez, E., Hagen, M., und Stein, B. (2018a). Crowdsourcing a Large Corpus of Clickbait on Twitter. In *27th International Conference on Computational Linguistics (COLING 2018)*, Seiten 1498–1507. The COLING 2018 Organizing Committee. URL: https://www.aclweb.org/anthology/C18-1127/.

[44] Potthast, M., Gollub, T., Komlossy, K., Schuster, S., Wiegmann, M., Garces Fernandez, E., Hagen, M., und Stein, B. (2018b). Webis-Clickbait-17. URL: https://webis.de/data/webis-clickbait-17.html.

[45] Puschmann, J. (2019). Crowdsourcing a Corpus for Clickbait Spoiling. Bachelorarbeit, Bauhaus-Universität Weimar, Fakultät Medien, Computer Science and Media. URL: https://webis.de/downloads/theses/papers/puschmann_2019.pdf.

[46] Qin, G. und Eisner, J. (2021). Learning How to Ask: Querying LMs with Mixtures of Soft Prompts. *arXiv preprint arXiv:2104.06599*.

[47] Qu, Y., Ding, Y., Liu, J., Liu, K., Ren, R., Zhao, W. X., Dong, D., Wu, H., und Wang, H. (2021). RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. *arXiv preprint arXiv:2010.08191*.

[48] Rajpurkar, P., Zhang, J., Lopyrev, K., und Liang, P. (2016). SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Seiten 2383–2392, Austin, Texas. Association for Computational Linguistics. DOI: `https://doi.org/10.18653/v1/D16-1264`.

[49] Roberts, I. und Gaizauskas, R. (2004). Evaluating Passage Retrieval Approaches for Question Answering. DOI: `https://doi.org/10.1007/978-3-540-24752-4_6`.

[50] Robertson, S. und Zaragoza, H. (2009). The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* DOI: `https://doi.org/10.1561/1500000019`.

[51] Sen, P. und Saffari, A. (2020). What do Models Learn from Question Answering Datasets? *arXiv preprint arXiv:2004.03490*.

[52] Seo, M., Kembhavi, A., Farhadi, A., und Hajishirzi, H. (2018). Bidirectional Attention Flow for Machine Comprehension. *arXiv preprint arXiv:1611.01603*.

[53] Song, K., Tan, X., Qin, T., Lu, J., und Liu, T.-Y. (2020). MPNet: Masked and Permuted Pre-training for Language Understanding. *arXiv preprint arXiv:2004.09297*.

[54] Tayyar Madabushi, H. und Lee, M. (2016). High Accuracy Rule-based Question Classification using Question Syntax and Semantics. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee. URL: `https://aclanthology.org/C16-1116`.

[55] Ter-Akopyan, B. (2017). Korpuskonstruktion und Entwicklung einer Pipeline für Clickbait-Spoiling. Bachelorarbeit, Bauhaus-Universität Weimar, Fakultät Medien, Computer Science and Media. URL: `https://webis.de/downloads/theses/papers/terakopyan_2017.pdf`.

[56] Tu, M., Huang, K., Wang, G., Huang, J., He, X., und Zhou, B. (2020). Select, Answer and Explain: Interpretable Multi-Hop Reading Comprehension over Multiple Documents. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, Seiten 9073–9080. AAAI Press. URL: `https://aaai.org/ojs/index.php/AAAI/article/view/6441`.

[57] Wu, W., Wang, F., Yuan, A., Wu, F., und Li, J. (2020). Coreference Resolution as Query-based Span Prediction. *arXiv preprint arXiv:1911.01746*.

[58] Xu, P., Wu, C.-S., Madotto, A., und Fung, P. (2019). Clickbait? Sensational Headline Generation with Auto-tuned Reinforcement Learning. *arXiv preprint arXiv:1909.03582*.

[59] Yamada, I., Asai, A., Shindo, H., Takeda, H., und Matsumoto, Y. (2020). LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention. DOI: `https://doi.org/10.18653/v1/2020.emnlp-main.523`.

[60] Yang, Y. und Joachims, T. (2008). Text categorization. *Scholarpedia*. DOI: `https://doi.org/10.4249/scholarpedia.4242`. Besucht am 28.08.2021.

[61] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., und Le, Q. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237*.

[62] Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., und Manning, C. D. (2018). HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. *arXiv preprint arXiv:1809.09600*.

[63] Yin, J., Jiang, X., Lu, Z., Shang, L., Li, H., und Li, X. (2016). Neural Generative Question Answering. *arXiv preprint arXiv:1512.01337*.

[64] Zaheer, M., Guruganesh, G., Dubey, A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., und Ahmed, A. (2021). Big Bird: Transformers for Longer Sequences. *arXiv preprint arXiv:2007.14062*.

[65] Zannettou, S., Sirivianos, M., Blackburn, J., und Kourtellis, N. (2019). The Web of False Information: Rumors, Fake News, Hoaxes, Clickbait, and Various Other Shenanigans. *J. Data and Information Quality*. DOI: `https://doi.org/10.1145/3309699`.

[66] Zaragoza, H., Craswell, N., Taylor, M., Saria, S., und Robertson, S. (2004). Microsoft Cambridge at TREC 13: Web and Hard Tracks.

[67] Zhang, D. und Lee, W. S. (2003). Question Classification Using Support Vector Machines. Association for Computing Machinery. DOI: `https://doi.org/10.1145/860435.860443`.

[68] Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., und Artzi, Y. (2020a). BERTScore: Evaluating Text Generation with BERT. *arXiv preprint arXiv:1904.09675*.

[69] Zhang, Z., Yang, J., und Zhao, H. (2020b). Retrospective Reader for Machine Reading Comprehension. *arXiv preprint arXiv:2001.09694*.

Anhang A

Klassifikation

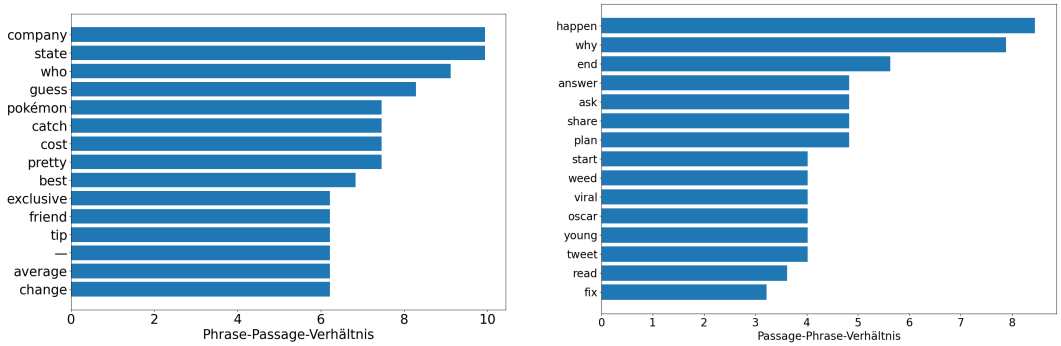


Abbildung 27: 15 höchsten Verhältnisse zu wie oft Wort-1-Gramme in Clickbait-Nachrichten mit (links) Spoilern der Klasse Phrase im Vergleich zu Passage und (rechts) umgekehrt vorkommen. Ein Verhältnis von zwei bedeutet doppelt so oft.

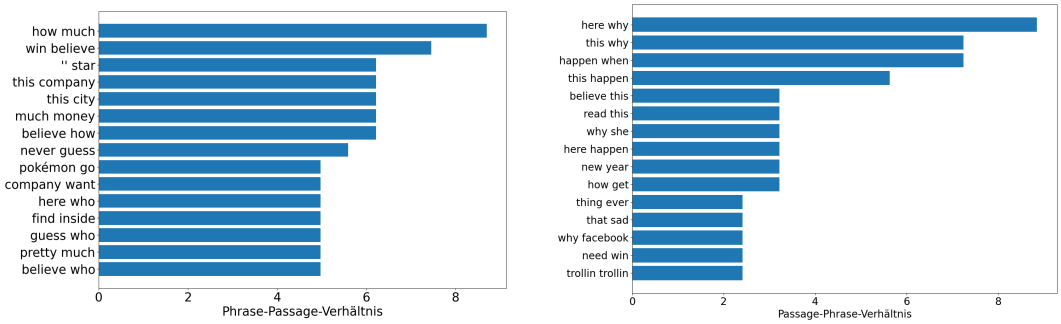


Abbildung 28: 15 höchsten Verhältnisse zu wie oft Wort-2-Gramme in Clickbait-Nachrichten mit (links) Spoilern der Klasse Phrase im Vergleich zu Passage und (rechts) umgekehrt vorkommen. Ein Verhältnis von zwei bedeutet doppelt so oft.

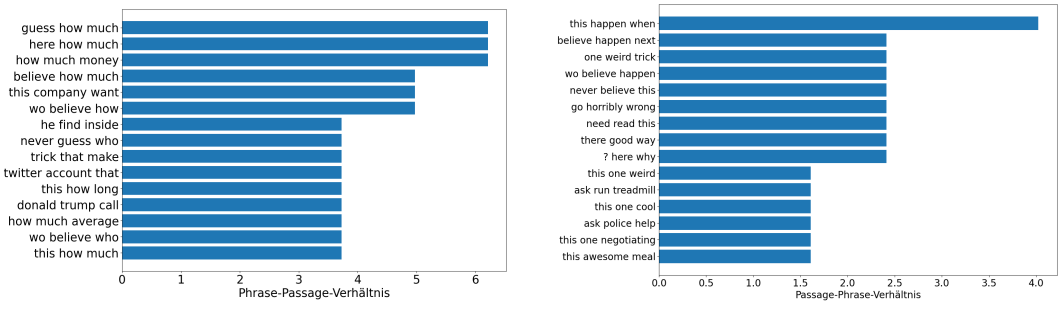


Abbildung 29: 15 höchsten Verhältnisse zu wie oft Wort-3-Gramme in Clickbait-Nachrichten mit (links) Spoilern der Klasse Phrase im Vergleich zu Passage und (rechts) umgekehrt vorkommen. Ein Verhältnis von zwei bedeutet doppelt so oft.

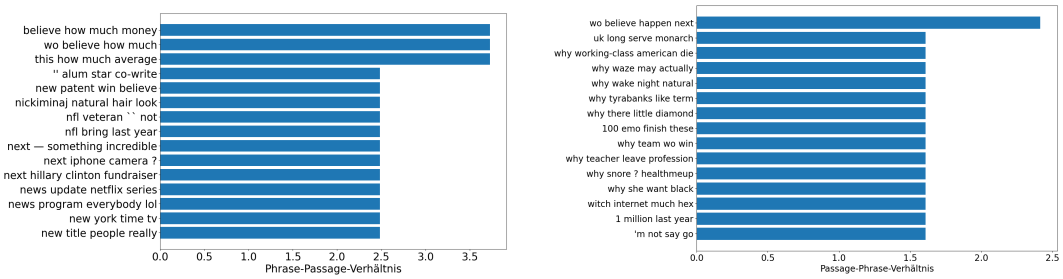


Abbildung 30: 15 höchsten Verhältnisse zu wie oft Wort-4-Gramme in Clickbait-Nachrichten mit (links) Spoilern der Klasse Phrase im Vergleich zu Passage und (rechts) umgekehrt vorkommen. Ein Verhältnis von zwei bedeutet doppelt so oft.

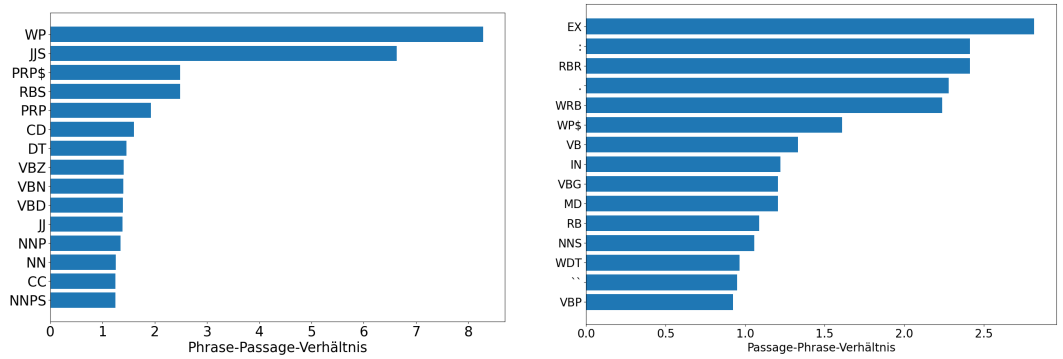


Abbildung 31: 15 höchsten Verhältnisse zu wie oft POS-1-Gramme in Clickbait-Nachrichten mit (links) Spoilern der Klasse Phrase im Vergleich zu Passage und (rechts) umgekehrt vorkommen. Ein Verhältnis von zwei bedeutet doppelt so oft.

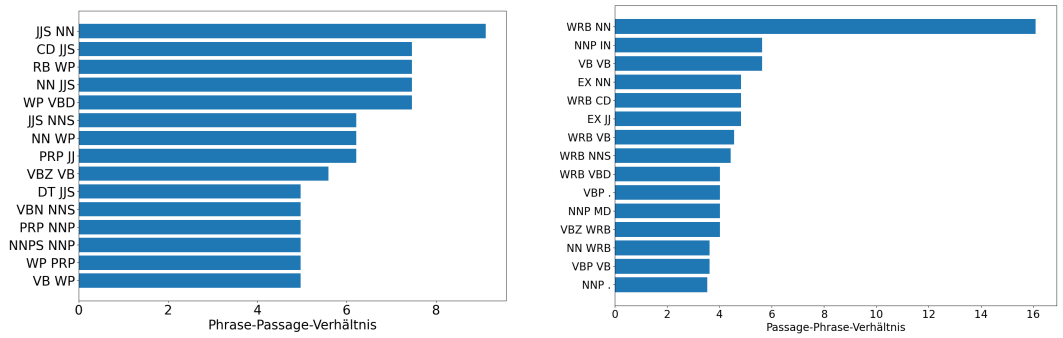


Abbildung 32: 15 höchsten Verhältnisse zu wie oft POS-2-Gramme in Clickbait-Nachrichten mit (links) Spoilern der Klasse Phrase im Vergleich zu Passage und (rechts) umgekehrt vorkommen. Ein Verhältnis von zwei bedeutet doppelt so oft.

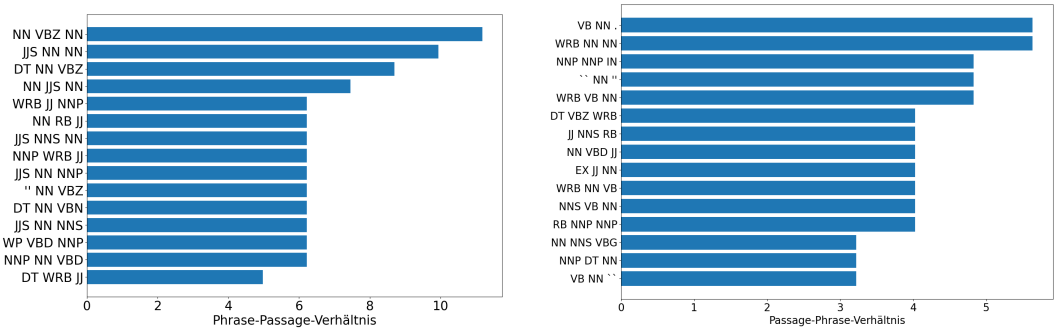


Abbildung 33: 15 höchsten Verhältnisse zu wie oft POS-3-Gramme in Clickbait-Nachrichten mit (links) Spoilern der Klasse Phrase im Vergleich zu Passage und (rechts) umgekehrt vorkommen. Ein Verhältnis von zwei bedeutet doppelt so oft.

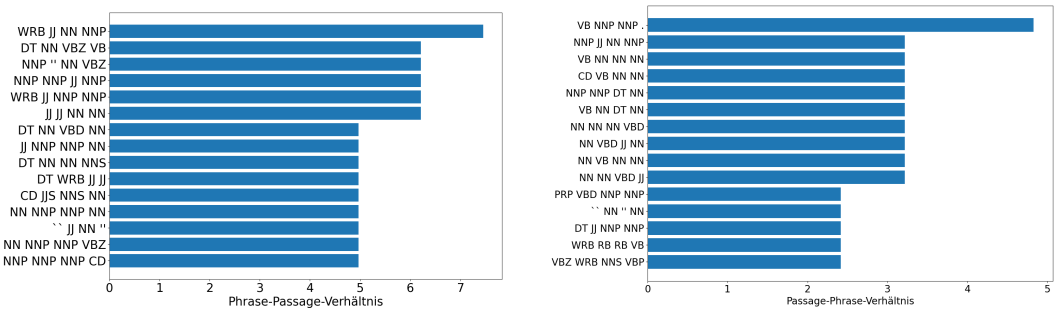


Abbildung 34: 15 höchsten Verhältnisse zu wie oft POS-4-Gramme in Clickbait-Nachrichten mit (links) Spoilern der Klasse Phrase im Vergleich zu Passage und (rechts) umgekehrt vorkommen. Ein Verhältnis von zwei bedeutet doppelt so oft.

Anhang B

Spoiling

Tabelle 44: Effektivität der auf SQuADv1.1 feingetunten Modelle beim Spoilern der Spoilerklasse Passage abhängig von der Anzahl an Trainingsschritten. Gemessen auf Passage-1kTrain-Valid (102 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem bei Metik auch in Klammern angegebenen Schwellwert. In der Fußnote ist die Version des Modells angegeben. Hyperparameter: Batchgröße 16, Sequenzlänge 384, Lernrate 3e-5, Rest Standardwerte im `run_qa.py` Skript von Huggingface. 3 Epochen entsprechen 16,500 Schritten. Bester Wert für jedes Modell hervorgehoben.

Modell	Metrik	Schritt6k	Schritt7k	Schritt8k	Schritt9k	Schritt10k	Schritt11k
$\text{ALBERT}_{\text{v2-base}}$	BLEU-4 (>30)	5.88 (8)	7.31 (9)	10.31 (14)	9.30 (12)	9.77 (13)	10.04 (14)
	METEOR-F (>70)	18.74 (6)	20.74 (6)	22.95 (12)	22.35 (10)	25.21 (11)	24.57 (10)
	BERTScore-F1 (>60)	31.90 (6)	33.05 (9)	34.78 (12)	33.99 (9)	34.67 (12)	33.64 (11)
$\text{BERT}_{\text{base-uncased}}$	BLEU-4 (>30)	6.46 (10)	8.26 (12)	4.95 (7)	7.09 (9)	7.71 (9)	5.91 (9)
	METEOR-F (>70)	18.38 (8)	18.64 (9)	17.21 (7)	19.74 (8)	21.34 (10)	18.02 (8)
	BERTScore-F1 (>60)	30.08 (9)	31.86 (11)	29.66 (10)	31.47 (11)	32.48 (13)	29.62 (8)
$\text{BERT}_{\text{base-cased}}$	BLEU-4 (>30)	7.68 (9)	6.28 (9)	6.01 (8)	4.60 (7)	2.73 (4)	6.79 (9)
	METEOR-F (>70)	18.82 (8)	17.74 (9)	17.49 (6)	17.55 (5)	14.46 (3)	18.18 (8)
	BERTScore-F1 (>60)	31.96 (9)	29.75 (10)	29.60 (8)	30.00 (7)	27.73 (5)	30.77 (8)
$\text{ELECTRA}_{\text{discriminator-base}}$	BLEU-4 (>30)	12.20 (16)	14.10 (17)	12.11 (16)	**13.64 (16)**	12.32 (15)	12.65 (16)
	METEOR-F (>70)	27.44 (13)	28.34 (16)	28.59 (13)	**29.48 (15)**	28.09 (15)	29.49 (16)
	BERTScore-F1 (>60)	33.92 (16)	35.79 (19)	33.73 (16)	**35.35 (18)**	34.97 (20)	34.01 (19)
$\text{FunnelTransformer}_{\text{small}}$	BLEU-4 (>30)	14.27 (18)	**15.73 (22)**	12.64 (16)	13.24 (17)	12.36 (14)	11.59 (16)
	METEOR-F (>70)	29.82 (18)	**32.14 (20)**	26.19 (15)	26.78 (16)	28.63 (15)	28.45 (16)
	BERTScore-F1 (>60)	38.14 (21)	**39.53 (20)**	36.08 (17)	35.63 (16)	35.87 (19)	36.61 (16)
$\text{MPNet}_{\text{base}}$	BLEU-4 (>30)	9.67 (11)	12.55 (14)	15.13 (18)	13.09 (18)	13.17 (16)	**16.44 (19)**
	METEOR-F (>70)	23.72 (10)	27.46 (15)	28.23 (17)	29.04 (15)	30.48 (15)	**31.06 (19)**
	BERTScore-F1 (>60)	32.01 (12)	33.67 (17)	36.53 (20)	34.62 (20)	34.31 (16)	**37.94 (18)**
$\text{RoBERTa}_{\text{base}}$	BLEU-4 (>30)	12.48 (16)	11.44 (17)	6.60 (10)	13.24 (19)	10.38 (14)	13.84 (19)
	METEOR-F (>70)	27.39 (18)	23.62 (15)	18.75 (9)	25.27 (15)	23.26 (13)	30.87 (19)
	BERTScore-F1 (>60)	35.07 (17)	34.50 (15)	30.93 (9)	35.60 (18)	34.15 (13)	35.88 (19)

Tabelle 45: Effektivität der auf SQuADv1.1 feingetunten Modelle beim Spoilern der Spoilerklasse Passage abhängig von der Anzahl an Trainingsschritten. Gemessen auf Passage-1kTrain-Valid (102 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem bei Metik auch in Klammern angegebenen Schwellwert. In der Fußnote ist die Version des Modells angegeben. Hyperparameter: Batchgröße 16, Sequenzlänge 384, Lernrate 3e-5, Rest Standardwerte im `run_qa.py` Skript von Huggingface. 3 Epochen entsprechen 16,500 Schritten. Bester Wert für jedes Modell hervorgehoben.

Modell	Metrik	Schritt12k	Schritt13k	Schritt14k	Schritt15k	Schritt16k	Schrittende
$\text{ALBERT}_{\text{v2-base}}$	BLEU-4 (>30)	9.72 (11)	8.80 (10)	**11.85 (15)**	9.40 (12)	10.52 (13)	10.60 (13)
	METEOR-F (>70)	24.24 (11)	23.20 (11)	**27.86 (15)**	25.74 (12)	26.34 (13)	26.47 (13)
	BERTScore-F1 (>60)	34.01 (13)	32.19 (13)	**34.59 (18)**	33.15 (15)	34.19 (16)	34.34 (16)
$\text{BERT}_{\text{base-uncased}}$	BLEU-4 (>30)	6.37 (8)	**8.13 (11)**	6.05 (9)	8.14 (11)	7.65 (10)	8.12 (11)
	METEOR-F (>70)	19.67 (7)	**19.87 (10)**	17.69 (8)	19.76 (10)	19.72 (9)	19.73 (10)
	BERTScore-F1 (>60)	30.97 (10)	**31.88 (12)**	30.71 (10)	31.59 (12)	31.24 (11)	31.43 (12)
$\text{BERT}_{\text{base-cased}}$	BLEU-4 (>30)	9.58 (12)	9.53 (12)	9.48 (12)	**10.40 (13)**	9.42 (12)	9.96 (13)
	METEOR-F (>70)	19.65 (10)	22.17 (10)	21.65 (10)	**22.21 (10)**	21.64 (9)	21.47 (10)
	BERTScore-F1 (>60)	30.40 (12)	32.22 (12)	33.50 (12)	**33.30 (13)**	32.00 (12)	32.52 (13)
$\text{ELECTRA}_{\text{discriminator-base}}$	BLEU-4 (>30)	13.43 (17)	10.18 (14)	10.44 (13)	12.61 (16)	12.64 (16)	11.90 (15)
	METEOR-F (>70)	28.40 (16)	24.47 (12)	25.10 (11)	29.51 (15)	29.11 (15)	28.36 (15)
	BERTScore-F1 (>60)	35.04 (21)	32.32 (15)	32.15 (16)	34.72 (20)	34.83 (19)	34.53 (19)
$\text{FunnelTransformer}_{\text{small}}$	BLEU-4 (>30)	13.34 (16)	13.59 (17)	13.67 (17)	13.48 (17)	11.56 (15)	12.82 (16)
	METEOR-F (>70)	30.37 (17)	28.11 (17)	28.60 (17)	28.32 (18)	27.80 (15)	27.24 (16)
	BERTScore-F1 (>60)	37.91 (19)	37.49 (18)	37.14 (17)	37.11 (18)	35.77 (16)	36.50 (18)
$\text{MPNet}_{\text{base}}$	BLEU-4 (>30)	13.86 (16)	13.80 (16)	12.71 (15)	15.56 (18)	14.57 (18)	12.50 (16)
	METEOR-F (>70)	25.34 (13)	26.91 (14)	23.26 (13)	28.89 (18)	28.18 (18)	26.39 (15)
	BERTScore-F1 (>60)	35.27 (16)	35.38 (15)	35.27 (14)	36.83 (18)	35.56 (19)	34.23 (16)
$\text{RoBERTa}_{\text{base}}$	BLEU-4 (>30)	11.56 (14)	15.23 (19)	13.21 (17)	**15.47 (19)**	13.17 (16)	14.47 (18)
	METEOR-F (>70)	24.93 (13)	28.42 (18)	28.85 (18)	**28.78 (20)**	25.07 (17)	26.44 (18)
	BERTScore-F1 (>60)	33.56 (16)	37.43 (21)	36.60 (20)	**38.63 (19)**	36.92 (17)	37.28 (17)

Tabelle 46: Effektivität der auf SQuADv1.1 feingetunten Modelle beim Spoilern der Spoilerklasse Passage abhängig von der Anzahl an Trainingsschritten. Gemessen auf Passage-1kTrain-Valid (102 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem bei Metik auch in Klammern angegebenen Schwellwert. In der Fußnote ist die Version des Modells angegeben. Hyperparameter: Batchgröße 8, Sequenzlänge 384, Lernrate 3e-5, Rest Standardwerte im `run_qa.py` Skript von Huggingface. 3 Epochen entsprechen 33,000 Schritten. Bester Wert für jedes Modell hervorgehoben.

Modell	Metrik	Schritt12k	Schritt14k	Schritt16k	Schritt18k	Schritt20k	Schritt22k
$\text{BigBird}_{\text{RoBERTa-large}}$	BLEU-4 (>30)	14.33 (18)	13.98 (18)	11.89 (16)	12.72 (15)	14.56 (17)	13.4 (16)
	METEOR-F (>70)	29.07 (19)	27.74 (19)	26.52 (15)	25.78 (15)	27.93 (18)	27.09 (16)
	BERTScore-F1 (>60)	36.80 (19)	35.88 (20)	34.86 (17)	36.77 (16)	36.99 (17)	36.18 (17)
$\text{DeBERTa}_{\text{large}}$	BLEU-4 (>30)	9.78 (12)	**16.02 (19)**	11.38 (13)	15.37 (18)	13.15 (16)	14.00 (17)
	METEOR-F (>70)	23.77 (13)	**27.71 (19)**	22.79 (13)	28.36 (18)	28.89 (16)	27.75 (17)
	BERTScore-F1 (>60)	33.49 (11)	**38.75 (19)**	35.10 (14)	38.79 (18)	36.38 (18)	37.02 (15)
$\text{RoBERTa}_{\text{large}}$	BLEU-4 (>30)	14.03 (15)	13.59 (16)	12.23 (14)	13.24 (15)	17.91 (22)	15.01 (17)
	METEOR-F (>70)	25.77 (15)	27.42 (17)	27.65 (15)	23.11 (16)	30.58 (24)	30.93 (18)
	BERTScore-F1 (>60)	35.05 (18)	35.70 (20)	34.82 (18)	32.88 (16)	39.40 (24)	36.42 (18)

Modell	Metrik	Schritt24k	Schritt26k	Schritt28k	Schritt30k	Schritt32k	Schrittende
$\text{BigBird}_{\text{RoBERTa-large}}$	BLEU-4 (>30)	**15.64 (19)**	10.39 (13)	13.49 (17)	12.40 (14)	14.38 (17)	14.01 (17)
	METEOR-F (>70)	**31.67 (21)**	23.38 (14)	27.13 (18)	23.46 (15)	25.30 (18)	25.13 (18)
	BERTScore-F1 (>60)	**36.66 (22)**	32.44 (14)	35.09 (18)	34.68 (16)	34.88 (18)	34.57 (18)
$\text{DeBERTa}_{\text{large}}$	BLEU-4 (>30)	13.51 (15)	11.03 (12)	12.75 (16)	11.69 (14)	11.77 (14)	12.65 (14)
	METEOR-F (>70)	26.45 (16)	22.86 (14)	25.76 (16)	24.07 (15)	23.50 (14)	25.83 (14)
	BERTScore-F1 (>60)	36.52 (15)	33.82 (15)	35.35 (18)	34.34 (14)	34.20 (15)	34.93 (16)
$\text{RoBERTa}_{\text{large}}$	BLEU-4 (>30)	16.30 (19)	**19.88 (23)**	17.52 (20)	18.14 (21)	18.95 (22)	19.24 (23)
	METEOR-F (>70)	27.74 (20)	**34.31 (24)**	31.24 (22)	32.04 (24)	31.93 (24)	32.59 (25)
	BERTScore-F1 (>60)	37.06 (22)	**41.31 (26)**	39.07 (23)	40.20 (26)	40.36 (25)	40.53 (26)

Tabelle 47: Effektivität der auf Passage-1kTrain-Train (398 Einträge) feingetunten Modelle beim Spoilern der Spoilerklasse Passage abhängig von der Anzahl an Trainingsschritten. Gemessen auf Passage-1kTrain-Valid (102 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem bei Metik auch in Klammern angegebenen Schwellwert. In der Fußnote ist die Version des Modells angegeben. Hyperparameter: Batchgröße 16, Sequenzlänge 384, Lernrate 3e-5, Rest Standardwerte im `run_qa.py` Skript von Huggingface. 15 Epochen entsprechen 1,200 Schritten. Bester Wert für jedes Modell hervorgehoben.

Modell	Metrik	Schritt100	Schritt200	Schritt300	Schritt400	Schritt500	Schritt600
$\text{BERT}_{\text{base-uncased}}$	BLEU-4 (>30)	9.89 (13)	**12.10 (16)**	8.84 (12)	8.64 (11)	7.19 (11)	8.99 (12)
	METEOR-F (>70)	20.96 (13)	**24.95 (16)**	20.68 (12)	20.44 (12)	19.01 (10)	22.19 (11)
	BERTScore-F1 (>60)	35.27 (10)	**37.12 (14)**	34.63 (10)	31.28 (11)	29.69 (8)	31.29 (11)
$\text{BERT}_{\text{base-cased}}$	BLEU-4 (>30)	13.16 (14)	11.96 (14)	**15.41 (19)**	13.07 (16)	10.26 (13)	10.32 (14)
	METEOR-F (>70)	26.21 (15)	22.25 (13)	**25.71 (17)**	24.42 (15)	21.77 (12)	24.42 (14)
	BERTScore-F1 (>60)	38.26 (14)	36.46 (12)	**38.94 (18)**	36.99 (14)	32.13 (11)	34.22 (12)
$\text{ALBERT}_{\text{v2-base}}$	BLEU-4 (>30)	15.71 (23)	17.33 (26)	**20.41 (27)**	16.55 (22)	15.39 (22)	15.48 (23)
	METEOR-F (>70)	28.99 (19)	30.39 (22)	**33.22 (24)**	29.67 (21)	28.15 (19)	29.14 (20)
	BERTScore-F1 (>60)	39.52 (15)	40.88 (14)	**42.49 (20)**	39.49 (18)	38.91 (15)	38.38 (19)
$\text{ELECTRA}_{\text{discriminator-base}}$	BLEU-4 (>30)	15.32 (20)	12.92 (18)	15.46 (22)	13.83 (19)	**16.81 (24)**	15.38 (22)
	METEOR-F (>70)	26.74 (16)	26.66 (14)	29.59 (17)	27.03 (14)	**33.22 (21)**	32.50 (18)
	BERTScore-F1 (>60)	39.36 (13)	37.79 (12)	38.83 (14)	35.32 (13)	**37.90 (18)**	35.32 (15)
$\text{RoBERTa}_{\text{base}}$	BLEU-4 (>30)	12.59 (16)	18.98 (24)	17.04 (22)	20.71 (26)	**21.24 (27)**	18.04 (24)
	METEOR-F (>70)	22.13 (15)	29.94 (20)	28.93 (19)	33.61 (23)	**31.46 (24)**	32.48 (21)
	BERTScore-F1 (>60)	38.14 (11)	42.34 (20)	40.52 (17)	43.56 (20)	**42.47 (20)**	40.65 (18)
$\text{MPNet}_{\text{base}}$	BLEU-4 (>30)	10.99 (17)	15.68 (21)	19.44 (25)	16.39 (20)	**21.06 (29)**	15.17 (18)
	METEOR-F (>70)	24.52 (14)	30.86 (19)	34.16 (22)	31.01 (19)	**35.93 (25)**	32.35 (16)
	BERTScore-F1 (>60)	36.72 (13)	38.84 (16)	40.91 (19)	37.21 (17)	**41.57 (22)**	33.94 (16)
$\text{FunnelTransformer}_{\text{small}}$	BLEU-4 (>30)	16.49 (21)	21.90 (27)	21.76 (29)	22.14 (30)	**21.09 (29)**	18.57 (25)
	METEOR-F (>70)	31.23 (21)	35.53 (26)	35.88 (26)	35.21 (25)	**36.04 (25)**	34.54 (22)
	BERTScore-F1 (>60)	40.76 (17)	44.49 (24)	41.89 (23)	43.95 (22)	**43.04 (23)**	41.05 (20)

Tabelle 48: Effektivität der auf Passage-1kTrain-Train (398 Einträge) feingetunten Modelle beim Spoilern der Spoilerklasse Passage abhängig von der Anzahl an Trainingsschritten. Gemessen auf Passage-1kTrain-Valid (102 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem bei Metik auch in Klammern angegebenen Schwellwert. In der Fußnote ist die Version des Modells angegeben. Hyperparameter: Batchgröße 16, Sequenzlänge 384, Lernrate 3e-5, Rest Standardwerte im `run_qa.py` Skript von Huggingface. 15 Epochen entsprechen 1,200 Schritten. Bester Wert für jedes Modell hervorgehoben.

Modell	Metrik	Schritt700	Schritt800	Schritt900	Schritt1k	Schritt1,1k	Schrittende
$\text{BERT}_{\text{base-uncased}}$	BLEU-4 (>30)	6.40 (10)	9.13 (13)	8.89 (11)	9.43 (13)	8.32 (11)	6.34 (9)
	METEOR-F (>70)	19.92 (9)	21.33 (11)	20.94 (10)	22.04 (12)	20.19 (10)	16.99 (9)
	BERTScore-F1 (>60)	29.89 (8)	31.22 (9)	30.35 (9)	30.30 (11)	29.94 (8)	28.68 (7)
$\text{BERT}_{\text{base-cased}}$	BLEU-4 (>30)	9.11 (12)	9.38 (11)	9.37 (13)	8.06 (12)	10.62 (14)	10.72 (14)
	METEOR-F (>70)	21.59 (11)	20.68 (11)	22.23 (11)	21.25 (10)	24.44 (13)	24.35 (13)
	BERTScore-F1 (>60)	31.77 (11)	31.24 (10)	32.24 (10)	30.76 (8)	32.86 (13)	33.21 (13)
$\text{ALBERT}_{\text{v2-base}}$	BLEU-4 (>30)	14.24 (19)	16.15 (23)	15.85 (22)	16.27 (23)	15.70 (22)	15.70 (22)
	METEOR-F (>70)	26.06 (17)	27.83 (18)	27.42 (19)	28.47 (20)	28.74 (19)	28.54 (19)
	BERTScore-F1 (>60)	38.06 (15)	38.61 (16)	38.42 (17)	38.97 (17)	38.26 (16)	38.28 (18)
$\text{ELECTRA}_{\text{discriminator-base}}$	BLEU-4 (>30)	10.94 (14)	13.43 (18)	11.47 (15)	10.89 (15)	10.08 (13)	10.36 (13)
	METEOR-F (>70)	26.00 (14)	29.88 (15)	27.78 (14)	28.23 (13)	25.71 (12)	26.57 (13)
	BERTScore-F1 (>60)	31.04 (11)	33.25 (13)	31.85 (10)	30.81 (10)	30.02 (9)	30.20 (13)
$\text{RoBERTa}_{\text{base}}$	BLEU-4 (>30)	18.01 (25)	15.18 (21)	15.91 (23)	14.15 (20)	15.10 (22)	14.55 (21)
	METEOR-F (>70)	33.50 (20)	30.29 (15)	30.04 (18)	28.60 (15)	29.96 (16)	29.69 (16)
	BERTScore-F1 (>60)	40.19 (17)	36.85 (13)	36.48 (14)	36.01 (12)	37.29 (12)	36.78 (17)
$\text{MPNet}_{\text{base}}$	BLEU-4 (>30)	12.38 (17)	13.50 (17)	10.37 (15)	12.33 (16)	10.40 (14)	11.85 (16)
	METEOR-F (>70)	27.84 (12)	28.57 (16)	23.58 (10)	26.37 (14)	24.62 (12)	26.37 (14)
	BERTScore-F1 (>60)	32.01 (12)	31.44 (13)	28.28 (9)	29.63 (12)	28.77 (11)	29.99 (12)
$\text{FunnelTransformer}_{\text{small}}$	BLEU-4 (>30)	21.10 (28)	16.78 (24)	16.57 (23)	15.69 (23)	15.78 (23)	15.78 (23)
	METEOR-F (>70)	36.15 (24)	31.41 (18)	31.61 (18)	30.80 (18)	31.27 (18)	31.19 (18)
	BERTScore-F1 (>60)	41.33 (22)	37.98 (17)	37.67 (17)	37.98 (16)	37.96 (16)	37.97 (21)

Tabelle 49: Effektivität der zuerst auf SQuADv1.1 feingetunten und danach weiter auf Passage-1kTrain-Train (398 Einträge) feingetunten Modelle beim Spoilern der Spoilerklasse Passage mit der Anzahl an Trainingsschritten, die zum besten Ergebnis führen. Gemessen auf Passage-1kTrain-Valid (102 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem bei Metik auch in Klammern angegebenen Schwellwert. In der Fußnote ist die Version des Modells angegeben. Hyperparameter: Batchgröße 16, Sequenzlänge 384, Lernrate 3e-5, Rest Standardwerte im `run_qa.py` Skript von Huggingface. 15 Epochen entsprechen 1,200 Schritten. Bester Wert für jedes Modell hervorgehoben.

Modell	Metrik	Schritt100	Schritt200	Schritt300	Schritt400	Schritt500	Schritt600
$\text{ALBERT}_{\text{v2-base}}$	BLEU-4 (>30)	17.79 (21)	18.82 (24)	18.36 (22)	**24.51 (33)**	22.17 (29)	21.34 (26)
	METEOR-F (>70)	31.44 (21)	35.01 (22)	31.53 (20)	**38.42 (27)**	35.37 (26)	32.94 (24)
	BERTScore-F1 (>60)	41.26 (18)	41.61 (19)	41.21 (16)	**44.61 (24)**	43.56 (23)	42.27 (21)
$\text{BERT}_{\text{base-uncased}}$	BLEU-4 (>30)	17.83 (22)	16.27 (20)	16.64 (19)	15.00 (19)	**18.05 (22)**	16.32 (20)
	METEOR-F (>70)	29.74 (20)	28.64 (19)	29.68 (21)	28.56 (18)	**32.50 (20)**	30.49 (18)
	BERTScore-F1 (>60)	40.81 (21)	40.14 (17)	40.83 (20)	38.94 (15)	**39.86 (18)**	39.11 (16)
$\text{BERT}_{\text{base-cased}}$	BLEU-4 (>30)	14.04 (18)	14.94 (18)	16.11 (20)	17.18 (20)	17.55 (22)	17.00 (21)
	METEOR-F (>70)	24.10 (17)	25.57 (17)	26.66 (17)	27.98 (18)	27.61 (19)	27.75 (19)
	BERTScore-F1 (>60)	38.00 (13)	38.36 (15)	39.81 (15)	40.28 (15)	40.48 (17)	39.77 (17)
$\text{ELECTRA}_{\text{discriminator-base}}$	BLEU-4 (>30)	**25.78 (32)**	24.57 (29)	25.44 (31)	23.60 (28)	21.69 (28)	23.94 (30)
	METEOR-F (>70)	**39.87 (30)**	36.87 (26)	36.80 (28)	36.40 (25)	36.16 (24)	36.30 (27)
	BERTScore-F1 (>60)	**46.64 (27)**	45.13 (24)	45.42 (26)	43.92 (24)	42.31 (21)	43.97 (25)
$\text{FunnelTransformer}_{\text{small}}$	BLEU-4 (>30)	25.63 (33)	24.37 (32)	**28.59 (36)**	23.29 (30)	23.35 (31)	22.58 (30)
	METEOR-F (>70)	38.84 (31)	38.33 (28)	**40.95 (32)**	37.26 (26)	37.17 (27)	37.09 (26)
	BERTScore-F1 (>60)	46.23 (30)	45.88 (26)	**47.93 (29)**	45.26 (24)	43.69 (23)	43.48 (24)
$\text{MPNet}_{\text{base}}$	BLEU-4 (>30)	**30.16 (36)**	29.25 (35)	25.28 (31)	25.92 (32)	24.53 (30)	25.47 (30)
	METEOR-F (>70)	**40.68 (35)**	41.06 (33)	39.21 (29)	39.85 (30)	36.01 (29)	36.73 (28)
	BERTScore-F1 (>60)	**50.07 (32)**	49.03 (32)	46.46 (27)	45.01 (30)	43.45 (28)	42.46 (26)
$\text{RoBERTa}_{\text{base}}$	BLEU-4 (>30)	**27.61 (35)**	24.70 (28)	24.51 (31)	26.66 (33)	23.39 (29)	22.08 (26)
	METEOR-F (>70)	**41.55 (35)**	38.02 (31)	37.64 (30)	40.20 (33)	36.80 (27)	36.63 (26)
	BERTScore-F1 (>60)	**48.76 (30)**	47.71 (29)	46.83 (25)	47.69 (31)	45.04 (25)	43.29 (25)

Tabelle 50: Effektivität der zuerst auf SQuADv1.1 feingetunten und danach weiter auf Passage-1kTrain-Train (398 Einträge) feingetunten Modelle beim Spoilern der Spoilerklasse Passage mit der Anzahl an Trainingsschritten, die zum besten Ergebnis führen. Gemessen auf Passage-1kTrain-Valid (102 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem bei Metik auch in Klammern angegebenen Schwellwert. In der Fußnote ist die Version des Modells angegeben. Hyperparameter: Batchgröße 16, Sequenzlänge 384, Lernrate 3e-5, Rest Standardwerte im `run_qa.py` Skript von Huggingface. 15 Epochen entsprechen 1,200 Schritten. Bester Wert für jedes Modell hervorgehoben.

Modell	Metrik	Schritt700	Schritt800	Schritt900	Schritt1k	Schritt1,1k	Schrittende
$\text{ALBERT}_{\text{v2-base}}$	BLEU-4 (>30)	19.55 (25)	19.90 (25)	19.06 (24)	19.06 (24)	18.19 (23)	18.19 (23)
	METEOR-F (>70)	34.15 (24)	33.90 (23)	33.05 (23)	33.03 (23)	32.13 (22)	32.17 (22)
	BERTScore-F1 (>60)	42.75 (20)	42.44 (20)	42.03 (19)	42.07 (19)	41.43 (18)	41.44 (25)
$\text{BERT}_{\text{base-uncased}}$	BLEU-4 (>30)	15.26 (20)	17.17 (20)	15.51 (20)	16.46 (20)	15.76 (20)	15.76 (20)
	METEOR-F (>70)	27.59 (15)	31.05 (18)	29.00 (16)	30.19 (17)	29.30 (17)	29.10 (17)
	BERTScore-F1 (>60)	38.08 (14)	39.85 (18)	39.14 (15)	39.72 (17)	39.24 (17)	39.26 (21)
$\text{BERT}_{\text{base-cased}}$	BLEU-4 (>30)	**17.65 (22)**	15.71 (20)	15.71 (20)	15.88 (20)	16.27 (21)	16.67 (22)
	METEOR-F (>70)	**28.09 (20)**	27.40 (18)	27.37 (18)	28.26 (18)	28.23 (18)	29.33 (19)
	BERTScore-F1 (>60)	**40.30 (16)**	39.36 (15)	39.19 (15)	39.83 (15)	40.18 (15)	40.60 (18)
$\text{ELECTRA}_{\text{discriminator-base}}$	BLEU-4 (>30)	22.32 (27)	22.60 (26)	23.15 (28)	22.23 (27)	22.37 (27)	21.26 (26)
	METEOR-F (>70)	35.32 (23)	35.75 (23)	36.53 (25)	36.65 (23)	36.71 (23)	35.12 (22)
	BERTScore-F1 (>60)	42.26 (22)	43.16 (24)	43.47 (24)	41.76 (21)	41.68 (22)	41.69 (24)
$\text{FunnelTransformer}_{\text{small}}$	BLEU-4 (>30)	21.98 (30)	23.14 (32)	24.15 (32)	22.04 (29)	22.92 (32)	22.04 (31)
	METEOR-F (>70)	37.25 (25)	37.34 (27)	37.54 (26)	36.57 (24)	38.18 (27)	37.43 (26)
	BERTScore-F1 (>60)	43.08 (22)	44.54 (23)	45.16 (24)	43.55 (22)	44.34 (23)	43.86 (27)
$\text{MPNet}_{\text{base}}$	BLEU-4 (>30)	23.76 (28)	24.89 (30)	25.56 (31)	26.86 (31)	27.25 (31)	25.98 (30)
	METEOR-F (>70)	36.70 (28)	36.48 (26)	39.00 (29)	39.38 (30)	39.85 (30)	38.25 (28)
	BERTScore-F1 (>60)	43.10 (25)	41.87 (27)	42.53 (26)	44.44 (28)	44.14 (31)	42.76 (28)
$\text{RoBERTa}_{\text{base}}$	BLEU-4 (>30)	19.56 (24)	23.37 (28)	23.23 (28)	24.75 (30)	24.01 (29)	24.01 (29)
	METEOR-F (>70)	34.85 (22)	37.28 (28)	37.35 (26)	38.94 (26)	38.25 (26)	38.30 (26)
	BERTScore-F1 (>60)	39.77 (19)	44.16 (25)	43.26 (24)	43.22 (25)	43.17 (24)	43.01 (26)

Tabelle 51: Effektivität der zuerst auf SQuADv1.1 feingetunten und danach weiter auf Passage-1kTrain-Train (398 Einträge) feingetunten Modelle beim Spoilern der Spoilerklasse Passage mit der Anzahl an Trainingsschritten, die zum besten Ergebnis führen. Gemessen auf Passage-1kTrain-Valid (102 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem bei Metik auch in Klammern angegebenen Schwellwert. In der Fußnote ist die Version des Modells angegeben. Hyperparameter: Batchgröße 8, Sequenzlänge 384, Lernrate 3e-5, Rest Standardwerte im `run_qa.py` Skript von Huggingface. 15 Epochen entsprechen 2,400 Schritten. Bester Wert für jedes Modell hervorgehoben.

Modell	Metrik	Schritt200	Schritt400	Schritt600	Schritt800	Schritt1k	Schritt1,2k
$\text{DeBERTa}_{\text{large}}$	BLEU-4 (>30)	**29.52 (38)**	27.39 (37)	26.35 (32)	26.06 (33)	28.17 (36)	25.29 (32)
	METEOR-F (>70)	**43.72 (36)**	43.94 (35)	38.41 (31)	39.12 (31)	41.88 (34)	39.33 (31)
	BERTScore-F1 (>60)	**49.63 (37)**	48.07 (36)	45.83 (34)	46.32 (34)	48.09 (35)	45.75 (32)
$\text{RoBERTa}_{\text{large}}$	BLEU-4 (>30)	25.96 (32)	23.85 (31)	25.16 (31)	28.63 (34)	24.95 (30)	**29.58 (35)**
	METEOR-F (>70)	39.74 (34)	39.34 (30)	39.68 (30)	43.54 (34)	39.63 (29)	**43.49 (32)**
	BERTScore-F1 (>60)	47.26 (35)	46.00 (32)	45.71 (31)	48.02 (33)	44.77 (29)	**48.65 (32)**
$\text{BigBird}_{\text{RoBERTa-large-384}}$	BLEU-4 (>30)	20.10 (26)	22.49 (29)	23.66 (32)	22.20 (29)	22.05 (30)	**23.89 (30)**
	METEOR-F (>70)	32.41 (24)	34.12 (26)	36.53 (29)	37.29 (25)	33.92 (27)	**36.20 (28)**
	BERTScore-F1 (>60)	40.24 (25)	41.56 (26)	41.59 (28)	43.30 (29)	41.80 (28)	**44.55 (27)**

Modell	Metrik	Schritt1,4k	Schritt1,6k	Schritt1,8k	Schritt2,0k	Schritt2,2k	Schrittende
$\text{DeBERTa}_{\text{large}}$	BLEU-4 (>30)	26.45 (34)	25.63 (33)	26.80 (34)	28.16 (35)	28.90 (36)	28.90 (36)
	METEOR-F (>70)	39.26 (32)	38.15 (31)	40.39 (33)	40.57 (34)	41.53 (35)	41.52 (35)
	BERTScore-F1 (>60)	46.25 (34)	44.57 (32)	45.79 (34)	46.64 (36)	47.29 (36)	47.28 (36)
$\text{RoBERTa}_{\text{large}}$	BLEU-4 (>30)	23.99 (28)	24.83 (32)	24.24 (29)	26.47 (32)	24.77 (30)	25.40 (31)
	METEOR-F (>70)	37.70 (25)	37.57 (28)	39.55 (25)	40.31 (28)	39.71 (28)	40.24 (29)
	BERTScore-F1 (>60)	44.89 (25)	45.38 (26)	43.85 (26)	45.42 (28)	44.78 (27)	45.08 (28)
$\text{BigBird}_{\text{RoBERTa-large}}$	BLEU-4 (>30)	22.14 (28)	21.07 (27)	22.84 (29)	20.46 (26)	21.99 (28)	21.99 (28)
	METEOR-F (>70)	34.50 (25)	32.12 (22)	33.18 (23)	33.91 (22)	35.22 (24)	35.11 (24)
	BERTScore-F1 (>60)	42.51 (26)	41.49 (22)	42.70 (24)	41.18 (22)	42.01 (25)	42.20 (25)

Tabelle 52: Effektivität der zuerst auf SQuADv1.1 feingetunten und danach weiter auf Passage-1kTrain-Train (398 Einträge) feingetunten BigBird-Modelle beim Spoilern der Spoilerklasse Passage mit der Anzahl an Trainingsschritten, die zum besten Ergebnis führen. Gemessen auf Passage-1kTrain-Valid (102 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem bei Metik auch in Klammern angegebenen Schwellwert. In der Fußnote ist die Version des Modells und die genutzte Sequenzlänge angegeben. Hyperparameter: Batchgröße 16, Sequenzlänge 384, Lernrate 3e-5, Rest Standardwerte im `run_qa.py` Skript von Huggingface. 15 Epochen entsprechen 1,200 Schritten (mit steigender Sequenzlänge werden es weniger (–)). Bester Wert für jedes Modell hervorgehoben.

Modell	Metrik	Schritt100	Schritt200	Schritt300	Schritt400	Schritt500	Schritt600
$\text{BigBird}_{\text{RoBERTa-large-384}}$	BLEU-4 (>30)	20.10 (26)	22.49 (29)	23.66 (32)	22.20 (29)	22.05 (30)	**23.89 (30)**
	METEOR-F (>70)	32.41 (24)	34.12 (26)	36.53 (29)	37.29 (25)	33.92 (27)	**36.20 (28)**
	BERTScore-F1 (>60)	40.24 (25)	41.56 (26)	41.59 (28)	43.30 (29)	41.80 (28)	**44.55 (27)**
$\text{BigBird}_{\text{RoBERTa-large-512}}$	BLEU-4 (>30)	20.62 (28)	20.94 (27)	19.11 (25)	**27.09 (34)**	20.63 (27)	20.69 (28)
	METEOR-F (>70)	33.68 (23)	35.79 (23)	33.88 (23)	**39.07 (29)**	35.01 (24)	34.86 (23)
	BERTScore-F1 (>60)	42.33 (24)	43.01 (25)	42.27 (27)	**45.64 (31)**	40.91 (24)	40.25 (23)
$\text{BigBird}_{\text{RoBERTa-large-768}}$	BLEU-4 (>30)	19.76 (25)	21.78 (30)	21.05 (28)	22.37 (31)	21.81 (30)	**23.13 (32)**
	METEOR-F (>70)	35.57 (25)	37.87 (24)	36.58 (25)	36.20 (24)	34.67 (24)	**38.24 (27)**
	BERTScore-F1 (>60)	38.72 (24)	39.32 (24)	37.48 (23)	41.69 (23)	40.96 (23)	**40.84 (26)**
$\text{BigBird}_{\text{RoBERTa-large-1024}}$	BLEU-4 (>30)	22.81 (29)	23.63 (31)	21.83 (29)	**24.41 (33)**	22.28 (29)	22.27 (29)
	METEOR-F (>70)	35.04 (28)	39.32 (28)	36.37 (28)	**39.96 (29)**	37.41 (26)	37.23 (26)
	BERTScore-F1 (>60)	44.16 (27)	43.44 (28)	41.72 (27)	**42.41 (28)**	41.72 (25)	41.27 (25)
$\text{BigBird}_{\text{RoBERTa-large-1280}}$	BLEU-4 (>30)	**25.86 (32)**	24.08 (32)	21.48 (26)	20.66 (26)	23.11 (29)	–
	METEOR-F (>70)	**39.31 (33)**	39.42 (29)	34.64 (24)	33.93 (24)	36.21 (26)	–
	BERTScore-F1 (>60)	**46.39 (34)**	43.66 (28)	41.94 (24)	41.28 (23)	42.84 (25)	–

Tabelle 53: Effektivität der zuerst auf SQuADv1.1 feingetunten und danach weiter auf Passage-1kTrain-Train (398 Einträge) feingetunten BigBird-Modelle beim Spoilern der Spoilerklasse Passage mit der Anzahl an Trainingsschritten, die zum besten Ergebnis führen. Gemessen auf Passage-1kTrain-Valid (102 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem bei Metik auch in Klammern angegebenen Schwellwert. In der Fußnote ist die Version des Modells und die genutzte Sequenzlänge angegeben. Hyperparameter: Batchgröße 16, Sequenzlänge 384, Lernrate 3e-5, Rest Standardwerte im `run_qa.py` Skript von Huggingface. 15 Epochen entsprechen 1,200 Schritten (mit steigender Sequenzlänge werden es weniger (–)). Bester Wert für jedes Modell hervorgehoben.

Modell	Metrik	Schritt700	Schritt800	Schritt900	Schritt1k	Schritt1,1k	Schrittende
$\text{BigBird}_{\text{RoBERTa-large-384}}$	BLEU-4 (>30)	22.14 (28)	21.07 (27)	22.84 (29)	20.46 (26)	21.99 (28)	21.99 (28)
	METEOR-F (>70)	34.50 (25)	32.12 (22)	33.18 (23)	33.91 (22)	35.22 (24)	35.11 (24)
	BERTScore-F1 (>60)	42.51 (26)	41.49 (22)	42.70 (24)	41.18 (22)	42.01 (25)	42.20 (25)
$\text{BigBird}_{\text{RoBERTa-large-512}}$	BLEU-4 (>30)	18.78 (26)	17.41 (23)	17.87 (23)	–	–	–
	METEOR-F (>70)	33.16 (21)	31.95 (21)	32.42 (21)	–	–	–
	BERTScore-F1 (>60)	38.69 (21)	38.66 (21)	38.75 (21)	–	–	–
$\text{BigBird}_{\text{RoBERTa-large-768}}$	BLEU-4 (>30)	22.91 (32)	–	–	–	–	–
	METEOR-F (>70)	37.24 (26)	–	–	–	–	–
	BERTScore-F1 (>60)	40.41 (25)	–	–	–	–	–
$\text{BigBird}_{\text{RoBERTa-large-1024}}$	BLEU-4 (>30)	–	–	–	–	–	–
	METEOR-F (>70)	–	–	–	–	–	–
	BERTScore-F1 (>60)	–	–	–	–	–	–
$\text{BigBird}_{\text{RoBERTa-large-1280}}$	BLEU-4 (>30)	–	–	–	–	–	–
	METEOR-F (>70)	–	–	–	–	–	–
	BERTScore-F1 (>60)	–	–	–	–	–	–

Tabelle 54: Effektivität der auf SQuADv1.1 feingetunten Modelle beim Spoilern der Spoilerklasse Phrase abhängig von der Anzahl an Trainingsschritten. Gemessen auf Phrase-1kTrain-Valid (80 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem bei Metik auch in Klammern angegebenen Schwellwert. In der Fußnote ist die Version des Modells angegeben. Hyperparameter: Batchgröße 16, Sequenzlänge 384, Lernrate 3e-5, Rest Standardwerte im `run_qa.py` Skript von Huggingface. 3 Epochen entsprechen 16,500 Schritten.

Modell	Metrik	Schritt6k	Schritt7k	Schritt8k	Schritt9k	Schritt10k	Schritt11k
$\text{ALBERT}_{\text{v2-base}}$	BLEU-4 (>50)	9.23 (6)	12.99 (9)	**18.48 (14)**	8.62 (5)	13.7 (10)	11.07 (8)
	METEOR-F (>70)	6.15 (6)	7.02 (9)	**8.71 (14)**	5.14 (5)	9.10 (10)	8.93 (9)
	BERTScore-F1 (>80)	31.60 (6)	34.60 (9)	**39.70 (13)**	29.9 (6)	37.66 (9)	35.37 (8)
$\text{BERT}_{\text{base-uncased}}$	BLEU-4 (>50)	18.36 (15)	17.80 (15)	**19.26 (16)**	21.94 (18)	20.43 (16)	16.78 (13)
	METEOR-F (>70)	9.72 (14)	9.51 (13)	11.78 (13)	11.02 (16)	10.86 (15)	10.04 (11)
	BERTScore-F1 (>80)	41.23 (14)	41.42 (13)	43.34 (15)	43.70 (17)	42.34 (16)	39.90 (13)
$\text{BERT}_{\text{base-cased}}$	BLEU-4 (>50)	**34.89 (29)**	30.00 (25)	29.17 (23)	28.82 (24)	31.31 (25)	22.83 (18)
	METEOR-F (>70)	**23.92 (26)**	16.90 (22)	19.60 (23)	16.78 (23)	21.45 (25)	15.05 (17)
	BERTScore-F1 (>80)	**53.86 (25)**	50.83 (21)	50.48 (22)	48.37 (21)	51.37 (24)	45.06 (17)
$\text{ELECTRA}_{\text{discriminator-base}}$	BLEU-4 (>50)	22.31 (18)	26.60 (21)	25.42 (20)	22.23 (17)	23.51 (18)	21.39 (17)
	METEOR-F (>70)	15.21 (19)	17.19 (20)	19.00 (20)	15.04 (17)	17.02 (17)	13.92 (16)
	BERTScore-F1 (>80)	43.97 (15)	47.19 (18)	47.45 (18)	45.38 (16)	45.72 (17)	43.17 (15)
$\text{FunnelTransformer}_{\text{small}}$	BLEU-4 (>50)	25.38 (19)	18.88 (14)	24.28 (18)	19.09 (14)	**28.77 (24)**	23.51 (17)
	METEOR-F (>70)	13.39 (19)	11.06 (14)	13.46 (19)	11.43 (13)	**16.39 (22)**	10.64 (18)
	BERTScore-F1 (>80)	43.03 (18)	37.64 (12)	43.80 (18)	41.22 (15)	**47.70 (20)**	40.68 (18)
$\text{MPNet}_{\text{base}}$	BLEU-4 (>50)	38.63 (32)	42.98 (35)	41.93 (33)	**50.64 (40)**	50.69 (41)	41.44 (34)
	METEOR-F (>70)	21.57 (28)	24.55 (33)	21.27 (32)	**31.85 (38)**	35.28 (37)	22.21 (31)
	BERTScore-F1 (>80)	56.77 (27)	60.28 (33)	57.35 (32)	**65.06 (38)**	65.24 (37)	56.90 (31)
$\text{RoBERTa}_{\text{base}}$	BLEU-4 (>50)	49.54 (41)	43.24 (36)	47.98 (38)	46.43 (36)	50.85 (41)	46.87 (38)
	METEOR-F (>70)	28.10 (36)	23.76 (32)	26.42 (37)	23.68 (35)	29.37 (38)	24.65 (36)
	BERTScore-F1 (>80)	61.51 (33)	57.81 (30)	61.42 (35)	58.67 (31)	62.74 (33)	59.58 (32)

Tabelle 55: Effektivität der auf SQuADv1.1 feingetunten Modelle beim Spoilern der Spoilerklasse Phrase abhängig von der Anzahl an Trainingsschritten. Gemessen auf Phrase-1kTrain-Valid (80 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem bei Metik auch in Klammern angegebenen Schwellwert. In der Fußnote ist die Version des Modells angegeben. Hyperparameter: Batchgröße 16, Sequenzlänge 384, Lernrate 3e-5, Rest Standardwerte im `run_qa.py` Skript von Huggingface. 3 Epochen entsprechen 16,500 Schritten. Bester Wert für jedes Modell hervorgehoben.

Modell	Metrik	Schritt12k	Schritt13k	Schritt14k	Schritt15k	Schritt16k	Schrittende
$\text{ALBERT}_{\text{v2-base}}$	BLEU-4 (>50)	11.43 (8)	11.64 (8)	12.73 (10)	11.17 (8)	13.05 (10)	13.21 (10)
	METEOR-F (>70)	7.61 (8)	7.62 (8)	8.80 (10)	8.66 (8)	9.30 (10)	9.41 (10)
	BERTScore-F1 (>80)	35.02 (8)	37.03 (9)	37.09 (9)	34.79 (8)	37.15 (9)	37.16 (9)
$\text{BERT}_{\text{base-uncased}}$	BLEU-4 (>50)	16.62 (14)	19.66 (16)	18.48 (14)	23.68 (19)	**24.81 (21)**	24.81 (21)
	METEOR-F (>70)	9.56 (11)	10.40 (14)	10.71 (14)	12.50 (18)	**12.77 (18)**	13.04 (18)
	BERTScore-F1 (>80)	40.12 (13)	42.05 (16)	41.77 (15)	46.07 (19)	**45.93 (19)**	46.01 (19)
$\text{BERT}_{\text{base-cased}}$	BLEU-4 (>50)	28.06 (23)	26.89 (21)	25.78 (21)	26.20 (21)	29.12 (23)	28.46 (23)
	METEOR-F (>70)	19.04 (21)	18.12 (20)	14.84 (19)	16.89 (19)	17.07 (22)	16.99 (21)
	BERTScore-F1 (>80)	49.51 (20)	48.06 (20)	47.52 (19)	48.36 (19)	50.02 (22)	49.86 (21)
$\text{ELECTRA}_{\text{discriminator-base}}$	BLEU-4 (>50)	28.66 (24)	30.56 (24)	31.91 (25)	30.27 (24)	**34.39 (28)**	30.64 (25)
	METEOR-F (>70)	16.81 (22)	17.16 (23)	18.73 (23)	18.67 (22)	**19.30 (25)**	17.65 (22)
	BERTScore-F1 (>80)	47.68 (18)	49.61 (22)	48.74 (21)	48.18 (21)	**49.78 (23)**	48.32 (21)
$\text{FunnelTransformer}_{\text{small}}$	BLEU-4 (>50)	22.77 (17)	25.53 (20)	27.22 (22)	26.80 (22)	27.29 (22)	26.29 (21)
	METEOR-F (>70)	12.60 (18)	13.74 (18)	14.24 (20)	14.63 (21)	14.22 (21)	14.29 (22)
	BERTScore-F1 (>80)	42.47 (17)	44.96 (18)	46.97 (19)	44.90 (17)	45.31 (18)	45.29 (18)
$\text{MPNet}_{\text{base}}$	BLEU-4 (>50)	44.60 (36)	45.08 (36)	43.25 (36)	44.53 (36)	44.54 (37)	42.42 (35)
	METEOR-F (>70)	21.47 (32)	23.82 (32)	21.65 (32)	23.88 (33)	23.38 (33)	22.85 (31)
	BERTScore-F1 (>80)	57.11 (30)	57.81 (31)	56.35 (30)	58.99 (32)	59.14 (32)	57.04 (30)
$\text{RoBERTa}_{\text{base}}$	BLEU-4 (>50)	**52.06 (42)**	51.53 (42)	46.38 (37)	43.88 (35)	45.87 (36)	45.20 (36)
	METEOR-F (>70)	**32.62 (40)**	28.19 (39)	25.52 (36)	23.82 (32)	25.44 (34)	25.10 (34)
	BERTScore-F1 (>80)	**64.61 (34)**	62.75 (34)	59.97 (32)	57.65 (29)	59.35 (30)	59.31 (31)

Tabelle 56: Effektivität der auf SQuADv1.1 feingetunten Modelle beim Spoilern der Spoilerklasse Phrase abhängig von der Anzahl an Trainingsschritten. Gemessen auf Phrase-1kTrain-Valid (80 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem bei Metik auch in Klammern angegebenen Schwellwert. In der Fußnote ist die Version des Modells angegeben. Hyperparameter: Batchgröße 8, Sequenzlänge 384, Lernrate 3e-5, Rest Standardwerte im `run_qa.py` Skript von Huggingface. 3 Epochen entsprechen 33,000 Schritten. Bester Wert für jedes Modell hervorgehoben.

Modell	Metrik	Schritt12k	Schritt14k	Schritt16k	Schritt18k	Schritt20k	Schritt22k
$\text{BigBird}_{\text{RoBERTa-large}}$	BLEU-4 (>50)	27.77 (22)	27.12 (20)	18.77 (13)	20.79 (15)	27.49 (21)	27.04 (20)
	METEOR-F (>70)	13.61 (21)	14.19 (20)	10.38 (13)	10.48 (15)	14.79 (21)	12.78 (19)
	BERTScore-F1 (>80)	45.48 (18)	44.49 (19)	37.45 (12)	39.90 (14)	45.78 (19)	44.14 (17)
$\text{DeBERTa}_{\text{large}}$	BLEU-4 (>50)	32.72 (24)	27.54 (22)	33.57 (25)	27.02 (22)	39.92 (31)	27.59 (21)
	METEOR-F (>70)	20.01 (24)	12.07 (20)	16.37 (21)	13.59 (20)	21.64 (29)	13.65 (19)
	BERTScore-F1 (>80)	50.24 (23)	43.84 (19)	46.50 (19)	43.34 (17)	53.83 (27)	44.40 (18)
$\text{RoBERTa}_{\text{large}}$	BLEU-4 (>50)	43.91 (36)	48.25 (38)	47.09 (37)	**48.63 (37)**	37.11 (29)	45.03 (35)
	METEOR-F (>70)	26.31 (33)	28.79 (37)	31.99 (35)	**31.11 (36)**	18.06 (27)	26.05 (34)
	BERTScore-F1 (>80)	58.36 (29)	61.48 (32)	61.26 (33)	**60.70 (33)**	51.47 (25)	58.83 (31)

Modell	Metrik	Schritt24k	Schritt26k	Schritt28k	Schritt30k	Schritt32k	Schritt22k
$\text{BigBird}_{\text{RoBERTa-large}}$	BLEU-4 (>50)	28.89 (22)	27.51 (21)	29.40 (22)	29.60 (22)	**33.62 (25)**	33.41 (25)
	METEOR-F (>70)	16.53 (22)	14.05 (21)	14.58 (22)	15.39 (22)	**19.54 (25)**	18.96 (25)
	BERTScore-F1 (>80)	47.77 (21)	46.61 (20)	46.3 (19)	46.97 (20)	**50.65 (22)**	50.35 (22)
$\text{DeBERTa}_{\text{large}}$	BLEU-4 (>50)	24.53 (20)	**39.11 (31)**	35.85 (28)	33.10 (25)	35.45 (27)	36.73 (28)
	METEOR-F (>70)	13.72 (16)	**24.94 (26)**	20.81 (25)	16.86 (22)	19.93 (23)	20.24 (24)
	BERTScore-F1 (>80)	43.32 (13)	**54.43 (24)**	50.60 (22)	47.70 (20)	50.68 (21)	51.18 (22)
$\text{RoBERTa}_{\text{large}}$	BLEU-4 (>50)	54.22 (44)	43.18 (34)	52.09 (41)	49.68 (40)	50.76 (40)	50.20 (40)
	METEOR-F (>70)	29.95 (42)	21.75 (31)	28.61 (40)	25.08 (37)	26.55 (39)	26.18 (39)
	BERTScore-F1 (>80)	64.82 (36)	54.32 (26)	63.18 (37)	60.84 (33)	61.98 (33)	61.72 (33)

Tabelle 57: Effektivität der auf Phrase-1kTrain-Train (320 Einträge) feingetunten Modelle beim Spoilern der Spoilerklasse Phrase abhängig von der Anzahl an Trainingsschritten. Gemessen auf Phrase-1kTrain-Valid (80 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem bei Metik auch in Klammern angegebenen Schwellwert. In der Fußnote ist die Version des Modells angegeben. Hyperparameter: Batchgröße 16, Sequenzlänge 384, Lernrate 3e-5, Rest Standardwerte im `run_qa.py` Skript von Huggingface. 15 Epochen entsprechen 800 Schritten. Bester Wert für jedes Modell hervorgehoben.

Modell	Metrik	Schritt100	Schritt200	Schritt300	Schritt400
$\text{ALBERT}_{\text{v2-base}}$	BLEU-4 (>50)	51.33 (39)	45.91 (38)	54.50 (43)	52.18 (42)
	METEOR-F (>70)	46.46 (39)	43.83 (35)	49.39 (41)	50.68 (39)
	BERTScore-F1 (>80)	67.29 (38)	62.81 (32)	68.25 (40)	67.14 (38)
$\text{BERT}_{\text{base-uncased}}$	BLEU-4 (>50)	33.50 (24)	37.00 (28)	**41.85 (32)**	38.64 (30)
	METEOR-F (>70)	28.79 (26)	28.94 (27)	**35.39 (33)**	28.32 (28)
	BERTScore-F1 (>80)	53.48 (24)	55.28 (25)	**60.42 (31)**	55.64 (24)
$\text{BERT}_{\text{base-cased}}$	BLEU-4 (>50)	42.59 (33)	42.93 (33)	44.91 (35)	44.25 (35)
	METEOR-F (>70)	36.95 (32)	36.15 (33)	36.70 (34)	36.92 (33)
	BERTScore-F1 (>80)	59.52 (30)	59.13 (29)	60.53 (31)	60.09 (29)
$\text{ELECTRA}_{\text{discriminator-base}}$	BLEU-4 (>50)	36.88 (27)	43.72 (33)	41.13 (31)	39.61 (29)
	METEOR-F (>70)	34.86 (29)	34.92 (35)	33.87 (33)	28.99 (31)
	BERTScore-F1 (>80)	57.49 (28)	61.20 (30)	59.94 (31)	57.45 (28)
$\text{FunnelTransformer}_{\text{small}}$	BLEU-4 (>50)	48.73 (40)	51.35 (42)	42.59 (36)	**50.67 (41)**
	METEOR-F (>70)	48.13 (40)	50.46 (38)	44.05 (32)	**51.53 (39)**
	BERTScore-F1 (>80)	65.09 (35)	64.49 (33)	59.74 (27)	**66.01 (36)**
$\text{MPNet}_{\text{base}}$	BLEU-4 (>50)	43.94 (35)	54.51 (43)	53.61 (43)	60.03 (48)
	METEOR-F (>70)	41.41 (32)	52.76 (42)	51.10 (42)	51.15 (48)
	BERTScore-F1 (>80)	59.89 (28)	67.22 (38)	68.01 (39)	71.73 (44)
$\text{RoBERTa}_{\text{base}}$	BLEU-4 (>50)	51.03 (41)	54.72 (44)	53.10 (42)	56.34 (45)
	METEOR-F (>70)	44.06 (40)	50.91 (42)	42.58 (40)	53.06 (42)
	BERTScore-F1 (>80)	65.47 (35)	68.08 (38)	66.16 (37)	69.08 (39)

Tabelle 58: Effektivität der auf Phrase-1kTrain-Train (320 Einträge) feingetunten Modelle beim Spoilern der Spoilerklasse Phrase abhängig von der Anzahl an Trainingsschritten. Gemessen auf Phrase-1kTrain-Valid (80 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem bei Metik auch in Klammern angegebenen Schwellwert. In der Fußnote ist die Version des Modells angegeben. Hyperparameter: Batchgröße 16, Sequenzlänge 384, Lernrate 3e-5, Rest Standardwerte im `run_qa.py` Skript von Huggingface. 15 Epochen entsprechen 800 Schritten. Bester Wert für jedes Modell hervorgehoben.

Modell	Metrik	Schritt500	Schritt600	Schritt700	Schritt800	Schrittende
$\text{ALBERT}_{\text{v2-base}}$	BLEU-4 (>50)	54.69 (44)	**56.36 (45)**	56.36 (45)	55.11 (44)	
	METEOR-F (>70)	49.35 (42)	**54.25 (43)**	54.25 (43)	53.31 (42)	
	BERTScore-F1 (>80)	68.16 (41)	**69.83 (42)**	69.83 (42)	69.63 (42)	
$\text{BERT}_{\text{base-uncased}}$	BLEU-4 (>50)	33.26 (25)	37.52 (29)	38.69 (30)	39.31 (30)	
	METEOR-F (>70)	21.09 (25)	25.45 (28)	29.23 (28)	28.60 (29)	
	BERTScore-F1 (>80)	51.95 (23)	55.43 (26)	55.75 (26)	56.00 (27)	
$\text{BERT}_{\text{base-cased}}$	BLEU-4 (>50)	**45.40 (36)**	42.84 (33)	43.55 (33)	44.53 (34)	
	METEOR-F (>70)	**35.81 (36)**	32.51 (34)	32.71 (34)	34.12 (35)	
	BERTScore-F1 (>80)	**60.76 (31)**	59.39 (30)	59.70 (31)	60.27 (31)	
$\text{ELECTRA}_{\text{discriminator-base}}$	BLEU-4 (>50)	**44.72 (34)**	42.86 (32)	40.40 (30)	39.94 (30)	
	METEOR-F (>70)	**37.45 (35)**	34.33 (34)	31.93 (32)	31.56 (32)	
	BERTScore-F1 (>80)	**61.13 (33)**	59.92 (31)	58.30 (29)	58.10 (29)	
$\text{FunnelTransformer}_{\text{small}}$	BLEU-4 (>50)	49.00 (39)	48.44 (39)	49.69 (40)	49.69 (40)	
	METEOR-F (>70)	50.71 (38)	50.14 (37)	50.39 (37)	50.39 (37)	
	BERTScore-F1 (>80)	65.45 (35)	65.21 (35)	65.39 (35)	65.39 (35)	
$\text{MPNet}_{\text{base}}$	BLEU-4 (>50)	**59.89 (48)**	57.08 (46)	56.86 (45)	56.86 (45)	
	METEOR-F (>70)	**54.16 (47)**	50.01 (45)	47.48 (45)	50.84 (45)	
	BERTScore-F1 (>80)	**71.88 (43)**	70.15 (41)	70.16 (42)	70.36 (42)	
$\text{RoBERTa}_{\text{base}}$	BLEU-4 (>50)	**58.82 (47)**	53.64 (43)	54.23 (43)	54.00 (42)	
	METEOR-F (>70)	**46.60 (46)**	42.79 (41)	42.19 (42)	35.45 (42)	
	BERTScore-F1 (>80)	**69.91 (42)**	66.02 (38)	66.72 (39)	65.48 (39)	

Tabelle 59: Effektivität der zuerst auf SQuADv1.1 feingetunten und danach weiter auf Phrase-1kTrain-Train (320 Einträge) feingetunten Modelle beim Spoilern der Spoilerklasse Phrase mit der Anzahl an Trainingsschritten, die zum besten Ergebnis führen. Gemessen auf Phrase-1kTrain-Valid (80 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem bei Metik auch in Klammern angegebenen Schwellwert. In der Fußnote ist die Version des Modells angegeben. Hyperparameter: Batchgröße 16, Sequenzlänge 384, Lernrate 3e-5, Rest Standardwerte im `run_qa.py` Skript von Huggingface. 15 Epochen entsprechen 800 Schritten. Bester Wert für jedes Modell hervorgehoben.

Modell	Metrik	Schritt100	Schritt200	Schritt300	Schritt400	Schritt500
$\text{ALBERT}_{\text{v2-base}}$	BLEU-4 (>50)	60.79 (48)	62.40 (48)	**63.82 (50)**	61.30 (47)	
	METEOR-F (>70)	59.55 (48)	55.07 (50)	**55.97 (49)**	60.41 (49)	
	BERTScore-F1 (>80)	73.98 (46)	74.91 (48)	**74.07 (46)**	73.94 (46)	
$\text{BERT}_{\text{base-uncased}}$	BLEU-4 (>50)	**62.36 (49)**	56.72 (44)	56.56 (44)	61.33 (48)	
	METEOR-F (>70)	**53.17 (47)**	52.18 (42)	46.73 (42)	57.72 (47)	
	BERTScore-F1 (>80)	**75.87 (47)**	72.95 (44)	71.04 (43)	74.68 (46)	
$\text{BERT}_{\text{base-cased}}$	BLEU-4 (>50)	56.05 (43)	57.15 (44)	**60.27 (49)**	58.92 (47)	
	METEOR-F (>70)	50.02 (43)	50.99 (44)	**58.87 (47)**	51.97 (45)	
	BERTScore-F1 (>80)	71.97 (43)	70.81 (42)	**73.55 (44)**	70.84 (42)	
$\text{ELECTRA}_{\text{discriminator-base}}$	BLEU-4 (>50)	64.15 (52)	**69.10 (55)**	67.85 (54)	68.94 (54)	
	METEOR-F (>70)	61.86 (50)	**65.97 (53)**	65.28 (52)	65.53 (53)	
	BERTScore-F1 (>80)	76.85 (47)	**79.26 (51)**	78.98 (50)	79.61 (52)	
$\text{FunnelTransformer}_{\text{small}}$	BLEU-4 (>50)	63.44 (51)	66.34 (52)	64.30 (50)	62.50 (50)	
	METEOR-F (>70)	62.99 (50)	66.99 (51)	62.08 (49)	60.85 (49)	
	BERTScore-F1 (>80)	74.88 (46)	78.23 (49)	76.10 (48)	74.33 (45)	
$\text{MPNet}_{\text{base}}$	BLEU-4 (>50)	69.89 (56)	67.38 (52)	68.06 (54)	71.92 (57)	
	METEOR-F (>70)	68.76 (54)	65.58 (53)	59.85 (54)	67.50 (55)	
	BERTScore-F1 (>80)	80.03 (53)	75.86 (49)	78.67 (52)	79.36 (53)	
$\text{RoBERTa}_{\text{base}}$	BLEU-4 (>50)	70.82 (57)	**73.02 (59)**	69.12 (55)	72.10 (57)	
	METEOR-F (>70)	68.88 (54)	**65.56 (57)**	58.17 (53)	67.40 (56)	
	BERTScore-F1 (>80)	78.95 (52)	**80.39 (54)**	77.54 (50)	80.48 (53)	

Tabelle 60: Effektivität der zuerst auf SQuADv1.1 feingetunten und danach weiter auf Phrase-1kTrain-Train (320 Einträge) feingetunten Modelle beim Spoilern der Spoilerklasse Phrase mit der Anzahl an Trainingsschritten, die zum besten Ergebnis führen. Gemessen auf Phrase-1kTrain-Valid (80 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem bei Metik auch in Klammern angegebenen Schwellwert. In der Fußnote ist die Version des Modells angegeben. Hyperparameter: Batchgröße 16, Sequenzlänge 384, Lernrate 3e-5, Rest Standardwerte im `run_qa.py` Skript von Huggingface. 15 Epochen entsprechen 800 Schritten. Bester Wert für jedes Modell hervorgehoben.

Modell	Metrik	Schritt600	Schritt700	Schrittende	
$\text{ALBERT}_{\text{v2-base}}$	BLEU-4 (>50)	60.84 (47)	59.59 (46)	59.59 (46)	59.59 (46)
	METEOR-F (>70)	59.18 (49)	56.50 (48)	56.26 (48)	56.26 (48)
	BERTScore-F1 (>80)	73.25 (46)	71.97 (45)	72.06 (45)	72.06 (45)
$\text{BERT}_{\text{base-uncased}}$	BLEU-4 (>50)	60.92 (48)	57.17 (45)	56.55 (45)	57.17 (45)
	METEOR-F (>70)	53.95 (47)	50.96 (44)	50.64 (43)	50.96 (44)
	BERTScore-F1 (>80)	73.67 (45)	71.23 (42)	70.82 (41)	71.23 (42)
$\text{BERT}_{\text{base-cased}}$	BLEU-4 (>50)	56.42 (45)	55.17 (44)	55.17 (44)	55.17 (44)
	METEOR-F (>70)	50.76 (44)	49.20 (43)	50.42 (43)	50.42 (43)
	BERTScore-F1 (>80)	69.54 (40)	68.68 (39)	68.89 (39)	68.89 (39)
$\text{ELECTRA}_{\text{discriminator-base}}$	BLEU-4 (>50)	66.60 (53)	65.98 (52)	65.35 (52)	65.98 (52)
	METEOR-F (>70)	63.79 (51)	61.35 (51)	62.12 (50)	61.62 (51)
	BERTScore-F1 (>80)	77.63 (50)	77.61 (50)	77.32 (49)	77.93 (50)
$\text{FunnelTransformer}_{\text{small}}$	BLEU-4 (>50)	62.13 (49)	**68.31 (54)**	64.88 (51)	65.81 (52)
	METEOR-F (>70)	57.85 (48)	**63.89 (53)**	59.93 (50)	61.54 (51)
	BERTScore-F1 (>80)	73.75 (45)	**78.78 (51)**	76.33 (48)	77.26 (49)
$\text{MPNet}_{\text{base}}$	BLEU-4 (>50)	71.50 (57)	70.14 (56)	**72.92 (58)**	69.63 (55)
	METEOR-F (>70)	65.32 (56)	64.48 (55)	**65.90 (57)**	63.48 (55)
	BERTScore-F1 (>80)	79.43 (54)	79.31 (54)	**80.26 (55)**	78.38 (53)
$\text{RoBERTa}_{\text{base}}$	BLEU-4 (>50)	73.35 (58)	70.4 0(56)	71.02 (56)	71.02 (56)
	METEOR-F (>70)	68.19 (57)	65.92 (54)	66.89 (55)	66.89 (55)
	BERTScore-F1 (>80)	81.35 (54)	79.28 (51)	79.80 (52)	79.80 (52)

Tabelle 61: Effektivität der zuerst auf SQuADv1.1 feingetunten und danach weiter auf Phrase-1kTrain-Train (320 Einträge) feingetunten Modelle beim Spoilern der Spoilerklasse Phrase mit der Anzahl an Trainingsschritten, die zum besten Ergebnis führen. Gemessen auf Phrase-1kTrain-Valid (80 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem bei Metik auch in Klammern angegebenen Schwellwert. In der Fußnote ist die Version des Modells angegeben. Hyperparameter: Batchgröße 8, Sequenzlänge 384, Lernrate 3e-5, Rest Standardwerte im `run_qa.py` Skript von Huggingface. 15 Epochen entsprechen 1,600 Schritten. Bester Wert für jedes Modell hervorgehoben.

Modell	Metrik	Schritt200	Schritt400	Schritt600	Schritt800	Schritt1k
$\text{BigBird}_{\text{RoBERTa-large}}$	BLEU-4 (>50)	65.63 (53)	66.67 (54)	**69.21 (55)**	64.35 (51)	
	METEOR-F (>70)	65.46 (50)	58.82 (51)	**64.80 (54)**	55.93 (47)	
	BERTScore-F1 (>80)	74.78 (46)	74.97 (46)	**77.39 (49)**	72.73 (43)	
$\text{DeBERTa}_{\text{large}}$	BLEU-4 (>50)	69.49 (55)	70.18 (55)	**70.19 (57)**	68.37 (54)	
	METEOR-F (>70)	63.86 (54)	60.93 (54)	**65.08 (56)**	65.00 (54)	
	BERTScore-F1 (>80)	77.96 (50)	75.25 (47)	**78.02 (50)**	77.98 (50)	
$\text{RoBERTa}_{\text{large}}$	BLEU-4 (>50)	72.98 (60)	**79.47 (66)**	77.94 (62)	71.56 (57)	
	METEOR-F (>70)	72.26 (55)	**78.61 (61)**	73.37 (58)	71.89 (55)	
	BERTScore-F1 (>80)	80.02 (52)	**84.04 (58)**	82.64 (57)	80.91 (54)	

Modell	Metrik	Schritt1,2k	Schritt1,4k	Schritt1,6k	Schrittende
$\text{BigBird}_{\text{RoBERTa-large}}$	BLEU-4 (>50)	64.54 (53)	64.00 (51)	63.80 (51)	62.06 (49)
	METEOR-F (>70)	59.47 (49)	60.63 (49)	59.96 (48)	56.34 (47)
	BERTScore-F1 (>80)	73.15 (43)	72.95 (43)	73.02 (43)	71.59 (42)
$\text{DeBERTa}_{\text{large}}$	BLEU-4 (>50)	68.39 (53)	67.86 (53)	68.26 (53)	68.26 (53)
	METEOR-F (>70)	67.10 (54)	66.14 (53)	66.29 (53)	64.53 (53)
	BERTScore-F1 (>80)	78.03 (50)	77.71 (50)	77.97 (50)	77.83 (50)
$\text{RoBERTa}_{\text{large}}$	BLEU-4 (>50)	68.08 (54)	70.75 (56)	69.01 (54)	69.01 (54)
	METEOR-F (>70)	60.50 (53)	67.23 (55)	65.24 (53)	65.53 (53)
	BERTScore-F1 (>80)	77.94 (52)	79.47 (53)	78.30 (51)	78.30 (51)

Tabelle 62: Effektivität der zuerst auf SQuADv1.1 feingetunten und danach weiter auf 1kTrain-Train (718 Einträge) feingetunten Modelle beim Spoilern der Spoilerklasse Phrase und Passage mit der Anzahl an Trainingsschritten, die zum besten Ergebnis führen. Gemessen auf 1kTrain-Valid (182 Einträge) und angegeben in Prozent. In Klammern erkennen wir die Anzahl an Spoilern über dem bei Metik auch in Klammern angegebenen Schwellwert. In der Fußnote ist die Version des Modells angegeben. Hyperparameter: Batchgröße 16, Sequenzlänge 384, Lernrate 3e-5, Rest Standardwerte im `run_qa.py` Skript von Huggingface. 8 Epochen entsprechen 1,00 Schritten. Bester Wert für jedes Modell hervorgehoben.

Modell	Metrik	Schritt100	Schritt200	Schritt300	Schritt400	Schritt500
$\text{MPNet}_{\text{base}}$	BLEU-4	40.87 (77)	39.31 (76)	40.71 (80)	**41.22 (82)**	37.93 (75)
	METEOR-F	35.44 (75)	36.80 (76)	38.78 (78)	**39.44 (79)**	38.02 (70)
	BERTScore-F1	57.43 (75)	56.19 (76)	56.33 (78)	**56.20 (76)**	53.95 (73)
$\text{RoBERTa}_{\text{base}}$	BLEU-4	37.42 (72)	38.36 (75)	**39.24 (79)**	37.27 (74)	38.66 (75)
	METEOR-F	35.63 (71)	34.30 (69)	**36.58 (75)**	34.81 (71)	35.41 (72)
	BERTScore-F1	53.50 (69)	53.89 (70)	**54.24 (67)**	53.89 (69)	53.99 (69)

Modell	Metrik	Schritt600	Schritt700	Schritt800	Schritt900	Schrittende
$\text{MPNet}_{\text{base}}$	BLEU-4	36.26 (70)	35.91 (69)	38.49 (76)	39.65 (77)	36.51 (72)
	METEOR-F	31.14 (66)	33.04 (62)	37.33 (71)	36.00 (72)	31.75 (66)
	BERTScore-F1	52.27 (68)	52.21 (63)	53.30 (69)	54.12 (71)	52.44 (66)
$\text{RoBERTa}_{\text{base}}$	BLEU-4	36.99 (73)	39.26 (77)	38.55 (77)	36.75 (70)	37.68 (74)
	METEOR-F	34.55 (68)	38.56 (74)	36.26 (73)	34.99 (68)	36.30 (70)
	BERTScore-F1	52.20 (63)	54.03 (72)	53.52 (70)	52.46 (68)	53.29 (68)

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und die allgemeinen Grundsätze guter wissenschaftlicher Praxis beachtet habe.

Halle (Saale), 8. September 2021

.....
Artur Jurk