

Bauhaus-Universität Weimar
Fakultät Medien
Studiengang Medieninformatik

Automatische Erkennung problematischer Webseiten zur Webanalyse

Bachelorarbeit

Fabienne Hubricht
geb. am: 14.04.1997 in Stollberg

Matrikelnummer 116357

1. Gutachter: Prof. Dr. Benno Stein
2. Gutachter: Prof. Dr. Ing. Volker Rodehorst

Datum der Abgabe: 3. Mai 2019

Erklärung

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

Weimar, 3. Mai 2019

.....
Fabienne Hubricht

Zusammenfassung

Das Internet hat weltweit 4,4 Milliarden aktive Nutzer¹, die sich auf die darin enthaltenen öffentlichen und schnell zugänglichen Informationen verlassen. Allerdings werden Seiten häufig entfernt, zum Beispiel dadurch dass die Domain gelöscht wurde. Zudem werden Webseiten auch regelmäßig verändert, zum Beispiel dadurch dass das Layout geändert wird oder sich der Inhaber der Domain ändert. Wenn Webseiten stark verändert werden, geht der ursprüngliche Seiteninhalt, wenn er nicht archiviert wurde, verloren. Aus diesem Grund werden Webarchive von gemeinnützigen Organisationen, wie zum Beispiel dem Internet Archive, oder Forschungsgruppen angelegt. Somit ist die Seite im Archiv entweder für die Öffentlichkeit zugänglich oder kann für Forschungszwecke verwendet werden, auch wenn die Webseite nicht mehr in ihrer ursprünglichen Form abrufbar ist. Zudem verwenden auch Journalisten oder andere Schreiber, die sich auf Informationen aus Webseiten beziehen wollen, Webarchive. Sie archivieren die Seite um das Archiv zitieren zu können und verwenden dafür den Service des Internet Archive, wo die Archivierung in Auftrag gegeben wird.

Angenommen man legt sich ein Archiv mit mehreren tausend Seiten an und möchte diese später analysieren. Später sind allerdings nur die Daten verfügbar die beim archivieren gespeichert wurden. Das Problem ist, dass dem Crawler keine Information vorliegt ob der eigentliche Webseiteninhalt gecrawlt wird oder nicht. Etwa durch Fehlerseiten, Captchas und PopUps wird der eigentliche Inhalt entweder gar nicht oder nur zum Teil geladen. Um trotzdem an den ursprünglichen Seiteninhalt zu kommen, erfordert es Interaktion mit der Seite, während diese archiviert wird. Da man allerdings so viele Seiten hat, ist es nicht möglich diese problematischen Seiten per Hand zu suchen. Aus diesem Grund ist es notwendig eine automatische Erkennung für die oben genannten Fälle von Seiten zu haben und eine Warnung oder Hinweis während des Crawlens zu erhalten sobald eine dieser Seiten erkannt wird. Damit man dann zum Beispiel weiß, auf welchen Seiten man ein Captcha lösen muss oder eventuell sogar eine Algorithmus hat der automatisch PopUps schließen kann sobald man weiß, dass welche vorhanden sind.

Im Folgenden wurde sich auf die Erkennung von diesen vier Kategorien beschränkt: Fehlerseiten, Captchas, PopUps und pornografische Inhalte. Letzteres stellt zwar keine Beeinträchtigung der Seite da, aber eine Erkennung ist dennoch hilfreich da pornografische Inhalte oft unerwünscht sind. Diese Kategorien wurden gewählt, weil sie häufig auftreten, einfach zu erkennen sind und

¹<https://www.statista.com/statistics/617136/digital-population-worldwide/>
(abgerufen am 12.03.2019)

sich leicht annotieren lassen. Für die automatische Erkennung wurden Merkmale aus dem Quellcode, aus dem Archivierungsformat .warc, welches mehrere digitale Quellen mit einander kombiniert und in einer Datei zusammenfügt, und den Screenshots von den Webseiten extrahiert, welche für diese Seiten charakteristisch sind. Mittels maschinellem Lernen und dem Framework Weka wurde anschließend analysiert welche Merkmale sich zur Erkennung besonders gut eignen. Die verwendeten Merkmale sind die Helligkeit und der Rot-, Grün- und Blauanteil der Screenshots der Webseiten, HTML Attribute wie id, class und source sowie der title Tag, der Webseiteninhalt, die Wortanzahl und Bilderkennung von Captchas.

Inhaltsverzeichnis

1	Einleitung	1
2	Verwandte Arbeiten	3
2.1	Archiv-Qualität	3
2.2	Captcha	4
2.3	Fehlerseiten	5
2.4	pornografischer Inhalt	6
3	Problembeschreibung	7
3.1	Fehlerseiten	7
3.2	Captcha	8
3.3	PopUp	8
3.4	pornografischer Inhalt	8
4	Vorbereitung	13
4.1	Web Archiv	13
4.2	Crowdsourcing Annotation	13
5	Merkmale	16
5.1	Helligkeit und Farbanteil	16
5.2	Webseiteninhalt	19
5.3	HTML Attribute	19
5.3.1	title Tag	19
5.3.2	id und class Attribut	20
5.3.3	iframe source Attribut	20
5.4	Wortanzahl	21
5.4.1	Captcha	21
5.4.2	PopUp	22
5.5	OpenCV SIFT	22

6 Aufbau der Experimente	26
6.1 Ablauf	26
6.2 Klassifizierung mit Weka	27
6.2.1 StringToWordVector	27
6.2.2 J48	28
6.2.3 Maße	28
7 Auswertung	31
8 Fazit	38
9 Future Work	41
Literaturverzeichnis	43

Danksagung

An dieser Stelle möchte ich dem Betreuer meiner Arbeit, Johannes Kiesel, für seine Unterstützung und konstruktiven Kritik während der Erstellung der Arbeit danken.

Ich danke Prof. Dr. Benno Stein und Prof. Dr. Ing. Volker Rodehorst dafür, dass sie meine Arbeit unter ihre Aufsicht angenommen haben.

Zudem möchte ich mich bei Anny Hißbach bedanken die sich meine Arbeit durchgelesen hat und bei meiner Schwester für ihre Unterstützung.

Kapitel 1

Einleitung

Es gibt über 1,6 Milliarden Webseiten¹ im Internet, davon sind ungefähr 200 Millionen Seiten aktiv und der Rest sind geparkte Domains oder ähnliches. Allerdings ändert sich diese Zahl ständig, da neue Seiten hinzukommen und alte entfernt oder gelöscht werden.

Deswegen werden Webarchive angelegt in denen die Seiten gespeichert werden, damit diese Informationen nicht verloren gehen und später noch zur Verfügung stehen. Diese können dann entweder zur Webanalyse verwendet werden oder die archivierten Seiten werden der Öffentlichkeit zugänglich gemacht, sodass sie weiter als Informationsquelle genutzt werden können. Zwei gute Beispiele sind an dieser Stelle das Internet Archive und Common Crawl. Das Internet Archive² ist eine gemeinnützige Organisation, welche es sich zur Aufgabe gemacht hat Webseiten, digitalisierte Bücher, Videos und Bilder der Öffentlichkeit, in einer digitalen Bibliothek frei zur Verfügung zustellen. 1996 haben sie mit dem Archivieren von Webseiten begonnen und seitdem wurden 349 Milliarden Webseiten³ gespeichert. Common Crawl ist ebenfalls eine gemeinnützige Organisation, welche seit 2011 monatlich Webseiten crawlt und mittlerweile Petabytes an Daten beinhaltet. Die Archive enthalten die Daten der Webseiten, sowie Metadaten und extrahierte Texte⁴. Diese stehen öffentlich zur Verfügung und können zur Webanalyse verwendet werden. Der hier verwendete Datensatz Webis-Web-Archive-17 wurde von Kiesel et al. [7] gecrawlt aus URLs von Webseiten aus dem Common Crawl um eine Mischung aus bekannten und weniger bekannten Webseiten zu erhalten.

Das Problem beim Erstellen von Webarchiven ist allerdings, dass später nur

¹<http://www.internetlivestats.com/total-number-of-websites/> (abgerufen am 12.03.2019)

²<https://archive.org/about/> (abgerufen am 12.03.2019)

³<https://archive.org/web/> (abgerufen am 12.03.2019)

⁴<http://commoncrawl.org/the-data/get-started/> (abgerufen am 12.03.2019)

die Daten abrufbar sind, die zum Zeitpunkt der Archivierung gespeichert wurden. Wenn man also Fehlerseiten archiviert hat, geht der eigentliche Inhalt der Seite verloren. Bei Captchas kann es sein, dass man erst das Captcha lösen muss, damit die gewünschte Seite geladen wird. Bei PopUps ist das Problem, dass der Inhalt der Webseite zum Teil erst vollständig geladen wird nachdem das PopUp geschlossen wurde.

Da diese Seite verhindern, dass der ursprüngliche Inhalt geladen wird, möchte man diese Seiten nicht archivieren und da die Archive meist mehrere tausende Webseiten enthalten ist eine Annotation per Hand zu Zeit aufwendig.

Aus diesem Grund ist es notwendig eine automatische Erkennung für solche problematischen Seiten zu haben welche eine Warnung beim archivieren ausgibt, sollte es sich um eine problematische Seite handeln.

Das Ziel der Arbeit ist es herauszufinden wie gut man solche Webseiten automatisch erkennen kann und welche Merkmale der Webseiten sich für eine solche Erkennung eignen. Neben den oben genannten drei Kategorien wurde außerdem noch die Erkennung von pornografische Inhalten für den Fall eingeschlossen, dass man diese aus seinem Datensatz filtern möchte.

Für die Erkennung liegt der Fokus auf einigen Merkmalen, welche problematische Webseiten von anderen differenzieren. Folgende wurden verwendet:

- Helligkeit, sowie Rot-, Grün- und Blauanteil des Screenshots der Webseite
- HTML Attribute wie id, class und source, sowie der title Tag
- Webseiteninhalt
- Wortanzahl
- Bilderkennung von Captchas

Aus diesen Merkmalen wurden Trainings- und Testdatensets erstellt. Mittels maschinellem Lernen und dem Framework Weka wurde für jede Kategorie und jedes Merkmal berechnet, wie gut es sich eignet, diese zu erkennen. Als Maße wurden dafür Precision, Recall und F-Measure, sowie J48 als Verfahren um Entscheidungsbäume zu lernen, verwendet.

Kapitel 2

Verwandte Arbeiten

Diese Arbeit befasst sich mit der Erkennung von vier Webseiten-Kategorien um die Qualität von Webarchiven, auch für spätere Analysen, zu verbessern. Dadurch reiht sie sich in existierende Arbeiten in Archiv-Qualität, Lösen von Captchas und dem Auffinden von pornografischen Inhalten auf Webseiten ein. Dieses Kapitel gibt eine Übersicht darüber, welche verwandten Arbeiten bereits auf dem Gebiet existieren. Als Erstes wird darauf eingegangen welche Faktoren die Archiv-Qualität beeinflussen. Danach wird genauer auf die einzelnen Kategorien eingegangen und welche Ansätze zur Erkennung bereits existieren.

2.1 Archiv-Qualität

Die Qualität von Webarchiven kann auf verschiedene Arten gemessen werden. Faktoren die darauf einen Einfluss haben sind zum Beispiel die Kohärenz der archivierten Seiten, Duplikatfreiheit, Einfluss von JavaScript und wie gut bestehende Archivierungstools Webseiten archivieren können.

Zum einen führt wiederholtes Crawlen von Seiten, die sich seit dem letzten Crawl nicht verändert haben dazu, dass Duplikate entstehen. Dadurch werden wertvolle Ressourcen wie Speicherplatz verschwendet. Neue Crawler, wie zum Beispiel Heritrix, verfügen allerdings über eine Funktionalität welche das Downloaden von Duplikaten¹ minimiert. Andererseits führt wiederholtes Crawlen von sich veränderten Webseiten dazu, dass diese nicht kohärent gespeichert werden. Das Crawlen von Webseiten für komplexere Webauftritte erstreckt sich meist über einen längeren Zeitraum. Wenn währenddessen Änderungen an den Webseiten vorgenommen werden führt das dazu, dass die Webseiten

¹<https://github.com/internetarchive/heritrix3/wiki/Duplication%20Reduction%20Processors> (abgerufen am 12.03.2019)

nicht kohärent archiviert werden. Damit haben sich Spaniol et al. [17] beschäftigt, mit dem Ziel einen Crawler zu entwickeln der die Kohärenz des Crawls in Bezug auf einen Zeitpunkt oder ein Zeitintervall sicher stellt.

Ein weiterer Faktor der die Qualität beeinträchtigt ist, dass sich die Webseiten durch JavaScript und Ajax immer mehr zu Applikationen weiter entwickelt haben. Diese benötigen Interaktionen mit dem Klienten damit dargestellt wird, was der Nutzer sieht. Brunelle et al. [3] haben zum einen analysiert wie gut das Web mit verschiedenen Tools archiviert werden kann und zum anderen gezeigt, dass es im Laufe der Zeit durch die Verbreitung von JavaScript schwieriger wurde, Webseiten zu archivieren.

Kelly et al. [5] haben mittels eines Set von Metriken evaluiert wie gut Archivierungstools darin sind, Webseiten zu erhalten. Dabei nahmen sie Bezug auf die Ähnlichkeit zwischen der archivierten Version und der, die der Nutzer im live Web sehen würde. Um die verschiedenen Tools zu testen wurde ein *archival acid test* verwendet. Normalerweise werden Acid-Tests dazu verwendet, mittels Testwebseiten Webbrowser auf ihre Konformität bezüglich der Standards des World Wide Web Consortiums (W3C)² zu testen. Für den *archival acid test* wurden Funktionen von Webbrowsern in Webseiten implementiert. Die Seiten wurden von allen Browsern gleich angezeigt, welches nicht immer der Fall bei der archivierten Version war. Diese Unterschiede zwischen den beiden Versionen können genutzt werden um herauszufinden, welche Funktionalität den Archivierungstools fehlt.

AlNoamany et al. [1] haben sich mit der Erkennung von Seiten beschäftigt die im Laufe der Zeit ihren Inhalt im Vergleich zu ihrer archivierten Version zum Archiv geändert haben. Das kann beispielsweise dadurch verursacht werden, dass die Seite gehackt wurde, der Account nicht mehr existiert oder die Domain abgelaufen ist. Diese Seiten wurden dann gekennzeichnet um sie wieder erkennen zu können.

Erdélyi et al. [4] haben einen Ansatz entwickelt wie man Spam in Webarchiven erkennen und dadurch vermeiden kann. Denn Webspam verschwendet wertvolle Ressourcen wie Speicher und Bandbreite. Jedoch kann es auch nützlich sein Spam zu archivieren um einen Korpus zur Verwendung für Forschungszwecke zu haben.

2.2 Captcha

Die Ansätze die für Captchas existieren beschäftigen sich damit, diese mit Maschinen zu lösen und nicht wie man Seiten erkennt, welche Captchas bein-

²<https://www.w3.org/> (abgerufen am 12.03.2019)

halten. Ein Ansatz³ ⁴ der für ReCaptcha v2 von Google funktionierte, war es die Captchas mit dem Spracherkennungsservice von Google zu lösen. Dabei wurde als Option um das Captcha zu lösen, die Audio Challenge ausgewählt. Diese Datei wurde dann heruntergeladen, das Dateiformat in .wav konvertiert und dann in den Sprachassistenten von Google gegeben. Dieser hat dann eine Textausgabe geliefert, welche die Lösung des Captchas ist und anschließend in die Textbox eingegeben wurde. Allerdings ist dies nicht mehr möglich da Google die Komplexität erhöht hat, so dass die Texte länger sind und zudem wurden Hintergrundgeräusche hinzugefügt.

Sivakorn et al. [16] haben sich mit einem Ansatz beschäftigt der mittels Deep Learning semantische Informationen aus Bildern extrahiert um dadurch die reCaptchas zu lösen, welche Bilder anzeigen. Die Ergebnisse wurden Google mitgeteilt und das Bild reCaptcha wurde überarbeitet, so dass es sicherer gegen solche Angriffe ist.

Stark et al. [18] haben sich ebenfalls damit befasst Captchas mittels Deep Learning zu lösen. Allerdings bezieht sich ihr Ansatz auf Captchas mit Texteingabe. Vor allem auf neuere Captchas, deren Texte verzerrt sind und eine Segmentierung des Textes schwieriger machen. Dafür wurde aktives Lernen mittels eines Deep Convolutional Neural Network , kurz CNN, durchgeführt und aus dem Deep CNN wurde eine Unsicherheit berechnet. Die Beispiele welche als korrekt, aber unsicher klassifiziert wurden, wurden verwendet um dadurch den Trainingsdatensatz zu vergrößern. Dadurch ist kein großer Trainingsdatensatz notwendig.

2.3 Fehlerseiten

Zur Erkennung von Fehlerseiten existieren mehr Möglichkeiten zum Beispiel link rot oder die Erkennung von soft-404 Seiten.

Sanderson et al. [14] haben sich mit link rot beschäftigt, was bedeutet das Hyperlinks nicht „aufgelöst“ werden können. Dabei wurde untersucht welche Webressourcen, die in akademischen Arbeiten referenziert wurden, immer noch erreichbar waren. Dabei wurde bestimmt, welche dieser Kopien in Archiven sind und es wurde der zeitliche Unterschied zwischen dem Datum der Veröffentlichung, der zitierten Arbeit und dem Datum der nächsten archivierten Kopie betrachtet.

³<https://east-ee.com/2017/02/28/rebreakcaptcha-breaking-googles-recaptcha-v2-using-google/> (abgerufen am 12.03.2019)

⁴<https://github.com/eastee/rebreakcaptcha> (abgerufen am 12.03.2019)

Der InternetArchiveBot⁵ ist ein PHP-basierter, Framework unabhängiger OAuth⁶ Bot zur Wartung von defekten Weblinks. Ziel ist es diese Links durch eine archivierte Version aus dem Internet Archive zu ergänzen. Dieser Bot ist Teil eines Kooperationsprojektes der Wikimedia Foundation mit dem Internet Archive.

Mit der Erkennung von sogenannten soft-404 Seiten haben sich beispielsweise Bar-Yossef et al. [2], Meneses et al. [11] und Prieto et al. [13] befasst. Man spricht von einer soft-404 Seite wenn der Server nicht wie zu erwarten mit einem 404 Statuscode antwortet, sondern stattdessen einen 200 Ok Statuscode als Antwort sendet und dann auf die Homepage oder eine andere Seite weiterleitet, welche einen Text mit Fehlermeldung beinhaltet. Bar-Yossef et al. [2] haben sich damit befasst wie man bestimmen kann ob eine Seite nicht mehr existiert oder nicht mehr gut gepflegt wird, indem sie den Verfall der Seite berechnet haben. Der Verfall einer Seite berechnet sich indem betrachtet wird wie viele Seiten die von der Seite verlinkt sind, schon nicht mehr existieren. Meneses et al. [11] haben sich damit befasst einen verlässlichen Mechanismus zu entwickeln, welcher soft-404 Seiten erkennt. Dazu wurde der Inhalt der Webseiten analysiert und die Metadaten die aus dem Titel des Quellcodes extrahiert wurden. Prieto et al. [13] haben einen Soft404Detector entwickelt, welcher basierend auf Webinhaltsanalyse soft-404 Seiten identifiziert.

2.4 pornografischer Inhalt

Mit der Erkennung von pornografischen Bildern oder Inhalten auf Webseiten haben sich unter anderem Marcial-Basilio et al. [10] und Sengamedu et al. [15] beschäftigt.

Marcial-Basilio et al. [10] haben einen Algorithmus entwickelt der pornografische digitale Bilder und Inhalte erkennt. Dabei wurde die Transformation von Farbmodellen verwendet. Zudem wurde das Bild mittels Hauterkennung segmentiert und der Prozentsatz der Pixel die als Hautfarbe erkannt wurden, wurden berechnet.

Sengamedu et al. [15] haben ein Framework zur Erkennung von pornografischen Inhalten auf Webseiten entwickelt. Dazu wurden die Erkennung von Körperteilen, Gesichtern, sowie Farb- und Textfeatures verwendet.

⁵<https://de.wikipedia.org/wiki/Benutzer:InternetArchiveBot> (abgerufen am 11.03.2019)

⁶OAuth ist ein offenes Protokoll welches eine standardisierte, sichere API Autorisierung bietet <https://de.wikipedia.org/wiki/OAuth> (abgerufen am 11.03.2019)

Kapitel 3

Problembeschreibung

In diesem Kapitel werden die einzelnen Kategorien genauer beschrieben und erklärt, weswegen diese problematisch sind. Zudem gibt es für jede Kategorie eine bildliche Darstellung zur Veranschaulichung. Für das weiterer Verständnis der Arbeit ist wichtig zu wissen was im Rahmen dieser Arbeit als Webauftritt und als Webseite bezeichnet wird.

Ein Webauftritt beinhaltet mehrere Webseiten die zusammen gehören.

Eine Webseite ist nur die einzelne Seite die man gerade sieht. Zu den meisten Webauftritten gehören mehrere Webseiten. Für die Klassifizierung wurde jede Webseite einzeln klassifiziert. Die Kategorien wurden wie folgt definiert:

3.1 Fehlerseiten

Bei Fehlerseiten wird der Inhalt der Webseite nicht geladen, zum Beispiel auf Grund von Serverwartungen oder weil die referenzierte Seite nicht mehr existiert. Der Server erhält auf jede HTTP-Anfrage einen HTTP-Statuscode als Antwort und mittels dieses Codes lassen sich Fehlerseiten identifizieren. Das hier verwendete Webis-Web-Archive-17 enthält nur sogenannte soft Fehlerseiten. Das heißt der Server sendet einen 200 Ok Statuscode und leitet auf eine Seite weiter, auf der dann die Fehlermeldung, beispielsweise 404 Not found oder 503 Service Unavailable, angezeigt wird.

Fehlerseiten können Webseiten unterschiedlich stark beeinträchtigen. Abbildung 3.1 zeigt ein Beispiel einer Seite, die zwar eine Fehlermeldung beinhaltet, aber nur leicht dadurch beeinträchtigt wird, da noch weiterer Inhalt der Seite geladen wird. Abbildung 3.2 zeigt ein Beispiel einer Seite bei der der ursprüngliche Seiteninhalt aufgrund der Fehlermeldung nicht angezeigt wird. Die Seite wird durch die Fehlermeldung also stark beeinträchtigt.

3.2 Captcha

Captchas sind automatisierte Tests bei denen Interaktion notwendig ist. Die Aufgaben eines Captchas sind für Menschen einfach zu lösen, sollen aber für Maschinen schwer zu lösen sein, deshalb werden sie genutzt um zu identifizieren ob der Nutzer ein Mensch ist. Captchas verhindern möglicherweise, dass die Seite korrekt geladen wird.

Abbildung 3.3 zeigt eine Webseite bei der das Captcha nur gelöst werden muss wenn man sich anmeldet. Die Seite beinhaltet zwar ein Captcha, der Seiteninhalt wird dadurch allerdings nicht beeinflusst, da es nur eine Interaktion, in dem Fall die Anmeldung, verhindert. Abbildung 3.4 zeigt ein Beispiel für eine Seite, die durch das Captcha stark beeinträchtigt wird und das Captcha erst gelöst werden muss bevor der Seiteninhalt geladen wird.

3.3 PopUp

PopUps tauchen ebenfalls häufig auf Webseiten auf und variieren in ihrer Erscheinung. Manche sind kaum wahrnehmbar und stellen keine Beeinträchtigung dar. Andere hingegen sind sehr groß, haben zum Teil dunkle Overlays und können verhindern, dass der Seiteninhalt vollständig geladen wird, bis das PopUp geschlossen wird.

Abbildung 3.5 zeigt ein Beispiel für eine Seite die ein deutlich erkennbares PopUp beinhaltet, aber der Seiteninhalt ist erkenntlich und wird nicht durch das PopUp beeinträchtigt. In Abbildung 3.6 überdeckt das PopUp jedoch den Seiteninhalt und zudem noch ein dunkles Overlay. Das Anzeigen des ursprünglichen Seiteninhalts wird durch das PopUp stark beeinträchtigt und erfordert Schließen des PopUps damit der ursprüngliche Seiteninhalt angezeigt wird.

3.4 pornografischer Inhalt

Pornografische Inhalte stellen keine Beeinträchtigung für die Webseite und deren Ansicht dar. Da es sich in diesem Fall allerdings um nicht jugendfreie Inhalte handelt, kann es wünschenswert sein diese herauszufiltern. Aus diesem Grund ist eine Erkennung solcher Seiten ebenfalls hilfreich. Abbildung 3.7 zeigt ein Beispiel einer Seite mit pornografischem Inhalt.

KAPITEL 3. PROBLEMDESCRIBUNG

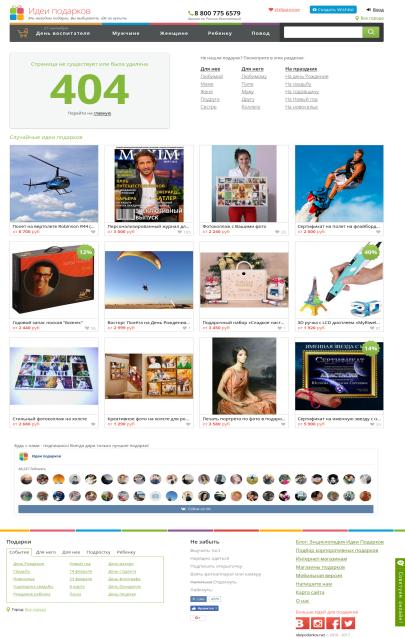


Abbildung 3.1: Zu sehen ist ein Screenshot der ein Beispiel einer Seite zeigt, die nur zum Teil durch eine Fehlermeldung beeinträchtigt wird.

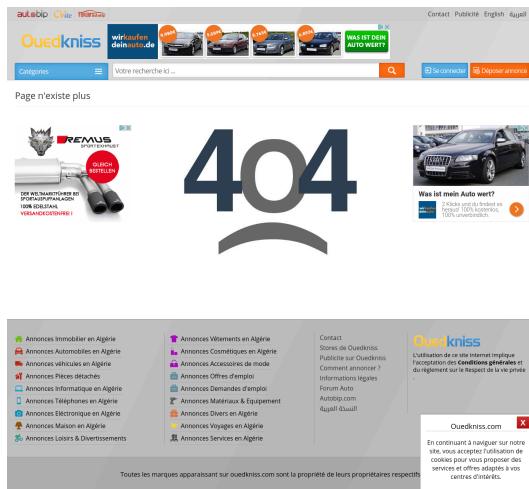
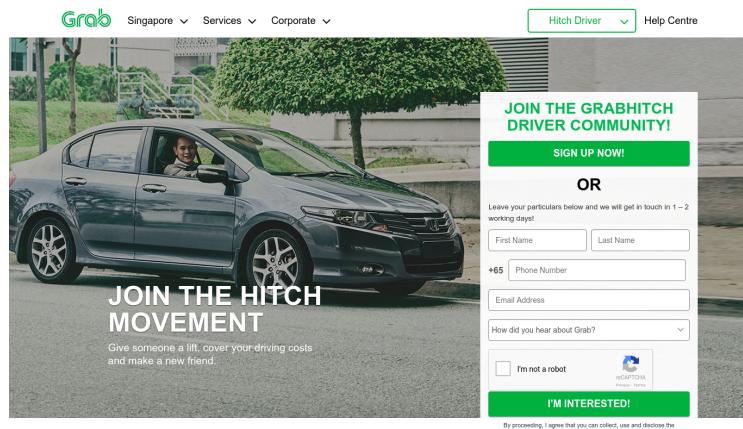


Abbildung 3.2: Beispiel-Screenshot einer Seite die stark durch die Fehlermeldung beeinträchtigt wird, da der ursprüngliche Seiteninhalt nicht angezeigt wird.



What is GrabHitch?



GrabHitch is a social carpooling service that enables regular drivers like you and me to give fellow commuters a lift, meet new people, and cover the costs of owning a car.



To make it truly hassle-free, GrabHitch accepts both GrabPay and cash payments so GrabHitch drivers can choose their preferred method for cash out. All GrabHitch driver registration and bookings can be done through the regular Grab Passenger App, and do not require commercial insurance or Z10 registration.

Abbildung 3.3: Beispiel-Screenshot einer Seite bei der das Captcha nur das Ausführen von bestimmten Aktionen, in dem Fall die Anmeldung verhindert. Der Seiteninhalt wird dadurch nicht beeinflusst. Zur Veranschaulichung wird nur ein Teil des Screenshots gezeigt.



Abbildung 3.4: Zu sehen ist ein Screenshot der ein Beispiel einer Seite zeigt bei der das Captcha den Zugang zur ursprünglichen Webseite verhindert und dadurch die Seite stark beeinträchtigt.

KAPITEL 3. PROBLEM Beschreibung

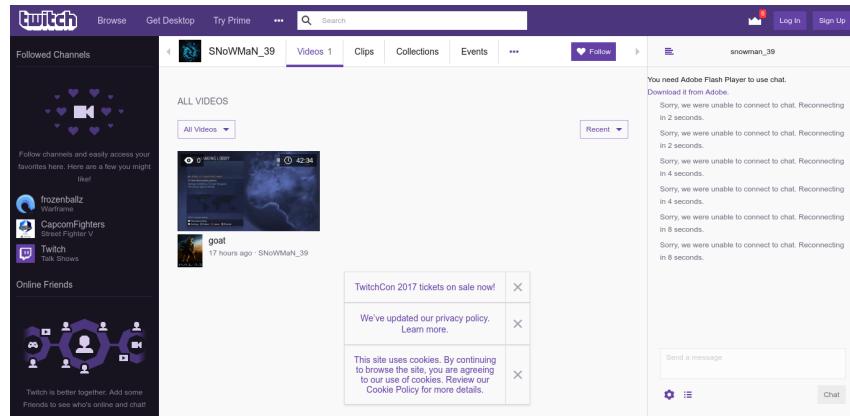


Abbildung 3.5: Zu sehen ist ein Screenshot der ein Beispiel einer Seite zeigt auf der das PopUp deutlich zu erkennen ist, aber der Seiteninhalt ist ohne Beeinträchtigung erkennbar.

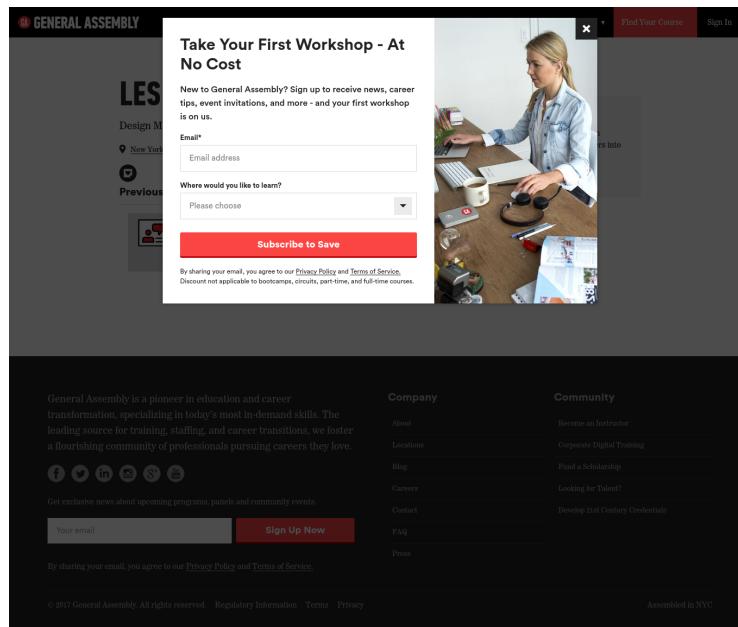


Abbildung 3.6: Beispiel-Screenshot einer Seite bei der das PopUp und das dunkle Overlay den Seiteninhalt verdecken und dadurch die Seite stark beeinträchtigen.

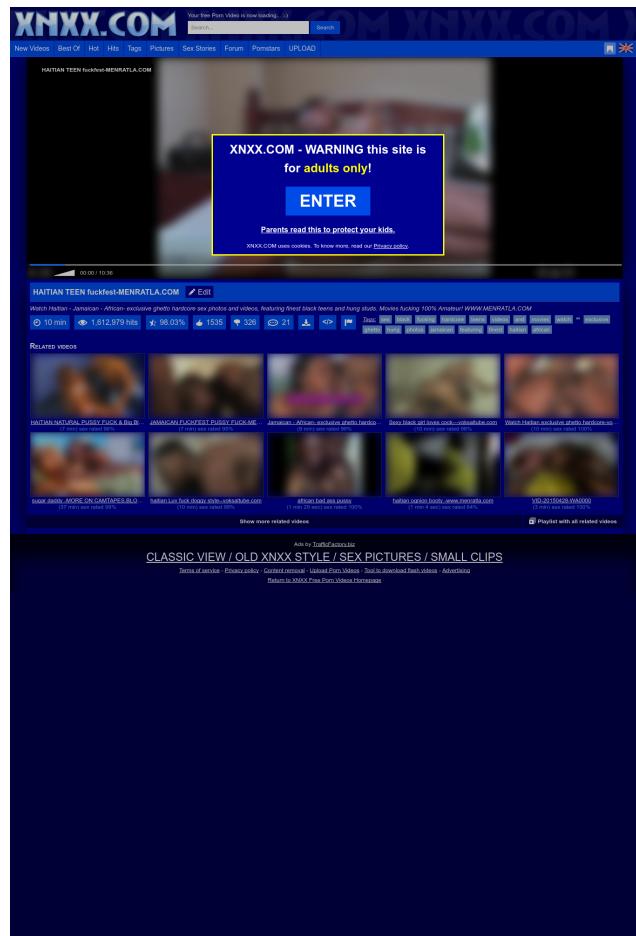


Abbildung 3.7: Zu sehen ist ein Screenshot der ein Beispiel einer pornografischen Seite zeigt.

Kapitel 4

Vorbereitung

Um eine automatische Klassifizierung durchführen zu können, wurde ein annotierter Datensatz als Grundlage für supervised learning benötigt. Dafür wurde das Webis-Web-Archive-17 verwendet, welcher mittels Crowdsourcing Annotation von Menschen annotiert wurde [8]. Das folgende Kapitel geht kurz auf das Archiv ein und danach auf die Crowdsourcing Annotation.

4.1 Web Archiv

Das Webis-Web-Archive-17 besteht aus 10000 Webseiten. Um sicher zustellen, dass sowohl bekannte als auch weniger bekannte Seiten vorkommen, wurde basierend auf URLs die aus dem Common Crawl stammen gecrawlt. Das Archiv beinhaltet alle Seiten als Quellcode, .warc Archiv und einen Screenshot der jeweiligen Seite. Die Screenshots waren Grundlage der ersten Crowdsourcing Annotation.

4.2 Crowdsourcing Annotation

Die Crowdsourcing Annotation wurde Anfang des Jahres 2018 als Vorbereitung für die Bachelorarbeit im Rahmen einer hilfswissenschaftlichen Tätigkeit durchgeführt [8].

Im ersten Durchlauf wurden die Webseiten von Menschen mit Amazons Crowdsourcing Plattform Mechanical Turk¹ annotiert. In Abbildung 4.1 ist das Interface zusehen welches verwendet wurde.

Die Aufgabe war es zu bewerten, ob ein Screenshot einer Webseite in eine der folgenden Kategorien passt: Seite welche größtenteils aus Werbung besteht, Seite lädt noch, Seiteninhalt abgeschnitten, Pornografie, PopUp, Captcha oder

¹<https://www.mturk.com/> (abgerufen am 12.03.2019)

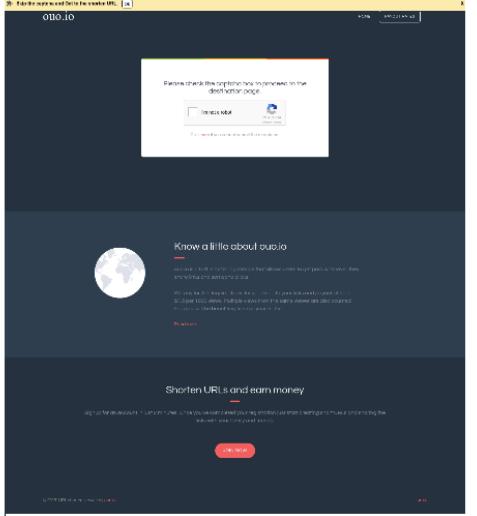
Fehlerseite. Für die ersten vier Kategorien konnten die Frage mit ja/nein beantwortet werden. In Kapitel 3 wurde beschrieben und anhand von Abbildungen gezeigt, dass Captchas, Fehlerseiten und PopUps in unterschiedlichen Beeinträchtigungen auftreten. Deswegen wurde für diese drei Kategorien zwischen not, abit und very differenziert, wobei not bedeutet das keine Beeinträchtigung sichtbar ist, abit sie ist bemerkbar aber verhindert nicht die Verwendung der Webseite und very bedeutet, dass die normale Verwendung der Seite durch die Beeinträchtigung nicht möglich ist.

Die Screenshots wurden von jeweils fünf verschiedenen Workern bewertet.

Instructions

- Select the answers that best describe the web page images.
- Look at each image carefully. Scroll to its bottom if it is not fully shown.
- Read the Guidelines for the second question-group carefully before answering.
- You have to select one answer for each question and for each of the 5 images.
- This task requires JavaScript.

Image 2



Select the answers that best describe the web page image on the left.

From the image, would you say the web page is... yes no

Mostly in a language that you understand?

Mostly advertisement? (page consists mostly of ads)

Still loading? (e.g., showing progress or activity indicator)

Pornographic? (contains sexually explicit content)

Cut off? (you can not scroll to the page bottom)

In the image, how dominant are the... not a bit very

Pop-Ups? (elements that cover other elements)

Captchas? (tests to prevent bots accessing/commenting/...)

Error messages? ("404", page not found, timed out, ...)

Guidelines:
not: not on this image or barely noticeable at all
a bit: clearly noticeable, but can be ignored (e.g., pop-ups with transparent overlays, pop-ups/error messages/Captchas near the page border)
very: can not be ignored (pop-up or Captcha that requires attention before using the page, error message or Captcha instead of content)

Comments for this image (optional):

Hints: Zoom by clicking on the image. Use the spacebar to go the next image.

1 2 3 4 5 Next

Abbildung 4.1: Interface welches zur Annotation genutzt wurde

Um bewerten zu können wie gut die Worker gearbeitet haben, wurde die Zeit gemessen welche für die Bearbeitung der Aufgabe benötigt wurde und die Übereinstimmung mit anderen Worker. Zudem wurden Seiten bei denen es Unstimmigkeiten zwischen den Workern gab nochmals kontrolliert. Abschließend wurde eine Kuratoren-Oberfläche verwendet um nochmals die Annotation

zu kontrollieren und gegebenenfalls die Annotation zu korrigieren. Dabei kam heraus, dass ungefähr 1.35 % der Seiten Captchas beinhalten, 4.5 % der Seiten Fehlerseiten sind, 7.03 % PopUps beinhalten und 4.09 % der Seiten pornografischen Inhalt haben. Aus der Annotation der Worker und den überarbeiteten Annotationen wurde dann eine Ground Truth Datei erstellt, welche die Grundlage für supervised learning und die Klassifizierung der Kategorien bildet.

Kapitel 5

Merkmale

Um die einzelnen Kategorien von problematischen Webseiten automatisch erkennen zu können, sind Merkmale notwendig die charakteristisch für diese Seite sind und sie von normalen Webseiten differenzieren. Im Folgenden wurde sich auf sieben Merkmale beschränkt. Diese konnten besonders gut aus den gegebenen Daten (Bilder, Quellcode und .warc Datei) extrahiert werden.

5.1 Helligkeit und Farbanteil

Beim durchschauen der Webseiten ist aufgefallen, dass pornografische Seiten und Seiten mit PopUps eher dunkel sind und sich deswegen die Helligkeit- und Farbwerte als Merkmal eignen könnten zur Erkennung dieser Seiten. Daraufhin wurden die Helligkeit- und die Farbwerte für alle Seiten berechnet und Histogramme für die einzelnen Kategorien erstellt um die Annahme zu bestätigen. Wie man in Abbildung 5.1 und Abbildung 5.2 erkennen kann, haben Seiten welche pornografische Inhalte oder PopUps beinhalten einen geringeren Helligkeitswert. Da man für die beiden Kategorien, pornografische Seiten und PopUps, einen deutlichen Unterschied zu den anderen, nicht problematischen Seiten erkennen kann, kann man davon ausgehen dass sich das Merkmal Helligkeit und Farbwert zur Erkennung der beiden Kategorien eignet. Im Vergleich dazu haben viele Seiten welche Captchas beinhalten einen großen Helligkeitswert, siehe Abbildung 5.3. Dieses Merkmal könnte sich demzufolge ebenfalls zur Erkennung von Captchas eignen. Bei Fehlerseiten kann man keinen Unterschied zu den normalen Seiten erkennen, siehe Abbildung 5.4, von daher kann man davon ausgehen, dass sich dieses Merkmal nicht zur Erkennung von Fehlerseiten eignen wird.

Um den Helligkeitswert für ein Bild zu berechnen wurde ImageMagick¹ ver-

¹<https://www.imagemagick.org/> (abgerufen am 09.04.2019)

wendet. Zu erst wurde das Bild in Graustufen umgerechnet und anschließend wurde der Mittelwert daraus gebildet. Die daraus errechneten Werte liegen zwischen 0(schwarz) und 1(weiß).

Um einen Rot-, Grün- und Blauanteil für das gesamte Bild zu berechnen, wurde das es zu erst auf eine Größe von ein Pixel verkleinert und dafür wurden dann die Farbwerte berechnen. Diese liegen immer zwischen 0 bis 255.

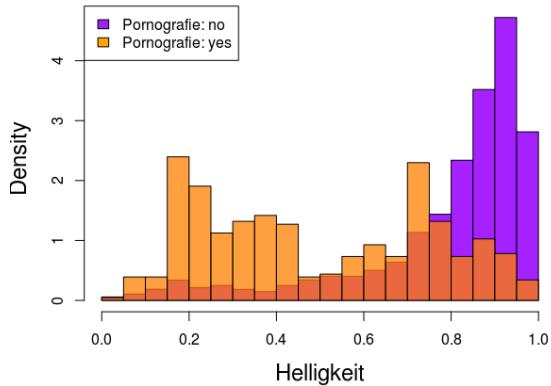


Abbildung 5.1: Histogramm für die Kategorie Pornografie, orange dargestellt Seiten mit pornografischen Inhalt, lila dargestellt alle anderen Seiten

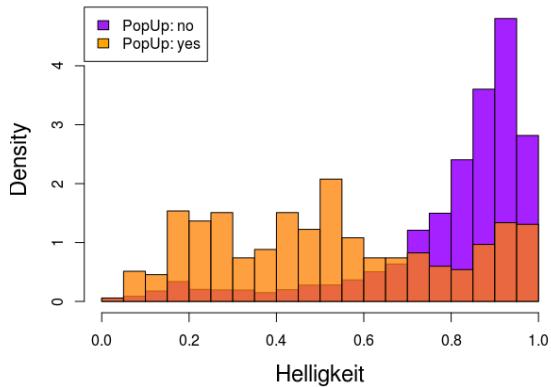


Abbildung 5.2: Histogramm für die Kategorie PopUp, orange dargestellt die Seiten welche PopUps beinhalten, lila dargestellt alle anderen Seiten

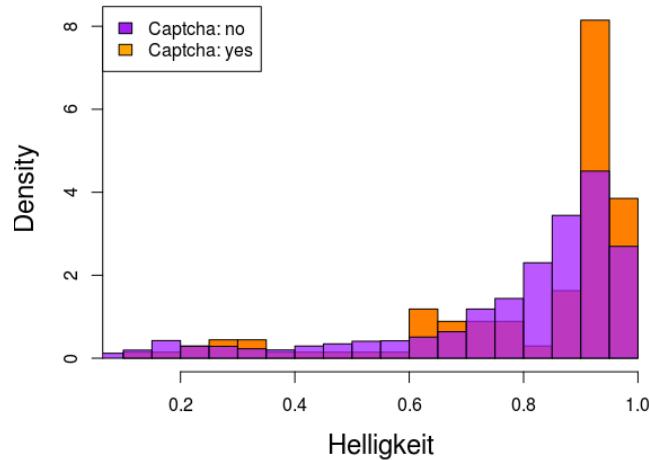


Abbildung 5.3: Histogramm für die Kategorie Captcha, orange dargestellt die Seiten welche CAPTCHAs beinhalten, lila dargestellt alle anderen Seiten

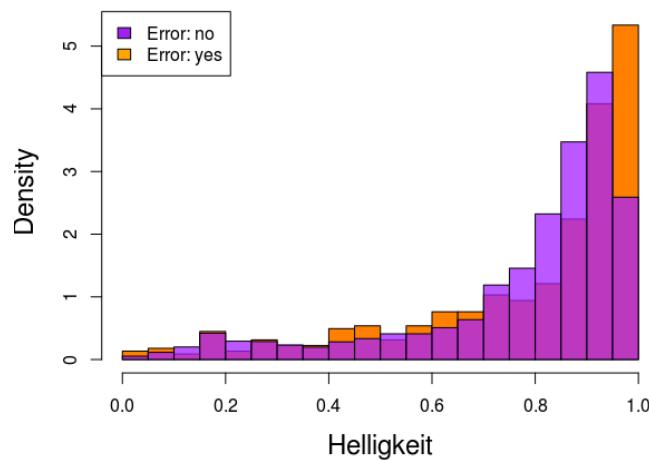


Abbildung 5.4: Histogramm für die Kategorie Fehlerseiten, orange dargestellt Fehlerseiten, lila dargestellt alle anderen Seiten

5.2 Webseiteninhalt

Der Webseiteninhalt könnte sich Erkennung von Fehlerseiten, Seiten mit Captchas oder pornografischen Inhalten eignen. Die Annahme ist dass der Webseiteninhalt von Fehlerseiten charakteristische Wörter wie 404 page not found oder 503 service unavailable beinhaltet und somit erkannt werden. Pornografische Seiten unterscheiden sich von anderen Webseiten durch ihren pornografischen Inhalt wodurch sie erkannt werden können und Seiten mit Captchas könnten dadurch erkannt werden, dass das Wort Captcha im Webseiteninhalt auftaucht.

Um den Webseiteninhalt zu extrahieren wurde der Open Source potthast-jericho-extractor[6] verwendet. Dieser rendert den Text der Webseite. Normalerweise werden Paragraphen mit weniger als 400 Textzeichen entfernt. Im Archiv sind jedoch einige Seiten enthalten die nicht viel Text beinhalten und um für möglichst viele Seiten, so viel Text wie möglich zu extrahieren, wurde die minimale Paragraphenlänge auf null gesetzt.

5.3 HTML Attribute

Um die Attribute aus dem Quellcode zu extrahieren wurde die Python Bibliothek Beautiful Soup verwendet. Mit Beautiful Soup kann man Daten aus HTML und XML Dateien extrahieren und selbst einen Parser wählen. Hierfür wurde der HTML Parser verwendet der in der Python Standardbibliothek eingebunden ist. Es wurden einige Attribute ausgewählt die sich besonders gut zur Erkennung eignen. Dazu gehören:

5.3.1 title Tag

Im <title> Tag ist der Titel der Webseite eingebunden und dieser ist für Fehlerseiten, siehe Abbildung 5.5, und pornografische Seiten, siehe Abbildung 5.6, sehr charakteristisch. Deswegen wurde angenommen, dass sich dieser zur Erkennung der beiden Kategorien eignet.

Um die Annahme zu bestätigen wurden die title Tags aus allen Seiten extrahiert und dann für die als Fehlerseiten und pornografisch klassifizierten Seiten Wordclouds aus den title Tags erstellt. Wie man in Abbildung 5.8a sehen kann, funktioniert es besonders gut für Fehlerseiten. Das lässt sich daran erkennen das „Page“ und „Found“ groß zu lesen sind, welches charakteristische Wörter für Fehlerseiten sind. In Abbildung 5.8b sieht man die Wordcloud für die Titel aller pornografischen Webseiten. Auch hier lässt sich sehr eindeutig erkennen, dass diese Titel zu pornografischen Seiten gehören. Die Annahme, dass sich der title Tag als Merkmal zur Erkennung der beiden Kategorien eignen kann

wurde durch die Wordclouds bestätigt. Die Wordclouds² wurden mit Python erstellt.

```
<title>Error 404 (Page not found) !!1</title>
```

Abbildung 5.5: Beispiel title Tag einer Fehlerseite

```
<title>Emily showing off - Pornhub.com</title>
```

Abbildung 5.6: Beispiel title Tag einer pornografischen Seite

5.3.2 id und class Attribut

Die id und classen Attribute wurden als Merkmal aufgenommen, da bei der Verwendung des CSS Framework Bootstrap³ oft „sprechende“ Klassennamen verwendet werden. So werden zum Beispiel PopUps zum Teil in Klassen eingebunden die den Namen „modal“ tragen. Deswegen eignet sich dieses Merkmal potentiell zur Erkennung von PopUps.

5.3.3 iframe source Attribut

Viele Webseiten verwenden die reCaptcha API von Google, welche über das source Attribut eingebunden wird, siehe Abbildung 5.7. Da man deutlich das Wort *recaptcha* lesen kann wurde die Annahme aufgestellt, dass sich dieses Merkmal zur Erkennung von Captchas eignet.

Um die Annahme zu bestätigen wurden die source Attribute extrahiert und dann für alle als Captcha klassifizierten Seiten in einer Wordcloud dargestellt. Abbildung 5.8c zeigt diese Wordcloud und man kann deutlich die Wörter *recaptcha* und Google erkennen. Das Merkmal kann sich demzufolge potentiell zur Erkennung von Captchas eignen.

```
<iframe src="https://www.google.com/recaptcha/api2  
/bframe?hl=en&v=r20170919161736&  
k=6LegWQETAAAAIIaaAhEnrkimbuOF5QJb0ZiYEK7#x96dgkyfcjq4"  
title="recaptcha challenge»</iframe>
```

Abbildung 5.7: Beispiel source Attribut einer Captchaseite, welche die Google API verwendet

²https://github.com/amueller/word_cloud (abgerufen am 12.03.2019)

³<https://getbootstrap.com/> (abgerufen am 28.03.2019)



(a) Inhalt der extrahierten title Tags von allen als Fehlerseite annotierten Seiten



(c) Inhalt der extrahierten source Attribute von allen als Captcha annotierten Seiten



(b) Inhalt der extrahierten title Tags von allen als pornografisch annotierten Seiten



(d) extrahierter Inhalt von allen Seiten

Abbildung 5.8: Wordclouds für verschiedene Kategorien von Webseiten

5.4 Wortanzahl

5.4.1 Captcha

Wie oben bereits genannt, verwenden viele Webseiten die Captchas einbinden die API von Google. Um ein weiteres Merkmal zu haben, wurden die URIs aus den .warc Dateien entnommen und pro Datei gezählt wie oft das Wort *Captcha* darin vorkommt.

WARC/1.0

WARC-Type: response

WARC-Record-ID: <urn:uuid:373d6f3c-3e37-40ad-8cb0-4a87978d1fd8>

WARC-Date: 2017-09-22T18:03:04Z

WARC-Target-URI: https://www.google.com/recaptcha/api.js?

onload=onloadCallback&render=explicit

Abbildung 5.9: WARC-Header (aus einer WARC-Datei) welcher die Antwort auf die Anfrage einer Captcha-Ressource zeigt

5.4.2 PopUp

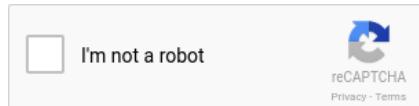
Ein ähnlicher Versuch wurde auch für PopUps durchgeführt. Dabei wurde allerdings die Anzahl des Wortes *PopUp* im Quellcode gezählt. Dieses Merkmal wurde gewählt um zu testen ob das Wort nur im Quellcode von Seiten vorkommt, die auch tatsächlich PopUps anzeigen.

5.5 OpenCV SIFT

Beim durchschauen der Webseiten ist aufgefallen, dass zwei Logos von Captchas besonders häufig auftauchen, siehe Abbildung 5.10. Um diese Captchas in den Screenshots der Webseiten zu finden, wurde Feature Matching in OpenCV mit Hilfe des SIFT(Scale-invariant Feature Transform) Algorithmus von David Lowe [9] verwendet. Dieser berechnet für ein Query Image, das Bild welches im Quellbild gesucht wird, in diesem Fall waren es zwei verschiedenen Bilder die nur das zu lösende Captcha enthielten, vergleichbare und skalierungsinvariante Schlüsselpunkte. Diese werden dann mit dem Schlüsselpunkten aus dem Quellbild, in dem Fall die Screenshots von allen Webseiten, verglichen. Damit man die Schlüsselpunkte vergleichen kann, wird jedem ein Deskriptor zugewiesen. Die beste Übereinstimmung für jeden Schlüsselpunkt wird gefunden, in dem der nearest neighbor aus der Datenbank des Quellbildes identifiziert wird. Als nearest neighbor wird der Schlüsselpunkt bezeichnet der die geringste euklidische Distanz hat. Alle Merkmale die keinen guten Treffer haben, werden verworfen in dem die Distanz zwischen dem nearest neighbor und dem second nearest neighbor verglichen wird. Diese Anzahl an guten Treffern wurde als Merkmal verwendet.

Alle Versuche wurden mit zwei verschiedenen Captchas durchgeführt. Einmal das Bild des blauen Captchas siehe Abbildung 5.10a und einmal ein rotes Captcha siehe Abbildung 5.10b. Diese wurden gewählt da sie am häufigsten vorkommen und sich am besten für das Feature Matching eignen, da sie auf jeder Seite wo sie vorkommen, gleich aussehen.

Es gibt auch noch andere Arten von Captchas, siehe Abbildung 5.11, welche nur ein Textfeld, in dem Buchstaben und Zahlen eingetippt werden müssen, beinhalten. Wie man Anhand von Abbildung 5.13a erkennt, funktioniert die Erkennung mit OpenCV auch für solches Captchas. Allerdings nur wenn man mit exakt dem Captcha sucht wie es auch vorkommt. Wie man anhand von Abbildung 5.13b erkennt funktioniert es nicht für andere Seiten, welche ähnliche Captchas beinhalten. Man müsste also genau wissen welche Captchas man sucht.



(a) Captcha Logo blau

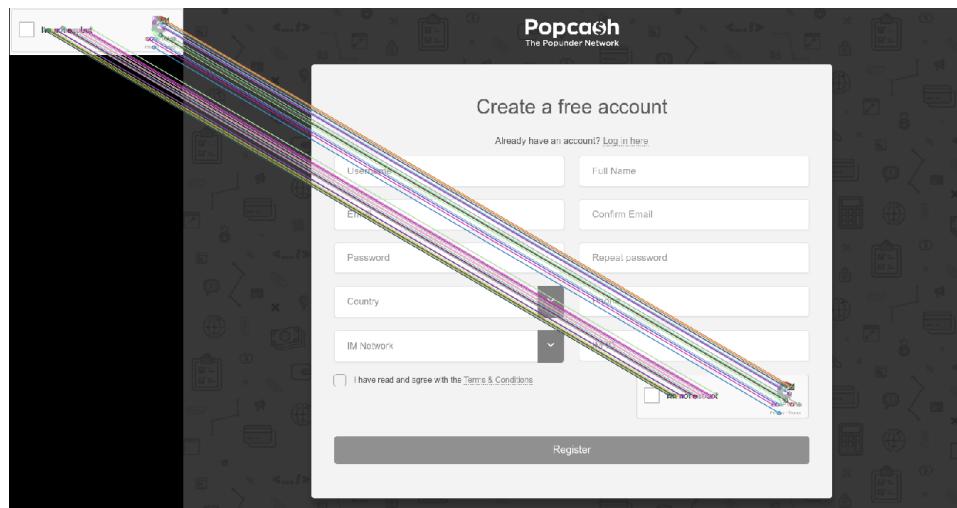


(b) Captcha Logo rot

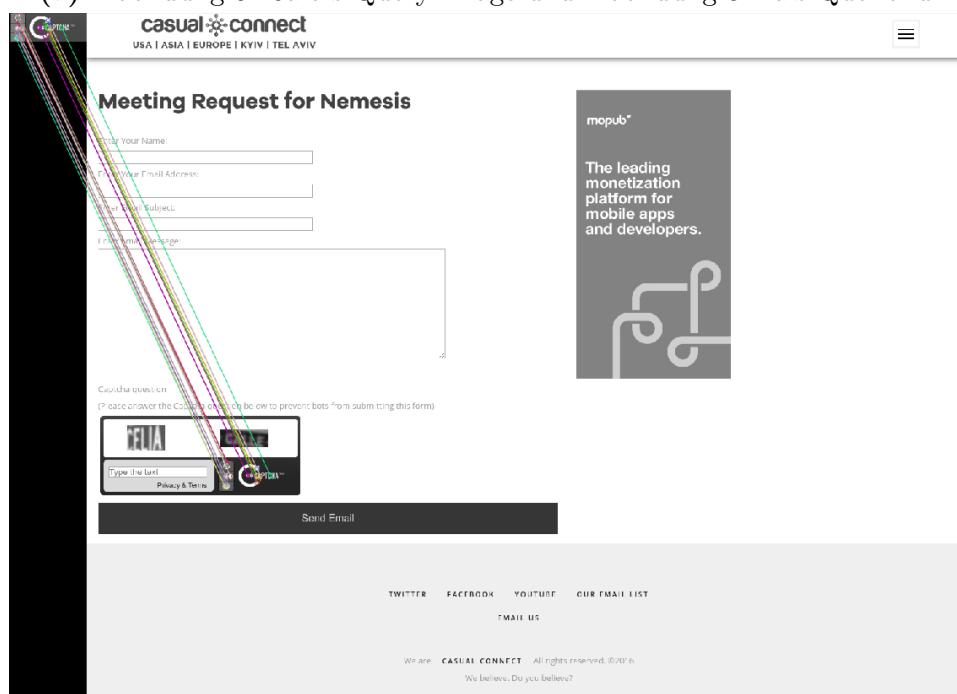
Abbildung 5.10: Captchas die als Query Image verwendet wurden



Abbildung 5.11: Beispiel eines Captcha welches nicht durch Feature-Matching erkannt wurde

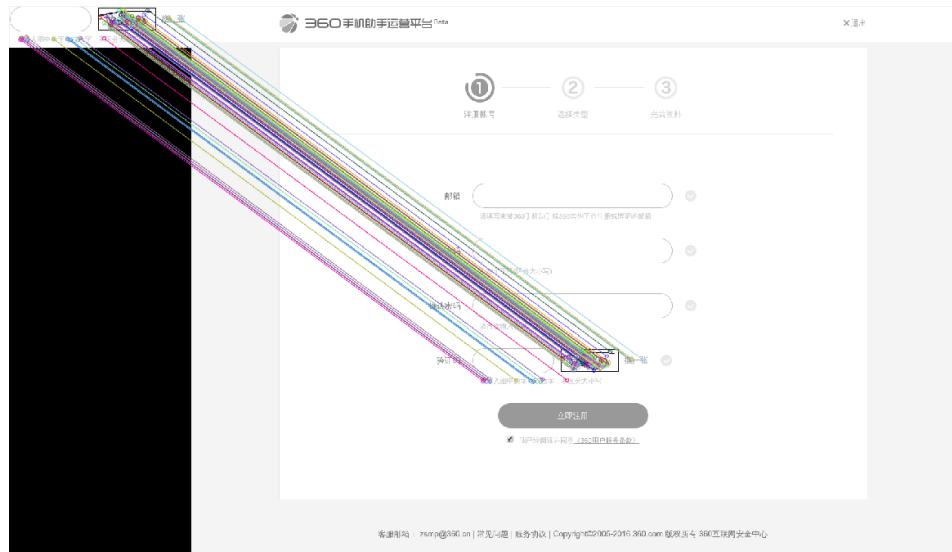


(a) Abbildung 5.10a als Query Image und Abbildung 3.4 als Quellbild



(b) Abbildung 5.10b als Query Image

Abbildung 5.12: Ergebnis des SIFT Feature Matching mit OpenCV. Die Linien zwischen dem Query Image(links oben) und dem Quellbild(rechts), repräsentieren die Anzahl der guten Treffer.



(a) Ergebnis des SIFT Feature Matching mit OpenCV mit Abbildung 5.11 als Query Image und als Quellbild die Seite welches tatsächlich dieses Captcha beinhaltet.



(b) Im Vergleich dazu Abbildung 5.11 als Query Image und als Quellbild eine Seite welche ein anderes, ähnliches CAPTCHA beinhaltet

Abbildung 5.13: In Abbildung 5.13a sieht man die Ergebnisse des Feature Matching zwischen dem Query Image und dem Quellbild, welches das Query Image beinhaltet. Die Linien repräsentieren die guten Treffer zwischen dem Query Image und dem Quellbild. Wie man erkennen kann gibt es sehr viele gute Treffer, da viele Übereinstimmungen gefunden werden. In Abbildung 5.13b sieht man wie viele gute Treffer es gibt wenn man mit dem gleichen Query Image in einem anderen Quellbild sucht. Wie man sieht gibt es nur einen Treffer, da es kaum Übereinstimmungen zwischen dem Captcha des Query Image und dem des Quellbild gibt.

Kapitel 6

Aufbau der Experimente

In diesem Kapitel wird der Aufbau und Ablauf der Experimente beschrieben. Außerdem wird darauf eingegangen welcher Klassifikator und welche Maße verwendet wurden um die Klassifizierung zu bewerten.

6.1 Ablauf

Als Erstes wurden für den gesamten Datensatz die Merkmale aus den Screenshots, dem Quellcode und den .warc Dateien extrahiert. Danach wurde kontrolliert ob für jede Webseite Werte vorhanden sind, wenn nicht wurden diese nachträglich ergänzt. So haben die Webseiten aus denen kein Inhalt extrahiert werden konnten, anstelle des Inhalts den Eintrag „none“ erhalten. Auch beim Feature Matching mit OpenCV konnten für die Screenshots die komplett weiß waren, mit dem Algorithmus keine Werte extrahiert werden. Diese Seiten haben nachträglich alle den Wert 0 bekommen.

Anschließend wurde der Datensatz in Trainings - und Testdatensatz aufgeteilt. Die Aufteilung erfolgt in zwei Drittel Trainingsdatensatz, was 6670 Seiten entspricht, und ein Drittel Testdatensatz, also 3330 Seiten. Bei der Aufteilung wurde beachtet dass Webseiten aus dem gleichen Webauftritt immer nur in einem der beiden Datensätze sind um zu vermeiden dass die Erkennung dadurch beeinflusst wird, weil eine ähnliche Seite schon im Trainingsdatensatz vorkam. Die Daten wurden dann im Eingabedateiformat .arff¹, welches von dem maschinellem Lernen Framework Weka verwendet wird, gespeichert.

Um zusehen welche Auswirkung die Einteilung in zwei oder drei Klassen auf die Klassifizierung hat, wurden zwei Testreihen gemacht.

In der ersten Testreihe wurden alle Kategorien in ihrer ursprünglichen, durch die Crowdsourcing Annotation erzielte, Klassifizierung klassifiziert.

¹<https://www.cs.waikato.ac.nz/ml/weka/arff.html> (abgerufen am 25.03.2019)

In der zweiten Testreihe wurde dann für die Kategorien Captcha, Fehlerseite und PopUp, die Klassen abit und very zu yes zusammengefasst.

Wenn man nur zwischen zwei Klassen unterscheidet ist die Entscheidung für den Klassifikator leichter, weil er nur zwischen no und yes entscheiden muss. Allerdings ist es sinnvoll die Unterteilung in drei Klassen beizubehalten. Da zum Beispiel in Fällen wo eine Seite mit Captcha abit klassifiziert ist, wird nur das ausführen bestimmter Aktionen, wie beispielsweise kommentieren verhindert. Hingegen bei Seiten die mit Captcha very klassifiziert sind, wird der ursprüngliche Seiteninhalt erst dann geladen wenn das Captcha gelöst ist. Wenn beide Seiten mit Captcha yes klassifiziert sind, kann man zwischen dieser Beeinträchtigung nicht differenzieren. Deswegen ist es besser drei Klassen zu haben, denn dadurch werden die meisten Seiten die Captchas haben erkannt und nur die die mit Captcha very klassifiziert wurden, können dann nochmal genauer betrachtet werden, zum Beispiel um in dem Fall das Captcha zu lösen. Jedoch kann abhängig vom Anwendungsfall auch eine Zwei-Klassen-Klassifizierung sinnvoll sein, wie zum Beispiel für pornografische Inhalte.

Das Model wurde auf dem Trainingsdatensatz trainiert und im Anschluss auf den Testdatensatz angewendet. Daraus ergaben sich dann die Werte für die Maße Precision, Recall und F-Measure.

6.2 Klassifizierung mit Weka

Für die automatische Klassifizierung wird das maschinelle Lernen Framework Weka verwendet und der Klassifikator J48.

6.2.1 StringToWordVector

Um Textdaten zur Klassifizierung verwenden zu können, wurde der StringToWordVector² verwendet. Dieser wandelt String Attribute in ein Set von numerischen Attributen um die widerspiegeln wie häufig ein Wort im Textstring auftaucht. Dieser wird aus dem ersten Datensatz, also dem Trainingsdatensatz erstellt.

Dafür wurden folgende Parameter gewählt:

TFIDF (Term frequency inverse document frequency) wird häufig im Information Retrieval verwendet und gibt an wie wichtig ein Wort in einem Dokument ist. Dabei gibt TF(Term frequency) an wie häufig ein Wort in einem Dokument vorkommt. IDF (Inverse document frequency) misst die allgemeine Bedeutung

²<http://weka.sourceforge.net/doc.dev/weka/filters/unsupervised/attribute/StringToWordVector.html> (abgerufen am 12.03.2019)

des Terms für die Gesamtmenge von Dokumenten.³

Ein Attribute Präfix wurde hinzugefügt, damit es keine Überschneidungen zwischen den bereits existierenden und den später extrahierten Attributen gibt. Zudem wurden lowerCaseTokens verwendet wodurch alle Buchstaben in Kleinbuchstaben umgewandelt werden bevor sie dem Wörterbuch hinzugefügt werden. So wird die Worthäufigkeit unabhängig von Groß- und Kleinschreibung berechnet. Andernfalls wird sie einmal für das großgeschriebene und einmal für das kleingeschriebene Wort gezählt. Zusätzlich wurde die Worthäufigkeit in Relation zur Länge des Dokuments berechnet mittels normalizeDocLength. Über WordCount wurde die Anzahl der Wörter ausgegeben. Außerdem wurden der IteratedLovinsStemmer verwendet, welcher eine iterierte Version des Lovinsstemmer ist, um die Wörter auf einen Wortstamm reduzieren. Als Tokenizer wurde der alkphabetic tokenizer verwendet, welcher nur Zeichen aus zusammenhängenden alphabetischen Sequenzen erstellt. Zu dem wurde auch eine Liste von Stopwörtern mit einbezogen, welche nicht mit beachtet wurden da diese keine Relevanz zur Erfassung des Dokumentinhaltes haben.

6.2.2 J48

J48 ist eine Open Source Java Implementation des C4.5 Algorithmus.

C4.5 wird verwendet um Entscheidungsbäume zu generieren, die zur Klassifizierung verwendet werden können. Die Entscheidungsbäume werden aus einem Set von Trainingsdaten unter Verwendung des höchsten Information Gain erstellt. Das Attribut mit dem höchsten Information Gain trifft die Entscheidung über die Spaltung in kleinere Sets.

Der Vorteil bei der Verwendung von J48 ist, dass man den Entscheidungsbaum sich anschauen kann. Dadurch kann sehen ob das was der Algorithmus lernt plausibel ist.

6.2.3 Maße

Um bewerten zu können wie gut sich die Modelle eignen stehen verschiedene Maße zur Auswahl. Die folgende Auswertung beschränkt sich auf drei Maße um die Leistung des Klassifikators zu analysieren. Die folgenden Maße wurden verwendet:

- Precision
- Recall
- F-Measure

³<https://de.wikipedia.org/wiki/Tf-idf-Ma%C3%9F> (abgerufen am 04.04.2019)

Für die folgenden Definitionen sind einige Begriffe notwendig welche kurz am Beispiel der Kategorie Captcha very erklärt werden, für alle anderen Kategorien funktioniert dies genauso.

- True Positive bezeichnet die Seiten die tatsächlich als Captcha very klassifiziert sind und auch als solches erkannt wurden.
- True Negative bezeichnet die Seiten die tatsächlich nicht als Captcha very klassifiziert sind und auch nicht als Captcha very erkannt wurden.
- False Positive bezeichnet die Seiten die tatsächlich nicht als Captcha very klassifiziert wurden, aber als Captcha very erkannt wurden.
- False Negative bezeichnet die Seiten die tatsächlich als Captcha very klassifiziert sind, aber nicht als Captcha very erkannt wurden.

		Erkannt			
		Captcha very	Captcha abit	Captcha not	
Tatsächlich	Captcha very	True Positive	False Negative	False Negative	Recall
	Captcha abit	False Positive	True Negative	True Negative	
	Captcha not	False Positive	True Negative	True Negative	

Precision

Precision berechnet sich aus:

$$\text{Precision} = \frac{\#\text{TruePositive}}{\#\text{TruePositive} + \#\text{FalsePositive}}$$

Im Bezug auf das Beispiel Captcha very gibt die Precision an welcher Anteil der als Captcha very erkannten Seiten tatsächlich als Captcha very klassifiziert wurden.

Recall berechnet sich aus:

$$\text{Recall} = \frac{\#\text{TruePositive}}{\#\text{TruePositive} + \#\text{FalseNegative}}$$

In Bezug auf das Beispiel Captcha very gibt der Recall an wie groß der Anteil der als Captcha very erkannten Seiten im Vergleich zu allen Webseiten ist.

F-Measure ist das harmonische Mittel und berechnet sich wie folgt aus Precision und Recall:

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Das F-Measure gibt eine gewichtete Auskunft über das Verhältnis zwischen Precision und Recall. Wenn beispielsweise der Recall sehr hoch ist, die Precision aber vergleichsweise niedrig, so ist auch das F-Measure niedrig. Gute F-Measure können also nur erzielt werden wenn sowohl Precision, als auch Recall gute Werte erzielen.

Kapitel 7

Auswertung

Im Folgenden werden die durchgeführten Testreihen beschrieben und ausgewertet. Dazu wird für jede Kategorie erst die Drei-Klassen-Klassifizierung und anschließend die Zwei-Klassen-Klassifizierung bewertet. Diese werden dann miteinander verglichen in Bezug darauf, wie gut die Klassifizierung für die jeweilige Kategorie funktioniert hat. Die Testreihen stellen den Hauptteil der Arbeit dar. Ziel ist es mittels dieser die Frage zu beantworten wie gut eine automatische Klassifizierung funktioniert.

Es ist zu beobachten, dass für alle Kategorien, unabhängig von der Aufteilung in zwei oder drei Klassen, für die mit no beziehungsweise not klassifizierten Seiten viel höhere F_1 Measure erzielt werden. Für alle Kategorien liegen diese bei über 90 %. Dies erscheint zwar hoch, klassifiziert man allerdings alle Seiten mit not beziehungsweise no so erreicht man auch bereits F_1 Measure von über 90 %. Dieser naive Klassifizierer, der alle Seiten mit beispielsweise no klassifiziert, würde in der Kategorie pornografische Seiten einen F_1 Measure von 98,2 % erzielen.

Am besten funktioniert die Erkennung der Kategorie pornografische Seiten, was sich an einem F_1 Measure von 54,2 % unter Verwendung aller Merkmale, erkennen lässt. Ebenfalls sehr gut funktioniert die Erkennung der Kategorie Captcha very, welches unter Verwendung aller Merkmale, ein F_1 Measure von 53,8 % erzielt.

Bei der Betrachtung der besten F_1 Measure für die einzelnen Merkmale fällt auf, dass kein Merkmal sich gut zur Erkennung von jeder Kategorie eignet.

Captcha Für die Klasse **very** (Tabelle 7.1) hat die Verwendung aller Merkmale ein F_1 Measure von 53,8 % erreicht, welches das höchste erzielte aller F_1 Measure ist. Daraus lässt sich schlussfolgern, dass unter Verwendung aller Merkmale die meisten als very klassifizierten Seiten erkannt werden.

Tabelle 7.1: Captcha Klassifikationsgenauigkeit gemessen an den Werten Precision(P), Recall(R) und F_1 (F-Measure) für die Klassen not (98,65 % der Seiten im gesamten Datensatz / 99,13 % im Testdatensatz), abit (0,60 % / 0,48 %) und very (0,75 % / 0,39 %)

Merkmals	not			abit			very		
	P	R	F_1	P	R	F_1	P	R	F_1
Helligkeit, RGB	99,2	96,9	98,0	00,0	00,0	00,0	02,7	07,7	04,0
id-class Attribut	99,5	98,3	98,9	06,8	18,8	10,0	30,8	61,5	41,0
src- Attribut	99,4	98,2	98,8	09,1	06,3	07,4	10,5	46,2	17,1
title-Tag	99,3	96,3	97,8	00,0	00,0	00,0	08,6	46,2	14,5
Webseiten Inhalt	99,2	97,6	98,4	00,0	00,0	00,0	06,5	23,1	10,2
Wort Anzahl	99,6	98,5	99,0	11,5	18,8	14,3	17,9	53,8	26,9
OpenCV	99,6	97,5	98,5	08,5	43,8	14,3	11,8	15,4	13,3
Alle Merkmale	99,6	99,5	99,6	21,1	25,0	22,9	53,8	53,8	53,8

Für das Einzelmerkmal id und class Attribut wurde das höchste F_1 Measure von 41,0 % erreicht. Für dieses Merkmal wurden auch die höchste Precision von 30,8 % und ein Recall von 61,5 % erzielt.

Für die Klasse **abit** hat die Verwendung aller Merkmale ein F_1 Measure von 22,9 % erreicht, welches ebenfalls das höchste erzielte aller F_1 Measure ist. Daraus lässt sich schlussfolgern, dass unter Verwendung aller Merkmale auch die meisten als abit klassifizierten Seiten erkannt werden.

Für die Einzelmerkmale Wort Anzahl und OpenCV wurde das höchste F_1 Measure von 14,3 % erreicht. Das Einzelmerkmal OpenCV erreicht eine hohen Recall von 43,8 %, allerdings ist die Precision mit 8,5 % sehr gering. Die geringe Precision bedeutet, dass der Anteil der Seiten die als Captcha erkannt wurden und tatsächlich Captchas sind, sehr gering ist.

Die beste Precision mit 11,5 % wurde für das Merkmal Wort Anzahl erzielt. Wenn man die Klassen **very** und **abit** miteinander vergleicht fällt auf, dass zwischen den F_1 Measure für alle Merkmale eine Differenz von 30,9 % liegt. Auch wenn man die beiden höchsten F_1 Measure der beiden Einzelmerkmale miteinander vergleicht, fällt auf, dass in der Klasse abit gerade mal ein F_1 Measure von 14,3 % erzielt wurde. Daraus lässt sich schlussfolgern, dass die Erkennung von Seiten die mit Captcha very klassifiziert wurden gut funktioniert, jedoch nicht für die Klasse abit.

Eine sehr geringe Precision von 2,7 % hat das Merkmal Helligkeit, RGB was überraschend ist, da wie man in Abbildung 5.3 sieht Seiten welche Captchas beinhalten deutlich heller sind, als Seiten ohne. Aufgrund dessen hätte man vermuten können, dass bessere Werte erzielt werden. Wie man allerdings anhand der Werte sehen kann, eignet Helligkeit und RGB Werte sich nicht als

Tabelle 7.2: Captcha Klassifikationsgenauigkeit gemessen an den Werten Precision(P), Recall(R) und F_1 (F-Measure) für die Klassen no (98,65 % der Seiten im gesamten Datensatz / 99,13 % im Testdatensatz) und yes (1,35 % / 0,87 %)

Merkmals	no			yes		
	P	R	F_1	P	R	F_1
Helligkeit, RGB	99,2	97,7	98,4	02,6	06,9	03,7
id-class Attribut	99,4	99,7	99,5	44,4	27,6	34,0
src- Attribut	99,3	98,3	98,8	09,7	20,7	13,2
title-Tag	99,3	96,3	97,8	06,1	27,6	10,0
Webseiten Inhalt	99,3	98,7	99,0	10,4	17,2	13,0
Wort Anzahl	99,7	98,2	98,9	23,1	62,1	33,6
OpenCV	99,6	97,6	98,6	17,0	55,2	26,0
Alle Merkmale	99,3	99,1	99,2	17,1	20,7	18,8

Merkmal zur Erkennung von Captchas.

Außerdem wäre Anhand Abbildung 5.8c zu erwarten gewesen, dass sich das Einzelmerkmal source Attribut gut zur Klassifizierung von Seiten mit Captchas eignet, der Recall liegt auch bei 46,2 %, aber die Precision beträgt nur 10,5 %, was in einem F_1 Measure von 17,1 % resultiert.

In der Zwei-Klassen-Klassifizierung hat die Verwendung aller Merkmale ein F_1 Measure von 18,8 % erreicht(siehe Tabelle 7.2), was eine Verschlechterung von 35 % im Vergleich zur Klasse very und 4,1 % schlechter im Vergleich zur Klasse abit ist.

Für die Einzelmerkmale id und class Attribute wird wieder das beste F_1 Measure erzielt, dies mal 34,0 %. Das Einzelmerkmal Wort Anzahl erreicht hier sogar ein Recall von 62,1 %, allerdings ist die Precision mit 23,1 % im Vergleich dazu sehr gering.

Wie man in Tabelle 7.1 erkennt, erreicht das Einzelmerkmal OpenCV in der Kategorie abit ein F_1 Measure von 14,3 % und für very 13,3 %, welches in der Zwei-Klassen-Klassifizierung in einem F_1 Measure von 33,6 % resultiert und damit deutlich besser funktioniert als in der Drei-Klassen-Klassifizierung.

Fehlerseiten Für die Klasse **very** (Tabelle 7.3) hat die Verwendung aller Merkmale ein F_1 Measure von 26,4 % erreicht, welches nicht das höchste aller erzielten F_1 Measure ist. Das heißt, dass ein Merkmal einzeln sich gut für den Trainingsdatensatz geeignet hat. Diese sind aber nur charakteristisch für den Trainingsdatensatz und nicht allgemein. Dadurch wird die Baumstruktur des Entscheidungsbaums komplexer und führt zu Overfitting.

Für das Einzelmerkmal title Tag wurde das höchste F_1 Measure von 39,4 %

Tabelle 7.3: Fehlerseiten Klassifikationsgenauigkeit gemessen an den Werten Precision(P), Recall(R) und F_1 (F-Measure) für die Klassen not (95,54 % der Seiten im gesamten Datensatz / 95,53 % im Testdatensatz), abit (0,83 % / 0,78 %) und very (3,63 % / 3,69 %)

Merkmal	not			abit			very		
	P	R	F_1	P	R	F_1	P	R	F_1
Helligkeit, RGB	96,3	89,9	93,0	00,0	00,0	00,0	12,9	24,4	16,9
id-class Attribut	96,9	91,3	94,0	00,0	00,0	00,0	19,2	44,7	26,8
src- Attribut	96,7	80,4	87,8	00,0	00,0	00,0	09,1	48,8	15,3
title-Tag	96,9	95,3	96,1	00,0	00,0	00,0	38,2	40,7	39,4
Webseiten Inhalt	97,1	88,1	92,4	00,0	00,0	00,0	13,8	44,7	21,1
Alle Merkmale	96,9	93,3	95,1	00,0	00,0	00,0	20,4	37,4	26,4

Tabelle 7.4: Fehlerseiten Klassifikationsgenauigkeit gemessen an den Werten Precision(P), Recall(R) und F_1 (F-Measure) für die Klassen no (95,54 % der Seiten im gesamten Datensatz / 95,53 % im Testdatensatz) und yes (4,46 % / 4,47 %)

Merkmal	no			yes		
	P	R	F_1	P	R	F_1
Helligkeit, RGB	96,1	94,1	95,1	13,3	19,5	15,8
id-class Attribut	97,0	94,0	95,5	23,4	38,9	29,2
src- Attribut	96,4	87,0	91,5	10,0	30,9	15,1
title-Tag	96,9	95,8	96,3	27,2	33,6	30,0
Webseiten Inhalt	97,5	84,3	90,4	13,7	53,0	21,8
Alle Merkmale	96,7	96,1	96,4	26,5	30,2	28,2

erreicht. Für dieses Merkmal wurden auch die höchste Precision von 38,2 % und einen Recall von 40,7 % erzielt. Dies entspricht den Erwartungen, dass sich der title Tag gut zur Klassifizierung von Fehlerseiten eignet, siehe Abbildung 5.8a.

Der höchste Recall erzielt das Einzelmerkmal source Attribut mit 48,8 %, jedoch ist die Precision mit 9,1 % sehr gering, was zu einem F_1 Measure von nur 15,3 % führt. Daraus lässt sich schlussfolgern, dass sich dieses Merkmal nicht zur Klassifizierung von Fehlerseiten eignet, da der Anteil der erkannten Seiten die tatsächlich Fehlerseiten sind, sehr gering ist.

Für die Klasse **abit** wurde kein Wert höher als 0 % erzielt, was zu einer Differenz in den F_1 Measures von 26,4 % führt. Daraus lässt sich schlussfolgern, dass keins der verwendeten Merkmale sich zur Erkennung der als Fehlerseite abit klassifizierten Seiten eignet.

Mit einem naiven Klassifizierer, der alle Seiten als abit klassifiziert, könnte

man ein F_1 Measure von 1,5 % erreichen.

Wenn man die F_1 Measure in der Zwei-Klassen-Klassifizierung betrachtet fällt auf, dass das F_1 Measure für alle Merkmale einen Wert von 28,2 % erreicht, was eine Verbesserung von 1,8 % ist.

Für das Einzelmerkmal title Tag wurde das höchste F_1 Measure von 30,0 % erreicht, welches eine Verschlechterung von 9,4 % im Vergleich mit der Klasse very ist. Also lässt sich daraus schlussfolgern, dass sich dieses Merkmal gut zur Klassifizierung von very Fehlerseiten, allerdings nicht für Fehlerseiten abit eignet.

Ähnlich wie für die Klasse very, erzielt das Einzelmerkmal Webseiten Inhalt einen hohen Recall von 53,0 %, aber die Precision beträgt nur 13,7 %.

Beim betrachten der F_1 Measures zwischen beiden Klassifizierungen fällt allerdings auf, dass für die gleichen Einzelmerkmale gute beziehungsweise schlechte Werte erzielt wurden.

Tabelle 7.5: PopUp Klassifikationsgenauigkeit gemessen an den Werten Precision(P), Recall(R) und F_1 (F-Measure) für die Klassen not (92,97 % der Seiten im gesamten Datensatz / 92,04 % im Testdatensatz), abit (3,15 % / 3,84 %) und very (3,88 % / 4,12 %)

Merkmal	not			abit			very		
	P	R	F_1	P	R	F_1	P	R	F_1
Helligkeit, RGB	95,8	87,6	91,5	16,1	31,3	21,2	21,9	44,5	29,3
id-class Attribut	92,6	91,9	92,3	06,0	07,0	06,5	10,4	10,2	10,3
src- Attribut	93,5	90,7	92,1	13,4	14,1	13,7	08,1	13,1	10,1
title-Tag	92,2	94,5	93,3	04,7	03,1	03,8	06,8	05,1	05,8
Webseiten Inhalt	92,6	90,5	91,5	06,9	06,3	06,6	03,7	05,8	04,5
Alle Merkmale	94,1	92,3	93,2	07,7	07,8	07,8	20,5	29,2	24,1

PopUp Für die Klasse **very** (Tabelle 7.5) hat die Verwendung aller Merkmale ein F_1 Measure von 24,1 % erreicht, welches nicht das höchste aller erzielten F_1 Measure ist. Das heißt, dass einzelne Merkmale sich gut für den Trainingsdatensatz geeignet haben, diese aber nur charakteristisch für den Trainingsdatensatz sind und nicht allgemein. Dadurch wird die Baumstruktur des Entscheidungsbaums komplexer und führt zu Overfitting.

Für das Einzelmerkmal Helligkeit, RGB wurde das höchste F_1 Measure von 29,3 % erreicht. Für dieses Merkmal wurden auch die höchste Precision von 21,9 % und einen Recall von 44,5 % erzielt.

Für die Klasse **abit** hat die Verwendung aller Merkmale ein F_1 Measure von 7,8 % erreicht, welches nicht das höchste aller erzielten F_1 Measure ist. Zudem

Tabelle 7.6: PopUp Klassifikationsgenauigkeit gemessen an den Werten Precision(P), Recall(R) und F_1 (F-Measure) für die Klassen no (92,97 % der Seiten im gesamten Datensatz / 92,04 % im Testdatensatz) und yes (7,03 % / 7,96 %)

Merkmal	no			yes		
	P	R	F_1	P	R	F_1
Helligkeit, RGB	95,1	89,4	92,1	27,4	46,4	34,5
id-class Attribut	92,8	92,9	92,8	16,8	16,6	16,7
src- Attribut	92,5	91,7	92,1	13,0	14,3	13,6
title-Tag	92,2	96,1	94,1	11,2	05,7	07,5
Webseiten Inhalt	92,3	92,0	92,2	10,9	11,3	11,1
Alle Merkmale	93,7	93,3	93,5	26,5	27,9	27,2

ist dies kaum eine Verbesserung zum naiven Klassifikator der in diesem Fall ein F_1 Measure von 7,4 % erzielen würde.

Für das Einzelmerkmal Helligkeit, RGB wurde das höchste F_1 Measure von 21,2 % erreicht. Für dieses Merkmal wurden auch die höchste Precision von 16,1 % und einen Recall von 31,3 % erzielt.

Auch hier fällt auf, dass die Klassifizierung von PopUp abit deutlich schlechter funktioniert als für die Klasse very, was sich einer Differenz des F_1 Measure von 16,3 % zeigt. Auch wenn man Precision und Recall mit einander vergleicht, sind diese für abit deutlich niedriger.

Ebenfalls erstaunlich ist das niedrige F_1 Measure für das Merkmal id und class Attribut. Es war zu vermuten, dass bessere Werte erzielt werden, da unter Verwendung des CSS Framework Bootstrap¹ oft sprechende Klassennamen verwendet werden, die Vermutung war, dass sich dieses Merkmal zur Erkennung von PopUps eignet.

Helligkeit und RGB erreichen zwar gute Werte für die Klassifizierung von PopUps, allerdings liefern diese keine Information darüber ob die Seite tatsächlich ein PopUp beinhaltet. PopUps welche dunkle Overlays haben, verursachen einen niedrigen Helligkeitswert, welches ein Hinweis dafür sein kann ob ein PopUp auf der Seite ist. Allerdings reicht dieses Merkmal allein nicht aus, da auch Seiten die beispielsweise schwarz sind und kein PopUp beinhalten, einen niedrigen Helligkeitswert haben.

Wenn man die F_1 Measure in der Zwei-Klassen-Klassifizierung betrachtet fällt auf, dass das F_1 Measure für alle Merkmale einen Wert von 27,2 % erreicht, was eine Verbesserung von 3,1% ist.

Das Einzelmerkmal Helligkeit und RGB hat wie in der Drei-Klassen-Klassifizierung das beste F_1 Measure erreicht. Auch wenn man die F_1 Measure der anderen

¹<https://getbootstrap.com/> (abgerufen am 11.03.2019)

Einzelmerkmale zwischen den Klassen vergleicht fallen keine großen Differenzen auf.

Tabelle 7.7: Pornografie Klassifikationsgenauigkeit gemessen an den Werten Precision(P), Recall(R) und F_1 (F-Measure) für die Klassen no (95,92 % der Seiten im gesamten Datensatz / 96,52 % im Testdatensatz) und yes (4,09 % / 3,48 %)

Merkmal	no			yes		
	P	R	F_1	P	R	F_1
Helligkeit, RGB	97,8	94,0	95,8	19,6	40,5	26,4
id-class Attribut	97,7	97,2	97,5	32,6	37,9	35,1
src- Attribut	97,9	98,4	98,2	48,5	41,4	44,7
title-Tag	98,0	98,2	98,1	47,3	45,7	46,5
Webseiten Inhalt	98,3	97,7	98,0	45,6	53,4	49,2
Alle Merkmale	98,3	98,5	98,4	56,0	52,6	54,2

pornografischer Inhalt Wie oben bereits erwähnt hat die Klassifizierung von pornografischen Seiten (Tabelle 7.7) im Vergleich zu den anderen Kategorien am Besten funktioniert, welches sich an einem F_1 Measure von 54,2 % für alle Merkmale erkennen lässt, welches auch das höchste erzielte aller F_1 Measure ist. Das heißt unter Verwendung aller Merkmale kann man die meisten pornografischen Seiten erkennen.

Für das Einzelmerkmal Webseiten Inhalt wurde das höchste F_1 Measure von 49,2 % erreicht, welches sich aus einer Precision von 45,6% und eine, Recall von 53,4 % ergibt. Ebenfalls gute F_1 Measure werden für die Einzelmerkmale title Tag mit 46,5 % und das source Attribut mit 44,7 % erzielt.

Wenn man die Precision und den Recall für jedes Einzelmerkmal betrachtet, fällt zu dem auf, dass es keine großen Differenzen zwischen den beiden Werten gibt und jedes Merkmal, im Vergleich zu anderen Kategorien, gute Werte erzielt.

Zur Klassifizierung der Kategorie Pornografie eignen sich am besten die source und title Tag, sowie der Webseiten Inhalt. Dies entspricht den Erwartungen wenn man zum Beispiel Abbildung 5.8b betrachtet.

Kapitel 8

Fazit

Das Ziel der Arbeit war es herauszufinden wie gut die automatische Klassifizierung von Webseiten funktioniert und welche Merkmale sich zur Erkennung problematischer Seiten eignen. Die Erkenntnisse aus dem Klassifizierungsexperiment mit 10000 Seiten sollten Aufschluss darüber geben.

Folgende Merkmale haben sich gut für die Kategorien geeignet:

Für die Kategorie Captcha haben sich die Merkmale id und classen Attribut und Wortanzahl gut geeignet. Das höchste F_1 Measure wird für das Merkmal id und classen Attribut mit einem Wert von 41,0 % in der Klasse very erzielt. Für die Kategorie Fehlerseiten haben sich die Merkmale title Tag, id und classen Attribute sowie der Webseiten Inhalt gut geeignet. Das höchste F_1 Measure wurde dabei für das Merkmal title Tag mit einem Wert von 39,4 % in der Klasse very erzielt.

Für die Kategorie PopUp hat sich das Merkmal Helligkeit und RGB am besten geeignet. Das höchste F_1 Measure wird dabei in der Zwei-Klassen-Klassifizierung mit einem Wert von 34,5 % erzielt.

Für die Kategorie Pornografie waren die Merkmale Webseiten Inhalt, title Tag, source Attribut und id und class Attribut gut geeignet. Das höchste F_1 Measure wird für das Merkmal Webseiten Inhalt mit 49,2 % erzielt. Wie man in Kapitel 7 sieht führt die Zusammenfassung der Klassen abit und very zu keiner signifikanten Verbesserung der Klassifizierung. In der Kategorie Captcha wurde die Klassifizierung um 35% schlechter und in den Kategorien Fehlerseiten und PopUp wurden nur minimale Verbesserungen von wenigen Prozent erzielt. Von daher ist es besser die Kategorien in ihren ursprünglichen Kategorien zu klassifizieren.

Zudem fällt auf, dass es kein Merkmal gibt welches jede Kategorie gut differenziert. Die Ergebnisse in Kapitel 7 zeigen ebenfalls, dass nicht immer die Merkmale bei denen es zu erwarten war gut für die jeweilige Kategorie funktioniert haben.

Zu erwarten wäre gewesen, dass sich für die Kategorie Captcha die Einzelmerkmale source Attribut, OpenCV, sowie Wort Anzahl und Helligkeit gut eignen. Die Ergebnisse in Tabelle 7.1 zeigen allerdings, dass diese keine hohen F_1 Measure erzielen. Die Wort Anzahl und OpenCV erreichen nur ein F_1 Measure von 14,3 % für abit und 26,9 % für die Wort Anzahl und 13,3 % OpenCV für very. Die niedrigen F_1 Measure für OpenCV könnten darauf zurück geführt werden, dass im Testdatensatz viele Seiten gelandet sind, die zwar Captchas haben, aber keins welches durch das Feature Matching mit SIFT abgedeckt wurden. Denn durch OpenCV wurden nur zwei Arten von Captchas erkannt, wie man in Abbildung 5.10a und Abbildung 5.10b erkennen kann. Der Datensatz enthält allerdings noch weitere Arten von Captchas siehe Abbildung 5.11. Das Modell kann also nicht richtig trainiert werden.

Bei dem source Attribut gibt es ein ähnliches Problem. Es sind auch viele als Captcha klassifizierte Seiten dabei die nicht diese URL einbinden.

Beide Merkmale würden vermutlich besser auf einem Datensatz funktionieren welcher nur Google Captchas beinhaltet. Man benötigt noch weitere Merkmale um auch Captchas wie in Abbildung 5.11 erkennen zu können.

In der Kategorie Fehlerseiten abit ist das Problem, dass viele Seiten davon „Plugin not supported“ oder ähnliches als Fehlermeldung haben und diese Fehlermeldungen wurden durch die Merkmale nicht abgedeckt. Um auch diese Seiten mit abzudecken benötigt man Merkmale die auch diese Fehlermeldungen mit einbeziehen.

In der Kategorie PopUp wäre zu erwarten gewesen, dass sich das Einzelmerkmal id und classen Attribut gut eignet. Wie man jedoch anhand der niedrigen F_1 Measures erkennet eignet es sich jedenfalls nicht für diesen Datensatz. Interessant wäre zu sehen ob sich die Werte, wenn man die Versuche auf einem größeren Datensatz durchführt, verbessern würden. Bei PopUps und der Helligkeit ist das Problem, dass die Helligkeit kein wirklich spezifisches Merkmal für PopUps ist, sondern auch auf andere Seiten zutrifft, wie zum Beispiel Seiten mit pornografischen Inhalt. Diese werden dann als PopUp erkannt, obwohl sie keins sind, was zu False Positives führt.

Anhand der hohen Precision und F_1 Measures könnte man schlussfolgern, dass die Erkennung von mit no beziehungsweise not klassifizierten Seiten, also nicht problematischen Seiten, besser funktioniert. Allerdings erreicht selbst ein naiver Klassifikator für alle Merkmale in allen Kategorien, Precision und F_1 Measures über 90 % auf Grund der hohen Anzahl der nicht problematischen Seiten im Datensatz.

Wenn man im Vergleich dazu die Werte betrachtet die ein naiver Klassifikator, für alle Merkmale, mit für abit oder very klassifizierten Seiten erreicht, so werden in den einzelnen Kategorien keine F_1 Measures über 10 % erzielt. Wenn man das mit den F_1 Measures vergleicht die hier erzielt werden, siehe die Ta-

bellen in Kapitel 7, ist dies in den meisten Fällen eine deutliche Verbesserung gegenüber dem naiven Klassifikator.

Die Testdatensätze bestanden jeweils aus 3330 Seiten, aber vor allem in der Kategorie Captcha waren nur 29 Seiten (16 davon abit, 13 davon very) die als solches klassifiziert wurden, was nur 0.87% ausmacht. Fehlerseiten machen 4,47 % aus, PopUps 7,96 % und pornografische Seiten 3,48 %. Wie man sieht ist der Anteil der problematischen Seiten in den Testdatensätzen sehr gering. Der Datensatz wurde zuerst durch die Worker von Mechanical Turk annotiert und dann nochmal von Hand kontrolliert. Es ist anzunehmen, dass dadurch die meisten Seiten korrekt klassifiziert wurden. Allerdings kann es trotzdem vorkommen, dass einige der Seiten im Datensatz falsch klassifiziert wurden und dadurch die Ergebnisse beeinflusst wurden. So wurden zum Beispiel anhand der Ergebnisse des Feature Matching mit OpenCV noch 8 Captchas gefunden, welche nicht durch die Worker erkannt wurden und auch in der zweiten Kontrolle mit Hand übersehen wurden. Somit konnte der Datensatz mit Hilfe der Arbeit verbessert werden.

Kapitel 9

Future Work

Dieses Kapitel soll zeigen welche Fragen noch offen geblieben sind und welche neuen Fragen sich ergeben haben. In dem Hauptexperiment in Kapitel 7 konnte gezeigt werden, dass die Erkennung der Kategorie Pornografie mit einem F_1 Measure von 54,2 % und der Kategorie Captcha very mit 53,8 % ganz gut funktioniert. Die anderen Kategorien, Captcha abit, sowie Fehlerseiten und PopUps konnten noch nicht ausreichend erkannt werden.

Es konnte gezeigt werden, dass eine automatische Erkennung von Webseiten möglich ist. Allerdings ist die Erkennung, zu diesem Zeitpunkt, noch nicht exakt genug um sie in der Praxis anzuwenden, da zu viele problematische Seiten unerkannt bleiben würden. Deswegen sind noch weitere Arbeiten erforderlich. Diese wären: charakteristische Merkmale zur Erkennung von PopUps finden, Merkmale finden die weitere Fehlermeldungen wie „Plugin not supported“ mit einbeziehen und die Erkennung von Captchas wie in Abbildung 5.11. Wie man beispielsweise anhand von Tabelle 7.3 erkennen kann, sind die Ergebnisse auch von Overfitting beeinflusst. Mitchell et al. [12] schreibt, dass Overfitting die Richtigkeit um 10-25 % beeinflussen kann. Zukünftige Arbeiten sollten dieses Problem beheben, beispielsweise durch Post Pruning des Entscheidungsbaums und unter Verwendung eines größeren Datensatzes. Ein anderer Ansatz wäre die Verwendung von Cross-Validation¹. Zudem könnte man für die einzelnen Kategorien auch nur die Merkmale zur Klassifizierung verwenden die sich gut geeignet haben, siehe Kapitel 7.

In zukünftigen Arbeiten könnte man auch noch weitere Content Error, wie zum Beispiel Seiten die abgeschnitten sind, nicht richtig geladen haben oder nur zum Großteil aus Werbung bestehen mit einbeziehen.

Ohne automatische Erkennung von problematischen Webseiten riskieren wir unvollständige Archive, ohne zu wissen dass sie unvollständig sind. Im Rah-

¹Dadurch wird das Trainingsset in weitere k-Subsets aufgeteilt und der Algorithmus trainiert iterativ auf diesen k-1 Subsets und verwendet die restlichen Subsets als Testset

men dieser Arbeit wurde gezeigt wie man diese Problematik beseitigen kann und welche Merkmale sich zur automatischen Erkennung von problematischen Webseiten eignen.

Literaturverzeichnis

- [1] Yasmin AlNoamany, Michele C. Weigle, and Michael L. Nelson. Detecting off-topic pages within timemaps in web archives. *International Journal of Digital Libraries (IJDL)*, 17(3):203–221, July 2016. doi: 10.1007/s00799-016-0183-5. 2.1
- [2] Ziv Bar-Yossef, Andrei Z. Broder, Ravi Kumar, and Andrew Tomkins. Sic transit gloria telae: Towards an understanding of the web’s decay. In *Proceedings of the 13th International Conference on World Wide Web*, WWW ’04, pages 328–337, New York, NY, USA, 2004. ACM. ISBN 1-58113-844-X. doi: 10.1145/988672.988716. URL <http://doi.acm.org/10.1145/988672.988716>. 2.3
- [3] Justin F. Brunelle, Mat Kelly, Michele C. Weigle, and Michael L. Nelson. The impact of javascript on archivability. *International Journal of Digital Libraries (IJDL)*, January 2015. doi: 10.1007/s00799-015-0140-8. 2.1
- [4] Miklós Erdélyi, András A. Benczúr, Julien Masanés, and Dávid Siklósi. Web spam filtering in internet archives. In *Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web*, AIRWeb ’09, pages 17–20, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-438-6. doi: 10.1145/1531914.1531918. URL <http://doi.acm.org/10.1145/1531914.1531918>. 2.1
- [5] Mat Kelly, Michael L Nelson, and Michele C Weigle. The archival acid test: evaluating archive performance on advanced html and javascript. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 25–28. IEEE Press, 2014. 2.1
- [6] Johannes Kiesel, Benno Stein, and Stefan Lucks. A Large-scale Analysis of the Mnemonic Password Advice. In *24th Annual Network and Distributed System Security Symposium (NDSS 2017)*. Association for Computational Linguistics, February 2017. doi: 10.14722/ndss.2017.23077. 5.2

- [7] Johannes Kiesel, Florian Kneist, Milad Alshomary, Benno Stein, Matthias Hagen, and Martin Potthast. Reproducible Web Corpora: Interactive Archiving with Automatic Quality Assessment. *Journal of Data and Information Quality (JDIQ)*, 10(4):17:1–17:25, October 2018. doi: 10.1145/3239574. URL <https://dl.acm.org/authorize?N676358>. 1
- [8] Johannes Kiesel, Fabienne Hubricht, Benno Stein, and Martin Potthast. A Dataset for Content Error Detection in Web Archives. In *18th ACM/IEEE on Joint Conference on Digital Libraries (JCDL 2019)*. ACM, June 2019. 4, 4.2
- [9] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. doi: 10.1023/B:VISI.0000029664.99615.94. URL <https://doi.org/10.1023/B:VISI.0000029664.99615.94>. 5.5
- [10] Jorge A Marcial-Basilio, Gualberto Aguilar-Torres, Gabriel Sánchez-Pérez, L Karina Toscano-Medina, and Héctor M Pérez-Meana. Detection of pornographic digital images. *International journal of computers*, 5(2):298–305, 2011. 2.4
- [11] Luis Meneses, Richard Furuta, and Frank Shipman. Identifying “soft 404” error pages: Analyzing the lexical signatures of documents in distributed collections. In Panayiotis Zaphiris, George Buchanan, Edie Rasmussen, and Fernando Loizides, editors, *Theory and Practice of Digital Libraries*, pages 197–208, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-33290-6. 2.3
- [12] Tom Mitchell, Bruce Buchanan, Gerald DeJong, Thomas Dietterich, Paul Rosenbloom, and Alex Waibel. Machine learning. *Annual review of computer science*, 4(1):417–433, 1990. 9
- [13] Víctor M Prieto, Manuel Álvarez, and Fidel Cacheda. Soft-404 pages, a crawling problem. *JDIM*, 12(2):73–92, 2014. 2.3
- [14] Robert Sanderson, Mark Phillips, and Herbert Van de Sompel. Analyzing the persistence of referenced web resources with memento. *arXiv preprint arXiv:1105.3459*, 2011. 2.3
- [15] Srinivasan H. Sengamedu, Subhajit Sanyal, and Sriram Satish. Detection of pornographic content in internet images. In *Proceedings of the 19th ACM International Conference on Multimedia*, MM ’11, pages 1141–1144, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0616-4. doi: 10.

- 1145/2072298.2071959. URL <http://doi.acm.org/10.1145/2072298.2071959>. 2.4
- [16] Suphanee Sivakorn, Iasonas Polakis, and Angelos D. Keromytis. I am robot: (deep) learning to break semantic image captchas. In *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 388–403, March 2016. doi: 10.1109/EuroSP.2016.37. 2.2
 - [17] Marc Spaniol, Dimitar Denev, Arturas Mazeika, Gerhard Weikum, and Pierre Senellart. Data quality in web archiving. pages 19–26, 01 2009. doi: 10.1145/1526993.1526999. 2.1
 - [18] Fabian Stark, Caner Hazirbas, Rudolph Triebel, and Daniel Cremers. Captcha recognition with active deep learning. In *GCPR Workshop on New Challenges in Neural Computation*, volume 10, 2015. 2.2