Generierung paralleler (Bild/Text-)Korpora mittels fokussiertem Webcrawling

Bachelor-Thesis von Dennis Werner
Tag der Einreichung:

1. Gutachten: Prof. Dr. Chris Biemann
2. Gutachten: Steffen Remus

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Generierung paralleler (Bild/Text-)Korpora mittels fokussiertem Webcrawling

Vorgelegte Bachelor-Thesis von Dennis Werner

1. Gutachten: Prof. Dr. Chris Biemann
2. Gutachten: Steffen Remus

Tag der Einreichung:

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 1. September 2015

(D. Werner)

Zusammenfassung

Um automatisiert mit Texten zu arbeiten, sind Korpora ein wichtiges Hilfsmittel. Darüber hinaus sind parallele Korpora wie Image-net [Deng et al. 2009] für viele Einsatzgebiete nutzbar. Um eine mögliche Generierungsmethode oder Vergrößerungsmethode eines Bild-Text-Korpus zu prüfen, wird in dieser Thesis ein fokussierter Webcrawler zur Erstellung paralleler Korpora aus Bild und zugehörigem Text geschrieben. Als Ausgangsbasis wird der Webcrawler Heritrix verwendet und um eigene Module erweitert. Es wird dazu ein Modul zur Textextraktion, Linkbewertung und Bildextraktion geschrieben. Der Crawler wird mit einem Sprachmodell auf 5-gramm Basis nach Kneser Ney trainiert und nutzt Perplexität als Textbewertungsmaß. Dazu wird ein kleines Ausgangskorpus anhand themenrelevanter Seiten erstellt. Zusätzlich wird der Crawler um eine Keywordgewichtung erweitert. Zur Ergebnissbewertung kommt als Baseline ein unfokussierter Crawlinglauf- sowie zu weiteren Vergleichszwecken ein fokussierter Lauf zum Einsatz. Ziel ist es, den Crawler mit einem Seed von themenrelevanten Seiten zu starten und möglichst viele Bilder zu erhalten, die thematisch zu dem kleinen Ausgangskorpus und somit auch zum Seed passen. Die so erhaltenen Bilder werden über Crowdsourcing im kleinem Rahmen annotiert und bewertet. Am Ende der Arbeit liefert der keywordbasierte Lauf von 2500 annotierten Bildern gut 414 korpustaugliche Bild-Text Kombinationen sowie 1074 zum Thema passende Kombinationen. In puncto themenrelevanter Bildbeschaffung schlägt der keywordbasierte Ansatz die beiden Baselinemethoden um den Faktor 5,7 (unfokussiert) bzw. 3,45 (fokussiert).

Abstract

Corpora are useful resources in natural language processing. Application scenarios for parallel corpora such as Image-net [Deng et al. 2009] are widespread. A focused crawler is developed for generating or extending an image-text corpus. Apache Heritrix is used as base and is extended with a self-written module. This module covers text-, link- and pictureextraction. To train the crawler, a 5-Gram Kneser Ney language model is used. Perplexity serves as a measure with respect to the trained text domain. Furthermore, the crawler is enhanced with a keyword-based weighting scheme. For evaluation purposes an unfocused crawl and a focused crawl, based on methods by Remus and Biemann (unpublished), are used as a baseline. The aim is to establish a focus based crawl by using domain related websites as a starting point for finding as many images as possible, that fit the selected domain. Afterwards the crawled pictures and their captions are annotated by a small-scale crowdsourcing. In the end the keyword based approach provides 414 text-picture combinations as a parallel corpus and 1074 domain related combinations out of 2500 annotated combinations. In terms of domain related collection of pictures, the keyword based approach outperforms both of the baseline methods by a factor of 5,7 (unfocused) and 3,45 (focused).

Inhaltsverzeichnis

1 Einleitung

1.1 Was sind parallele Korpora?

Möchte man automatisiert mit Sprache arbeiten, ist der Gebrauch von Korpora ein wertvolles Hilfsmittel. Ein Korpus ist eine Sammlung von Texten, *"deren Texte nach nachvollziehbaren Kriterien ausgewählt sind. Seit den sechziger Jahren gibt es nun schon maschinenlesbare Korpora als empirische Datensammlung für die sprachwissenschaftliche Arbeit"* [Teubert 1998, S. 131]. Teubert führt weiterhin aus, dass die ersten beiden Jahrzehnte der Korpuslinguistik von der Suche nach dem sog. repräsentativen Korpus geprägt waren, man jedoch zur Einsicht gekommen sei, dass ein Korpus für die Gesamtheit einer Sprache nicht realisierbar sei. Bei maschinellem Lernen wird daher auch oft ein einfacheres bzw. eingegrenzteres Korpus genutzt, um einer Maschine eine gewisse Entscheidungskraft zu verleihen. Ein Korpus kann beispielsweise aus gesammelten Zeitungsartikeln bestehen, deren Sätze von Menschen per Hand annotiert wurden, indem jedes Wort in Kategorien wie Verb, Nomen, Pronomen usw. eingeordnet wurde. Ein solches Korpus bietet die Möglichkeit, eine Maschine mit diesen zusätzlichen, den Worten zugeordneten Informationen auf ein bestimmtes Verhalten bzw. Entscheidungsmuster zu trainieren. Ein aus dem Alltag bekanntes Beispiel sind Spamfilter, die den E-Maileingang auf unnütze Massenwerbesendungen untersuchen. Einem Korpus entsprächen hier die bereits vom Benutzer annotierten, sprich einkategorisierten E-Mails in Spam und nicht-Spam. Diese Form von Korpus ist jedoch auf ihre Domäne beschränkt, d.h. der Spamfilter wird nur zufriedenstellend funktionieren, wenn die annotierten E-Mails auch in der Sprache vorliegen, wie die eingehenden E-Mails.

Ein paralleles Korpus hingegen schlägt eine Brücke in eine andere Domäne. Blickt man etwas zurück, stellt man fest, dass parallele Korpora bereits seit dem Altertum bekannt sind. Als Beispiel kann hier eine mehrsprachige Bibel dienen, welche den Text in hebräisch, Latein und griechisch vorhält. Die Bibel ist gleichzeitig auch einer der wichtigsten parallelen Texte, da sie in ca. 320 Sprachen übersetzt wurde, das neue Testament sogar in ca. 900. Eine Struktur zur Orientierung, die Vergleichbarkeit ermöglicht, ist in der Bibel bereits mit Kapiteln und Versen vorgegeben [Borin 2002; Trosterud 2002]. Zweck eines solchen parallelen Korpus ist meistens die Möglichkeit der Übersetzung zwischen zwei oder mehr Domänen bzw. die domänenübergreifende Darstellung des selben Inhalts. Der gebräuchlichste Fall ist sicherlich Sprache, wie die bereits erwähnte Bibel, die mehrere Übersetzungen nebeneinander für den gleichen Satz enthält. Ein paralleles Korpus muss jedoch nicht polyglott oder bilingual sein. Selbst ein kleines Wörterbuch mit Übersetzungen stellt in diesem Sinne ein oft gebräuchliches paralleles, bilinguales und bidirektionales Korpus dar. Bezogen auf die Informatik ergibt sich der Nutzen eines parallelen Korpus beim Trainieren und Evaluieren z.B. einer Übersetzungsmaschine. Je nach Algorithmus, muss die Maschine erst lernen, welches Objekt einem anderen zugeordnet ist. Ein naheliegendes Beispiel wäre ein Text über Äpfel. Erst wenn der Algoritmus die Zuordnung von "apple" zu "Apfel" in der Sprache anhand der Trainingsdaten gelernt hat, kann er auf den Testdaten eine solche Übersetzung wiedergeben. Doch um zu einem konkreten Beispiel für ein bekanntes paralleles Korpus zu kommen, sei "Europarl" von Koehn (2005) genannt. Europarl wurde per Crawling gesammelt und stellt Übersetzungen von Reden im Europaparlament bereit. Die so aufbereiteten Daten bieten eine einfache Möglichkeit den selben Satz in mehreren Sprachen zu beziehen und etwaige Übersetzungen zu vergleichen (s. Tabelle 1.1). Anhand dieses Korpus trainierte Koehn eine Maschine zur Übersetzung auf statistischer Basis und führte Versuche zur Hin- und Rückübersetzung eines Textes von einer Sprache in die Andere durch.

Analog verhält es sich in anderen Bereichen wie z.B. Computer Vision. Erst wenn der Maschine ein ballförmiges, rotes Stück Obst als "Apfel" vielfach bekannt gemacht wird, ist sie in der Lage ein ähnliches Objekt auch als Apfel zu identifizieren. Dafür benötigt es viele Beispiele, sodass die Maschine lernt, dass z.B. eine Tischdecke unter dem Apfel kein entscheidendes Merkmal für die Erkennung eines Apfels ist,

stattdessen aber möglicherweise die Form und die Farbe. Die Transition findet hier jedoch nicht zwischen zwei Sprachen statt, sondern zwischen Bild- und Text. Ein paralleles Korpus ist daher nicht nur für Sprachtechnologie und Übersetzungen von Text zu Text interessant, es ist auch ein wichtiges Mittel, um Maschinen zu trainieren, welche die Transition zwischen Bild und Text ermöglichen sollen.

Tabelle 1.1.: Beispiel Europarl (Auszug aus Koehn (2005, S.82))

Dänisch	det er næsten en personlig rekord for mig dette eftear
Deutsch	das ist für mich fast persönlicher rekord in diesem herbst.
Englisch	that is almost a personal record for me this autumn!
Finnisch	se on melkein minun ennatykseni tänä syksynä!
Französisch	c' est pratiquement un record personnel pour moi, cet automne!
Niederländisch	dit is haast een persoonlijk record deze herfst.
Portugiesisch	e quase o meu recorde pessoal deste semestre!
Schwedisch	det är nästan personligt rekord for mig denna höst!

1.2 Fokussiertes Webcrawling

Sucht man im Internet nach bestimmten Informationen, ist meist eine Suchmaschine das Mittel der Wahl. Gegenwärtig ist in Deutschland Google der dominante Anbieter was Informationsbeschaffung- und Verarbeitung online angeht. Doch auch der derzeit größte Suchmaschinenanbieter ist darauf angewiesen, die vorgehaltenen Informationen, Texte und Seiten aktuell zu halten und stets Neue zu entdecken. Damit der Suchmaschinenbetreiber viele Suchergebnisse vorhalten kann, werden Seiten automatisiert von einer Software besucht und indiziert. Diese Software wird allgemein auch als Webcrawler bezeichnet und der Indiziervorgang als Crawl. Die Software gibt sich, sofern der Betreiber die Daten freigibt und nicht verschleiert, auch gegenüber dem Webserver als solche zu erkennen. Eine solche Software meldet sich selbst bei den Webservern als Bot, sodass man umgangsprachlich sagt, dass ein Bot die Seite besucht hat. Der "Googlebot", als Crawlingagent von Google, ist zum aktuellen Zeitpunkt der Bekannteste. Ein solcher Bot extrahiert die nötigen Informationen aus der Seite, die anschließend vom Suchmaschinenbetreiber verarbeitet werden müssen, um die neu oder wiederholt eingelesene Seite in den zutreffenden Themenkategorien einzusortieren. Page et al. (1999) haben dazu den Informationsgehalt von eingehenden und ausgehenden Links der Webseiten mit dem bekannten PageRank genutzt um ihre Suchmaschine automatisiert eine Rangfolge generieren zu lassen und damit gezielter relevante Ergebnisse geliefert als andere. Wie wir heute wissen, war diese automatisiert crawlende und rankende Suchmaschine letztendlich schneller und effizienter als Verzeichnisdienste und Suchmaschinen mit menschlichen Redakteuren dabei, das gegen Ende der neunziger Jahre stark wachsende Internet zu erfassen [Menczer et al. 2004; Chakrabarti et al. 1999]. Namen wie "AltaVista", "FireBall" oder "Yahoo" sind heute nicht mehr im ursprünglichen Sinne existent oder spielen eine untergeordnete Rolle beim Suchen im Internet. Diese automatisierte linkbasierte Gewichtung von relevanten Seiten macht es bei der Suche jedoch schwer, kleine Seiten zu finden, die wenige eingehende Links haben. Durch die enorme Anzahl der im Internet erreichbaren Seiten bleibt es weiterhin eine Ressourcen bindende Herausforderung, neue und relevante Seiten zu finden. An dieser Stelle setzen fokussierte Webcrawler an, auch bekannt als "Topical Web Crawlers" [Menczer et al. 2004]. Ein fokussierter Crawler versucht die Ressourcen effizienter zu nutzen, um Seiten zu einem Thema zu finden. Zum Vergleich: der allgemeine Crawl-Ansatz analysiert zunächst Seiten und weist im Anschluss den Seiten die Suchworte zu und sortiert die passenden Ranglisten. Im Gegenensatz dazu versucht der fokussierte Crawler nicht jedem Link auf einer Seite zu folgen, sondern zu ermitteln ob es im Sinne des Themas lohnenswert ist, dem Link zu folgen. Hier tritt bereits ein erstes Problem zu Tage: Möchte man zu einem bestimmten Thema Inhalte suchen, ist man zunächst einmal auf eine Suchmaschine angewiesen, welche zum Thema passende Seiten vorhält. Eine Seite, die

oben in den Suchergebnissen steht, hat dies vornehmlich durch eingehende Links und eine geschickte Anpassung der Textinhalte an die Sortierstrategie erreicht. Daraus folgt, dass zur Relevanz maßgeblich die Verlinkung beigetragen hat, der Bot die Seite jedoch nicht wegen des Themas indizierte, sondern das Thema nachträglich erkannt- und die Seite den Kategorien zugewiesen wurde.

Wie ein solcher allgemeiner Crawler genau agiert, bleibt zumindest bei Google ein Geheimnis, da die Veröffentlichungen wie ”The Anatomy of a Large-Scale Hypertextual Web Search Engine” von Brin und Page (1998) zum Zeitpunkt der Thesis bereits 16 Jahre zurückliegen. In den Anfängen der Crawler wurden zunächst Breitensuchecrawls von Pinkerton (1994) und Tiefensuchecrawls von De Bra und Post (1994a) verwendet, um eine im Vergleich zu heutigen Verhältnissen beschauliche Anzahl Internetseiten zu Crawlen. In Anbetracht der heutigen Größe des Internets sind diese Strategien jedoch überholt, sodass man bereits Ende der neunziger Jahre nach effizienteren Methoden sucht, relevante Inhalte im Netz zu finden und zu indizieren. Einen solchen Ansatzpunkt stellt ein fokussierter Crawler dar. Ein fokussierter Crawler wird auf ein Themengebiet angelernt und versucht möglichst stets den Links zu folgen, die am vielversprechendsten auf themenrelevante Seiten verweisen. Dazu werden verschiedene Entscheidungsmechanismen bemüht, um die Warteschlange der noch zu besuchenden Links effizient zu sortieren. Effizient heißt in dem Fall, dass Links mit dem größten Potential, themenbezogene Inhalte zu liefern, als erstes angelaufen werden sollen. Es können dabei verschiedene Ansätze zur Entscheidungsfindung genutzt und kombiniert werden. Zum Beispiel könnte man mittels Machine Learning angelernte Seiten lexikalisch auf Relevanz testen, oder über mathematische Modelle. Eine Auswahl findet sich im Kaptiel ”Verwandte Arbeiten”. Der mathematisch statistische Ansatz soll als Grundlage innerhalb dieser Thesis zum Einsatz kommen.

1.3 Motivation/Forschungsfrage

Da es ressourcen- und kostenintensiv im Sinne von menschlichen Arbeitstunden sein kann, ein annotiertes Text-Bild-Korpus zu erstellen, wird versucht einen Zeitgewinn und Kostenvorteil zu erzielen, indem dieser Prozess ganz oder zum Teil automatisiert wird. Dazu sollen die Vorteile eines fokussierten Crawls genutzt werden. Zum einen wird per fokussiertem Crawling Bandbreite und Speicherplatz gespart, zum anderen findet eine automatisierte Vorselektion der Bilder statt. Auf diese Weise wird die Menge der den menschlichen Annotatoren vorzulegenden Bilder reduziert, was letztlich den Vorgang beschleunigt und günstiger macht. Dabei soll versucht werden, so wenig Sprachabhängigkeit wie möglich in den Versuchsaufbau des Crawls einfließen zu lassen. D.h. es wird auf syntaktische und grammatikalische Analyse verzichtet, Tools wie NLTK[1] [Bird 2006] bleiben außen vor. Die eigentliche Forschungsfrage stellt sich dahingehend, ob es möglich ist, Bilder aus einem gewählten Themenbereich aus dem Internet zu extrahieren, ohne dabei Vorwissen wie z.B. positiv oder negativ gelabelte Daten einzusetzen. Als Konsequenz wird so auf den Einsatz von bereits trainierten Klassifizierern z.B. aus dem Computer Vision Bereich verzichtet. Es wird versucht, die benötigten Informationen lediglich aus dem HTML-Code und dem Umfeld des Bildes zu erhalten, sodass hauptsächlich N-Gramme, daraus resultierende Perplexitätswerte und Keywords als Bewertungskriterien genutzt werden können. Die Perplexitätswerte werden aus einem Modell zum Smoothing nach Kneser Ney mittels 5-Grammen gewonnen.

Bildlich gesprochen stellt sich die Frage, ob es möglich ist, mit einem ”blinden” Crawler, der nicht in der Lage ist, ein Bild anhand der optischen Informationen zu klassifizieren, themenrelevante Bilder unter Zuhilfenahme von fokussiertem Crawling zu extrahieren und inwieweit eine vermutete Verbesserung gegenüber einem unfokussierten Crawl ausfällt. Zur Evaluation wird versucht, anhand von Perplexitätskurven und im Nachgang durch menschliche Annotation, den Gütegrad der exrahierten Bilder zu bestimmen. An dieser Stelle soll so der Frage nachgegangen werden, inwieweit die Fokussierung des Crawls Einfluss auf die extrahierten Bilder hat. Desweiteren soll geklärt werden, wie sehr die Erweiterung

[1] `http://www.nltk.org` (aufgerufen August 2015)

des fokussierten Crawls durch statistische Keywordgenerierung und Gewichtung bei der Linkextraktion ebenfalls Einfluss auf die extrahierten Bilder nimmt.

2 Verwandte Arbeiten

Crawling allgemein ist seit der frühen Mitte der Neunziger Jahre ein sehr aktives Thema. 1994 wurde Brian Pinkerton mit seinem Breitensuche verwendenden Crawl aktiv, was gleichzeitig auch die Klasse dieser Crawler darstellt, indem eine ”Breadth-First”-Strategie genutzt wird. In diese Klasse fallen auch alle Strategien, die der FIFO-Philosophie (First-In-First-Out) folgen und Links, die als erstes entdeckt wurden, auch als erstes ansurfen. Die in dieser Thesis genutzte unfokussierte Crawlstrategie lässt sich auch am ehesten in diese Klasse einsortieren, wobei der verwendete Crawler eine Mischung aus Breiten- und Tiefensuche nutzt. Fokussiertes Crawling als eigenes Thema tritt ein paar Jahre später, Ende der Neunziger Jahre, in Erscheinung und tritt direkt im Anschluss in eine kreative Phase neuer Algorithmen ein. Der Begriff fokussierter Crawl wurde von Chakrabarti et al. geprägt, jedoch sind Linksortieralgorithmen zum effektiveren Crawlen bereits seit 1994 mit Fish-Search von De Bra und Post (1994b) ein Thema, nachdem sie zunächst auf Tiefensuche setzten [De Bra und Post 1994a]. Menzcer dagegen teilt Crawler 2004 bereits in Klassen ein, sodass man bei fokussierten Crawlern auch von ”Best-First”-Strategie Crawlern sprechen kann [Menczer et al. 2004]. ”Best-First” ist in diesem Fall eine Strategieklasse, die thematisch vielversprechende Links zuerst ansurft. Einen kleinen Vergleich der damaligen Best-First-Methoden stellen Cho et al. (1998) zusammen und vergleichen jeweils die Sortierkriterien Breadth-First, Backlinkbasiert und PageRank, wobei PageRank im allgemeinen am besten abschneidet. Mit ”Shark-Search” stellen Herscovici et al. (1998) eine erweiterte Form des Fish-Search vor. Ein Kernpunkt bei der Verbesserung von Fish-Search zu Shark-Search ist der Austausch der binären Entscheidung, ob eine Seite relevant ist oder nicht, gegen eine Bewertung durch eine ”Ähnlichkeitsengine”. Der andere Kernpunkt ist die Beachtung von Kontexten, in denen die Links im HTML-Dokument liegen. Die Anzahl der gefunden relevanten Dokumente konnte so in Tests um den Faktor 1.15 bis 3.71 gegenüber Fish-Search verbessert werden. Die Ergebnisse werden anschließend in eine nutzerinteressensbasierte Karte übertragen, sodass der Suchende letztlich eine optische Karte der für ihn interessant erscheinenden Internetseiten erhält. Eine weitere Strategie nutzen McCallum et al. (1999) bei ihrer Dokumentensuchmaschine ”Cora”. Cora durchsucht wissenschaftliche Veröffentlichungen der Universitäten Brown University, Cornell University, University of Pittsburgh und University of Texas. Um die Ergebnisse zu kategorisieren und durchsuchbar zu machen kommen hier Hidden Markov Modelle zum Einsatz.

Auch wenn die Bearbeitung des Themas im Gegensatz zu den frühen 2000ern etwas nachgelassen hat, ist Crawling in vielen Bereichen immer noch ein wichtiger Bestandteil einer Verarbeitungskette von Informationen. Es lohnt sich also, dem Thema weiterhin Beachtung zu schenken. Eine zum Zeitpunkt dieser Thesis noch nicht veröffentlichte Arbeit von Remus und Biemann (unpublished) stellt einen wichtigen Baustein dieser Thesis dar. Remus und Biemann nutzen für einen fokussierten Crawl ein statistisches Sprachmodell mit Smoothing als Bewertungskriterium, um die Relevanz einer Seite zu bestimmen. Dazu binden sie ein 5-Gramm Kneser Ney Modell an einen Crawler an und versuchen über die Menge an Daten mit einer unkomplexen Strategie einen fokussierten Webcrawl auszuführen. Die Methode, auf welche in Kapitel ”Fokussiertes Crawling mittels Sprachmodellen” näher eingegangen wird, wird als Möglichkeit gesehen, ein bereits bestehendes Korpus einzusetzen, um selbiges zu erweitern.

2.1 Crawling: Archive.org / Apache Heritrix

Ein Crawler ist ein Programm, das ähnlich zu einem normalen Browser Hypertext auswertet und durch das Netz surfen kann, z.B. mit dem Zweck Webseiten in Archive aufzunehmen und diese chronologisch zu katalogisieren. Für diese Aufgabe eignen sich theoretisch eine Menge Tools, angefangen mit einem Bash-Script und dem Download-Tool ”wget”. Effektiv hat sich mittlerweile aber eine kleine Anzahl von Softwareprojekten auf Crawling spezialisiert, die als Grundlage dienen können. Die GPL Software HT-

Track[1] bietet sich dabei gleichermaßen an wie eine Bibliothek wie z.B. crawler4J[2]. Bei den genannten Beispielen können Politenesseinstellungen gegen übermäßigen Trafficverbrauch und die Wiederaufnahme eines Crawls geregelt werden. Die beiden populärsten Vertreter von Crawlingsoftware hingegen sind jedoch Apache Nutch[3] und Apache Heritrix[4] [Mohr et al. 2004]. Nutch und Heritrix sind lang gereifte stand-alone Crawler, die jeweils in Java geschrieben sind und zum Stand der Thesis noch aktiv weiterentwickelt werden. Beide stehen unter der Apache Lizenz 2, sind gut erweiterbar, per API steuerbar und können mit Anbindung an verschiedene Datenbankformate wie Hadoop, MongoDB, Apache Solr und andere aufwarten. Heritrix setzt auf das Spring-Framework [Johnson et al. 2004] auf und wird von der Internet Archive Community (Archive.org) entwickelt und unterstützt. Archive.org ist eine Webseite, die seit den späten Neunzigern mit eigenen Crawlern (seit 2004 Heritrix) das Ziel verfolgt, das Internet zu archivieren und gibt an, derzeit 432 Milliarden gecrawlte Seiten vorzuhalten[5]. D.h. Heritrix wird bereits dafür eingesetzt, milliardenfach Webseiten stabil wiederholt anzusurfen. Der Umgang mit Millionen von Links, Dokumenten und Warteschlangen ist dabei in der Basisausführung stabil möglich. Heritrix ist in der Lage, auch aus Javascript Links zu extrahieren und so die Anzahl der entdeckten Links möglichst groß zu halten. Die für die Thesis genutzte Version 3.2 von Heritrix ermöglicht die Erstellung von großen Snapshots bzw. Checkpoints. Dieses Feature ist besonders bei Software im Experimentalstatus wichtig, da nicht jeder Crawl von neuem begonnen werden muss, sobald ein Fehler auftritt. Weiterhin versteht sich Heritrix gut auf Multithreading, sodass man über Parameter wie ”max-toe-threads” und ”expected-concurrency” die Auslastung des Servers gut einstellen kann.

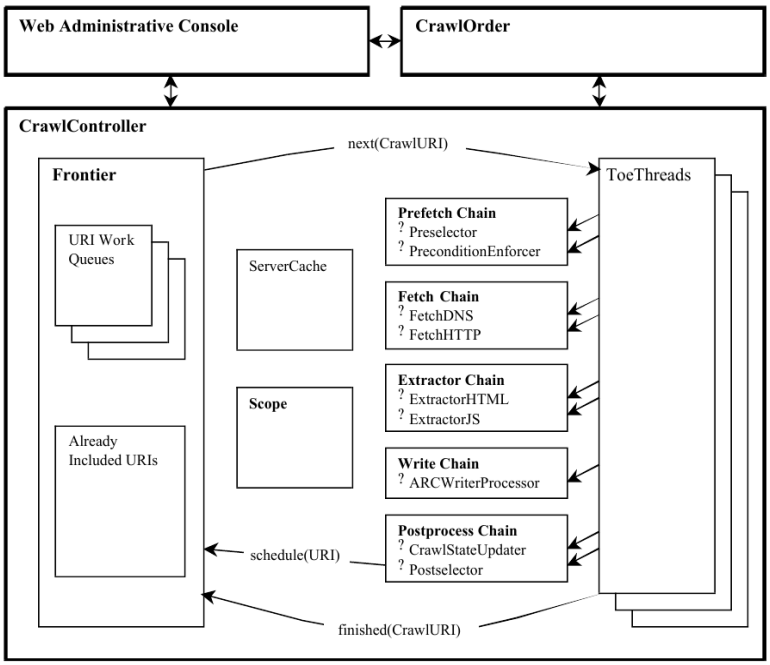


Abbildung 2.1.: Chainstruktur Heritrix (Abbildung aus Mohr et al. 2004)

Neben den oben genannten Punkten ist einer der Hauptgründe für Heritrix jedoch die Modularität. Das in der Thesis entstandene Modul kann so ohne großen Aufwand in weiteren Arbeiten verwendet werden. Für das Ziel, einen fokussierten Crawl laufen zu lassen, spricht weiterhin für Heritrix, dass relativ einfach Warteschlangen mit Priorisierung gebaut werden können, was bei Nutch wegen dem verteilten Datenmodell schwieriger ausfällt.

[1] `https://www.httrack.com` (aufgerufen Juni 2015)
[2] `https://github.com/yasserg/crawler4j` (aufgerufen Juni 2015)
[3] `http://nutch.apache.org` (aufgerufen Juni 2015)
[4] `https://webarchive.jira.com/wiki/display/Heritrix/Heritrix` (aufgerufen Juni 2015)
[5] `http://archive.org` (aufgerufen Juli 2015)

Daher soll hier nun weiter auf Heritrix an sich eingegangen werden, um später Bezug auf einzelne Bestandteile nehmen zu können. Die einzelnen Arbeitsschritte beim Crawlen werden bei Heritrix von spezialisierten Prozessormodulen erledigt. Die Module werden wiederum in sogenannten Chains platziert (siehe Abbildung 2.1). So werden in dieser Thesis z.B. für den normalen fokussierten Crawl zwei Module von Remus und Biemann verwendet, ohne dass in den Code eingegriffen werden muss. Es reicht hierzu, in der Konfigurationsdatei eine entsprechende ”Java-Bean” anzulegen und diese in einer der Verarbeitungschains zu platzieren. Dabei steht die Reihenfolge der Platzierung analog für die Reihenfolge der Verarbeitungsschritte eines Dokuments während des Crawls. Ein Dokument durchwandert dann zunächst die Chains und innerhalb der Verarbeitungsketten jedes eingebundene Modul sequentiell. Wobei hier darauf geachtet werden muss, dass jeder vom CrawlController angestoßene ToeThread eine eigene Verarbeitungskette hat, welche die zu verarbeitenden URLs aus dem Frontiermodul bezieht. Module müssen also möglichst threadsicher geschrieben werden. In Version 3.2 von Heritrix kommt bereits ein vereinfachtes Kettensystem zum Einsatz wie in Abbildung 2.2 zu sehen ist, welches jedoch den Ablauf nicht grundlegend ändert.

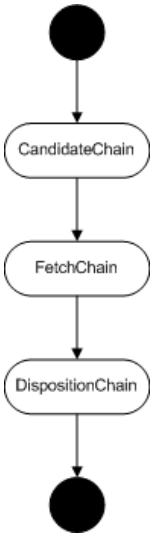


Abbildung 2.2.: Heritrix Prozessorkette (Bild: webarchive.jira.com)

In der Candidate-Chain werden URIs, die theoretisch gecrawlt werden könnten, jedoch noch nicht eingesammelt wurden, zuerst verarbeitet. Das soll z.B. sicherzustellen, dass gewisse Vorbedingungen gelten. Wird eine Candidate-URL zum Crawl ausgewählt, wandert diese in die FetchChain, wo Prozessoren wie ”Fetch-DNS”und ”Fetch-HTTP” zunächst die Adressauflösung und den Aufbau einer HTTP-Verbindung zur in der URI hinterlegten Seite, Bild oder anderen Objekten übernehmen. In dieser Chain liegen auch die Extraktoren, die den vom fetchHTTP Modul übermittelten Inhalt überprüfen. An dieser Stelle befindet sich somit der Teil, der z.B. neue Links entdeckt, wenn das vom FetchHTTP-Prozessor übergebene Objekt ein HTML-Dokument ist. Ob ein Prozessor ein Objekt verarbeiten kann, prüft der Prozessor selbst, in dem er eingangs für ihn relevante Bedingungen überprüft, wenn er die URI vom CrawlController erhält. Ist der Prozessor mit der Bearbeitung fertig, meldet er dies an den CrawlController, welcher das soeben bearbeitete Objekt an den nächsten Prozessor weiter reicht. Zu guter Letzt werden in der DispositionChain Module eingebunden, die für die Nachbearbeitung zuständig sind. Das kann z.B. ein MirrorWriter-Modul sein, welches den gecrawlten Inhalt auf der Platte ablegt, oder, je nach Anwendung, einen Bewertungsprozessor für die aktuelle Seite, sodass sich der Kreislauf schließt.

2.2 Verwandte Arbeit: Fokussiertes Crawling mittels Sprachmodellen

Mit dem Ziel bestehende Textkorpora zu erweitern bzw. zu erzeugen, crawlen Remus und Biemann (unpublished) mittels eigens geschriebener Heritrixmodule in Kombination mit einem Sprachmodell.

Als Maß wird Perplexität zur Satzbewertung verwendet. Da Heritrix aus Politenessgründen nur eine Warteschlange pro Server verwendet, um u.a. die Vorgaben einer robots.txt zu respektieren, ziehen die geschriebenen Module eine zusätzliche Priorisierungsstelle in die Heritrixstruktur ein. Zum Zwecke der Bewertung eines Links, wird die verlinkende Seite in Sätze zerlegt und dem Sprachmodell zur Bewertung vorgelegt. Der erhaltene Perplexitätswert wird dem Link zugeordnet und bei der Priorisierung berücksichtigt, sodass durch diesen Mechanismus eine Fokussierung des Crawls auf Seiten entsteht, welche vom Sprachmodell günstig bewertet werden. In einem Experiment von Remus und Biemann (unpublished) werden in Summe jeweils 500GB HTML-Daten einmal unfokussiert und fokussiert heruntergeladen. Aus den so erhaltenen Sätzen wurden für die Evaluation jeweils 300 Millionen Tokens verwendet. In der Auswertung zeigt sich, dass der fokussierte Lauf die Perplexität mit zunehmender Satzmenge reduzieren kann, während der unfokussierte Crawl verstärkt themen- und sprachfremde Seiten herunterlädt und die Perplexität steigt[6]. Die Vorteile der Methode liegen darin, dass bedingt durch das Sprachmodell sprachenunabhängig gearbeitet werden kann und vielmehr auf die Bildung einer Taxonomie wie z.B. bei Chakrabarti et al. (1999) verzichtet werden kann. So ist es möglich, ein Bewertungssystem lediglich mit Positivbeispielen zu schaffen - auf ein zur Abgrenzung dienendes Trainingskorpus mit negativen Trainingsdaten kann so verzichtet werden. Perplexität kann so als ”Zugehörigkeitsmaß” eines unbekannten Textes zu einem trainierten Korpus betrachtet werden.

2.2.1 Sprachmodell und Perplexität

Um Text auf Themenrelevanz bewerten zu können, kommt ein Modell mit 5-Grammen und ein Kneser Ney Smoothing zum Einsatz. Der von Kneser und Ney (1995) vorgestellte Algorithmus wird von Chen und Goodman (1996) im Rahmen der 1998 erweiterten Version ihres Vergleichs verschiedener Smoothingverfahren sowohl in der ursprünglichen Variante, als auch in der von ihnen modifizierten Version *kneser-kney-mod* als Sieger präferiert. Kneser Ney eignet sich somit für die statistische Vorhersage, welche Fortsetzung eines Textes am Wahrscheinlichsten ist. Als Maß für die Bewertung von Texten wird daher Perplexität genutzt, die als $PP(X)$ definiert wird:

$$PP(X) = 2^{H(X)}$$

wobei X die Menge aller Test N-Gramme ist, $p(x)$ die bedingte Wahrscheinlichkeit eines N-Gramms und $H(X)$ als Kreuzentropie des unbekannten, gesehenen Textes in Verbindung mit dem zugrundeliegenden Trainingsmodell definiert wird durch:

$$H(X) = -\frac{1}{|X|}\sum_{x \in X} log_2 p(x)$$

Bildlich gesprochen ist Perplexität ein Wert für die ”Überraschtheit” eines Modells über den erhaltenen Text. Je höher dieser Wert ausfällt, um so unwahrscheinlicher war der bewertete Text für das trainierte Modell - analog dazu auch vom trainierten Thema.

[6] Eine kleine Perplexität ist besser als eine Große.

2.3 Parallele Korpora: Bild zu Text

Geht es um Bild zu Text Korpora, so gibt es hier bereits einige sehr relevante Arbeiten, die z.T. als Projekte noch immer in vollem Gange sind. ImageNet[7] [Deng et al. 2009] ist ein sehr bekanntes Projekt und wird derzeit von Professoren und Studenten verschiedener Universitäten betreut (Stanford University, Princeton University und University of North Carolina, Chapel Hill). Es hat sich zur Aufgabe gemacht, WordNet[8] als Basis zu nutzen und darauf eine Bilddatenbank mit gleicher Struktur aufzubauen. WordNet [Fellbaum 1998] ist eine Datenbank für Englisch und andere Sprachen und ist gruppiert in Nomen, Verben, Adjektive, usw.. Wörter werden beispielsweise in sogenannte "Synsets" eingeordnet - Synsets sind Gruppierungen von Synonymen wie z.B. "Automobil" zu "Wagen" - *Zitat: "S: (n) car, auto, automobile, machine, motorcar (a motor vehicle with four wheels; usually propelled by an internal combustion engine) "he needs a car to get to work"* - WordnetSearch 3.1[9].

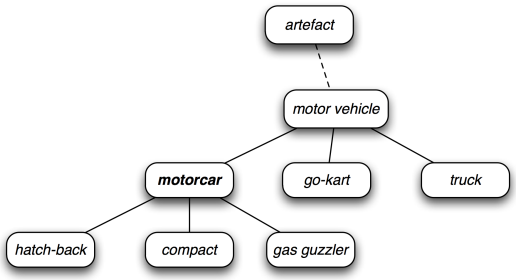


Abbildung 2.3.: Hierarchie Beispiel in Wordnet aus 'Natural Language Processing with Python' [Bird et al. 2009]

Darüber hinaus gibt es viele weitere Relationen und sortierte Abstufungen in der Datenbank, sodass in einer Baumstruktur Verbindungen und Abstufungen wie z.B. "President → Barack Obama" vorgehalten werden (s. Anhang A.1 (a)). Dabei bildet Barrack Obama eine Instanz der Kategorie "Präsident" und damit gleichzeitig ein terminierendes Blatt im Baum der Hierarchie (s. Abbildung 2.3).

Das als Beispiel genannte ImageNet stützt sich auf diese in WordNet gebildete Hierarchie und zielt darauf ab, ca. 50 Millionen gelabelte Bilder in hoher Auflösung vorzuhalten. Für jedes Synset werden dazu jeweils ca. 500-1000 Bilder annotiert und zugeordnet. Dabei kommt der Relation "is-a" die größte Bedeutung zu Gute (Bsp: "a cat - is a - mammal" - s. Abbildung 2.4) um die Relation von Hyponymen und Hypernymen aufzubauen. Zur Zeit der Veröffentlichung anno 2009 lag der Bestand bei 3,2 Millio-

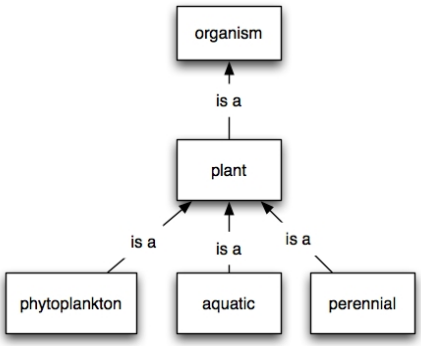


Abbildung 2.4.: Beispiel 'Is-a' Relation [(c) stevenloria.com]

[7] `http://image-net.org` (aufgerufen August 2015)
[8] `http://wordnet.princeton.edu` (aufgerufen Juli 2015)
[9] `http://wordnetweb.princeton.edu/perl/webwn` (aufgerufen August 2015)

nen annotierten Bildern in 12 Bäumen. Gegenwärtig zur Zeit dieser Thesis zählt der Counter auf der Projektwebseite 14 Millionen annotierte Bilder in 21.841 Synsets. ImageNet bezieht Bilder über Bildersuchmaschinen, um diese den Synsets zuzuordnen. Dabei liegt die Accuracy beim Sammeln bei ca. 10%, sodass um das Ziel von ca. 500-1000 sauberen Bildern zu erreichen, pro Synset ca. 10.000 Bilder gesammelt werden. Dabei muss wegen verschiedener Restriktionen der Suchmaschinen die Suchanfrage mit jedem erreichten Suchlimit erneut modifiziert werden. *"For example, when querying "whippet", according to WordNet's gloss a "small slender dog of greyhound type developed in England", we also use "whippet dog" and "whippet greyhound"."* [Deng et al. 2009, S.3]. Weiterhin werden die Suchanfragen zusätzlich in andere Sprachen wie Chinesisch, Spanisch, Niederländisch und Italienisch übersetzt, um weitere Bilder zu erhalten. Der so erhaltene Bilderpool wird von Menschen zu "candidate Images" aussortiert und gefiltert.

Die Bewertungsaufgaben werden dazu über Crowdsourcing mittels "Amazon Mechanical Turk" auf viele Menschen weltweit verteilt, wobei die Inanspruchnahme des Dienstes Kosten verursacht. Das Kofferwort Crowdsourcing bezeichnet das Auslagern von einfachen Aufgaben an eine große Menge von Menschen (Crowd und Outsourcing). Dazu ist es wichtig, dass eine Aufgabe sehr gut parallelisierbar ist. Zusätzlich ist es wegen dem Faktor Mensch und z.B. unterschiedlicher Bildungsgrade in Bezug zur Aufgabe wichtig, die Aufgaben möglichst einfach zu halten. Amazon ist mit seinem Dienst "Mechanical Turk" ein Anbieter solch einer Plattform für internationales Crowdsourcing. Weiterhin sei hier noch "Crowdflower" als Crowdsourcingmetaplattform genannt, die sich zur Abarbeitung von Aufträgen Anbietern wie Mechanical Turk, oder in Deutschland "CrowdGuru" bedient. Deng et al. stellen fest, dass Synsets um so schwieriger zu annotieren sind, je tiefer sie in der Hierarchie von Wordnet stehen [Deng et al. 2009, S.5]. Beispielsweise fällt es einem ungeübten Worker noch leicht einen Hund von einer Katze zu unterscheiden, geht es aber bis tief in die einzelnen Rassen, wird die Aufgabe immer schwieriger (siehe Abbildung 2.5).

User 1	Y	Y	Y
User 2	N	Y	Y
User 3	N	Y	Y
User 4	Y	N	Y
User 5	Y	Y	Y
User 6	N	N	Y

Abbildung 2.5.: Antworten zu "is this a burmese cat?" - Deng et al. (2009)

Am Ende der Arbeit steht jedoch ein ambitioniertes Korpus, das zur Klassifikation in anderen Arbeiten als Referenz oder als Trainings/Testkorpus genutzt werden kann. Hier noch ein konkretes Beispiel, das gut die aus Wordnet entnommene Hierarchie übertragen auf die "is-a" Relation mit dem Ergebnis in Image-Net demonstriert, indem die Abstufungskette von Säugetier bis herunter zum Husky und von einem Gefährt bis herunter zu einem Trimaran zeigt (siehe Abbildung 2.6).

Ein anderer, maschinenlastigerer Ansatz wird von Kuznetsova et al. (2013) in "Generalizing Image Captions for Image-Text Parallel Corpus" verfolgt, um ein Bild-Text Korpus zu erzeugen. Sie definieren mit "image caption generalization" eine neue Aufgabe so, dass Bildbeschreibungen so umgeschrieben werden, dass der Text stärker den Bildinhalt bzw. die visuell relevanten Anteile beschreibt. Es wird versucht, die Texte mittels Kompression auf nur bildrelevante Features bzw. Inhalte zu reduzieren. Wichtig und erhaltenswert sind dabei im Text vorkommende Objekte, die auf dem Bild zu sehen sein können, wie z.B. eine Lampe, eine Tür oder eine Kamera. Die so gelabelten Bilder werden mit Computer Vision Techniken auf Ähnlichkeit zu anderen, bereits annotierten Bildern, überprüft. Dafür werden SVMs (Support Vector Machines) eingesetzt, die mit dem oben aufgeführten Image-Net Projekt arbeiten und aus 7404 visuellen Klassifizierern für die Erkennung von Word-net Synsets auf Blattebene bestehen. Jeder Klassifi-

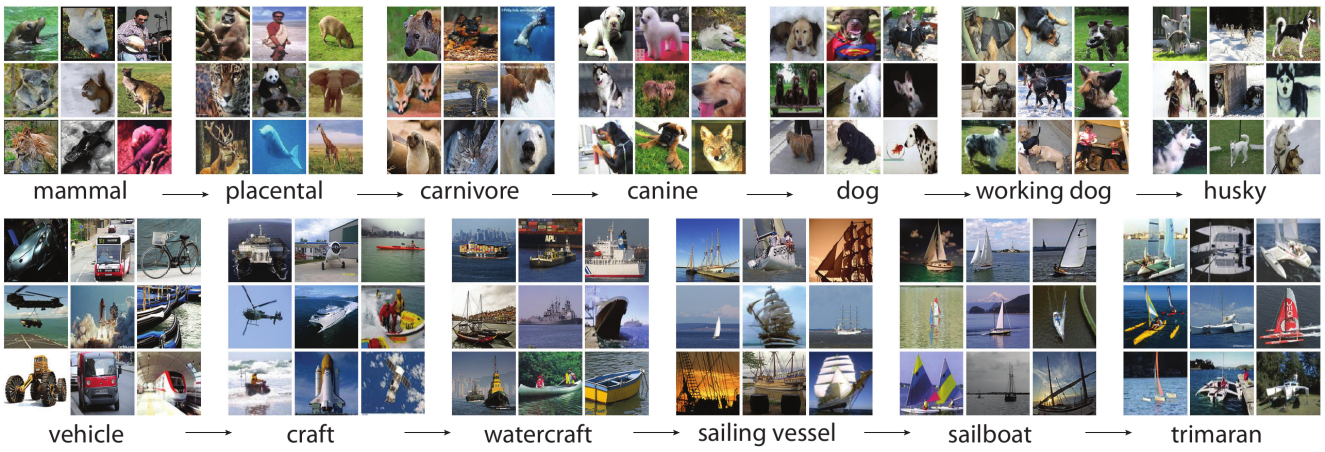


Abbildung 2.6.: ImageNet Hierarchie Beispiel (Entnommen aus Deng et al. 2009 - Seite 2)

zierer wird mit den handgelabelten Bildern aus den ImageNet Datensätzen trainiert. Die Bildähnlichkeit wird über einen Spatial Pyramid Match Kernel (SPM) mit LLC (Locality-constrained Linear Coding) anhand von formbasierten SIFT (Scale-invariant feature transform) Features bestimmt [Kuznetsova et al. 2013, S.2].

Für die Evaluation wird auch hier Amazon Mechanical Turk eingesetzt. Von Menschen wird zuerst ein Goldstandard mit 1000 Bildern erstellt. Ein Goldstandard stellt eine eindeutige Referenz für die Worker dar, an der sie sich orientieren müssen und auf dessen Einhaltung sie stichprobenartig kontrolliert werden. Die Evaluation findet zunächst intrinsisch statt, indem den Workern zwei Bildtexte zum Vergleich vorgelegt werden und die Worker entscheiden müssen, welcher der beiden ”enger” am Bild orientiert ist. Dabei schneidet das CV-basierte Modell in 81,75% der Fälle besser ab, als die Original-Überschrift. Der zweite Teil der Evaluation erfolgt extrinsisch, indem ein bereits bestehendes Korpus [Ordonez et al. 2011] genutzt wird um die 5 ähnlichsten Bilder aus einer Millionen Photos zu wählen und aus deren bestehenden Beschreibungen eine neue Überschrift für das gewählte Bild zu erstellen [Kuznetsova et al. 2013]. Das hierfür genutzte Korpus mit einer Millionen Bildern wurde aus der Online Foto-Community Flickr erzeugt [Ordonez et al. 2011].

Problematisch zeigte sich in dieser Arbeit, dass Bildbeschreibungen oft Inhalte oder Wissen enthalten, die auf dem Bild nicht zu erkennen sind bzw. abgebildet sind. Betrachtet man die rechten Beispiele in Abbildung 2.7, so ist der Satz *”James the cat is dreaming of running in a wide valley”* ein gutes Beispiel dafür, dass klar sein sollte, dass nicht einmal der Fotograf genau sagen kann, was die Katze gerade träumt. Der Text enthält dementsprechend nicht einmal unsichtbares Hintergrundwissen sondern ist rein spekulativ. Dodge et al. (2012) setzen sich mit der Frage außeinander, was sichtbares Wissen darstellt und präsentieren eine Algorithmus zur verbesserten Erkennung von nicht sichtbarem Wissen [Dodge et al. 2012].

Die Frage die sich dabei nicht nur der Maschine stellt ist ”was ist sichtbares Wissen?”. Konkreter zu dieser Frage passt der Begriff ”Traumauto”. Hier ein Beispiel: Wenn in einem Bild ein Ferrari auf der Straße neben anderen Wagen steht und der Text spricht von ”Mein Traumauto”, ist für einen Menschen klar, welcher Wagen gemeint ist. Selbst wenn Vorlieben individuell sind, vermutet man zu wissen, welchen Wagen der Textschreiber meint. Eine Maschine müsste dazu zunächst die Wertung und Bedeutung eines teuren Autos im gesellschaftlichen Umfeld kennen und in die Bewertung mit einfließen lassen können. Doch auch für den Mensch kann diese Bildbeschreibung schwierig werden: Steht der Ferrari auf dem Bild neben einem Lamborghini, Jaguar oder Bentley, ist die Frage nicht mehr so leicht, es sei denn man kennt den Textschreiber persönlich gut. Klare Definitionen und Bildbeschreibungen sind also derzeit wichtige Maßgabe für die maschinelle Arbeit mit Text- und Bild.

Abbildung 2.7.: Beispiele aus Kuznetsova et al. (2013, S.5, Figure 4). Gute Beispiele auf der linken, schlechte Beispiele auf der rechten Seite.

Ist der Ursprungstext jedoch gut bildbeschreibend, kann die Maschine den Text wie in den linken Beispielen in Abbildung 2.7 gut komprimieren und damit auf die wesentlichen bildbeschreibenden Merkmale reduzieren.

Da, wie in den beiden vorher erwähnten Arbeiten, die Annotation bzw. Evaluation von Bildern oft weiterhin final von Menschen vorgenommen wird, ist es wichtig, einige beim Crowdsourcing auftretende Probleme zu adressieren: Rastchian et al. legen in "Collecting image annotations using amazon's mechanical turk" [Rashtchian et al. 2010] dar, wie sie mittels Mechanical Turk Bilder annotieren lassen und welche Fallstricke sich dabei ergeben. Dabei wird in der Methode nahezu komplett auf Sprachtechnologie, Computer Vision und andere Hilfsmittel wie maschinelles Lernen verzichtet. Stattdessen wird ein Korpus von 9000 Bildern mit 40000 Beschreibungen durch freies Schreiben generiert. Durch die Offenheit der Aufgabe - ohne Vorgaben und Wahlmöglichkeiten - erklärt sich auch das Ergebnis als Zuweisung von 5 unterschiedlichen Bildbeschreibungen zu jeweils einem Bild. Dabei stellen sie in den Vordergrund, dass in ihrem Experiment die Auswahl der Annotatoren über Vorselektionstests eine entscheidende Rolle für die Qualität der Beschreibungstexte darstellt (siehe Abbildung 2.8).

Without qualification test
(1) lady with birds
(2) Some parrots are have speaking skill.
(3) A lady in their dining table with birds on her shoulder and head.
(4) Asian woman with two cockatiels, on shoulder
head, room with oak cabinets.,
(5) The lady loves the parrot
With qualification test
(1) A woman has a bird on her shoulder, and another bird on her head
(2) A woman with a bird on her head and a bird on her shoulder.
(3) A women sitting at a dining table with two small birds sitting on her.
(4) A young Asian woman sitting at a kitchen
table with a bird on her head and another on her shoulder.
(5) Two birds are perched on a woman sitting in a kitchen.

Abbildung 2.8.: Auszug aus Rashtchian et al. (2010, S.145, Figure 5)

Ein ähnliches Bild zeigte sich z.B. auch in einem Crowdsourcingworkshop[10] des Bereiches UKP an der TU-Darmstadt 2012: Am Ende des Workshops stand neben den Ergebnissen die Erkenntnis, dass man seine Aufgabe so weit es geht in sogenannte Microtasks zerlegen muss, damit zum einen die Fehlerrate gering gehalten werden kann (Aufmerksamkeitsspanne, Schwierigkeit der Aufgabe) und zum anderen die Gesamtaufgabe schnell durch viele Worker parallel abgearbeitet werden kann (Divide &

[10] `https://www.ukp.tu-darmstadt.de/teaching/courses/ws-1213/text-analytics` (aufgerufen August 2015)

Conquer). Die Veröffentlichung ”Leveraging crowdsourcing for paraphrase recognition” [Tschirsich und Hintz 2013, S.207-208], welche aus diesem Workshop hervorging, adressiert neben der eigentlichen Aufgabe das gleiche Problem wie Rashtchian et al. (2010) und weist gute Beispiele, gepaart mit ausführlichen Instruktionen und einer Einstiegshürde gegen schlecht qualifizierte Worker als wirksames Mittel gegen schlechte bzw. ungenaue Workerarbeit aus . Mit Blick auf Kosten und Effizienz zeigen Rashtchian et al. (2010) jedoch, dass Korpora je nach Zweck auch effizient, günstig und in kurzer Zeit erstellbar sind. Für 894$ (USD) wurde das Korpus innerhalb von anderthalb Wochen erstellt.

3 Methode

Um automatisiert Aussagen über einen Text treffen zu können, wird eine Methodik benötigt, nach der Texte maschinell klassifiziert bzw. eingeschätzt werden können. Da diese Thesis zum einen stark auf der Arbeit von Remus und Biemann (unpublished) fußt, wird analog ein Sprachmodell verwendet, zum anderen auch, um die in der Motivation angesprochene Sprachunabhängigkeit zu gewährleisten. Insbesondere wegen der sprachlichen Unabhängigkeit fallen Methoden wie z.B. die Verwendung von Grammatiken weg. Weiterhin wird, wie in Kapitel 2.2 erwähnt, durch die Wahl von Perplexität als Maß erreicht, dass nur auf Positivlisten trainiert werden kann, sodass als Trainingsmenge lediglich Dokumente benötigt werden, die dem gesuchten Themenbereich zugehörig sind. Ferner werden daher keine Negativbeispiele mehr benötigt, wie sie andere Methoden wie z.B. SVMs einsetzen.

Ein statistisches Sprachmodell kann Aussagen darüber treffen, inwieweit mit welcher Wahrscheinlichkeit Tokens zu einer Sprache gehören, die durch ein Trainingskorpus dargestellt wird. Übertragen auf N-Gramme lassen sich Wahrscheinlichkeiten für Tokenfolgen generieren, die unter Umständen bereits kurze komplette Sätze einschließen können, aber häufig jeweils nur Teilstücke eines Satzes abbilden. Mittels des Sprachmodells wird so versucht, den Fortgang eines Satzes anhand der erlernten Wahrscheinlichkeiten der N-Gramme vorherzusagen. Dabei muss für den Lernprozess eine Abwägung getroffen werden, wie groß das N zu wählen ist. Das gewählte N ist dann auch für den Bewertungsprozess fix. Ist das N klein, gehen zu viele Informationen verloren, es werden jedoch viele Beispiele von Tokenfolgen gelernt, sodass die Aussagekraft der Wahrscheinlichkeiten größer ist. Ein großes N verliert dagegen weniger Informationen, bedeutet hingegen einen wesentlich größeren Rechen- und Speicheraufwand. Gleichzeitig reduziert sich jedoch die Anzahl gelernter Tokenfolgen, sodass bei wenigen Beispielen die Aussagekraft gering ist, bei vielen Gelernten dagegen jedoch sehr groß. Auf unterstützendes Stemming, also der Reduzierung der Tokens auf die Wortstämme, wird wegen der gewünschten Sprachunabhängigkeit verzichtet. In Anlehnung an Remus und Biemann wird N=5 und Kneser-Ney Smoothing verwendet.

3.1 Unterscheidung Bild und Gesamtseite

Anders als bei einem fokussierten Crawl, betrachtet der unfokussierte Crawl die Gesamtseite nicht. Alle extrahierten Links werden unbearbeitet weiter an die DispositionChain gegeben (vergl. Abbildung 2.2). Die Vorgehensweise im fokussierten Crawl nach Remus und Biemann (unpublished) bewertet einmalig die Gesamtseite und gibt allen ausgehenden Links den für die Seite ermittelten Perplexitätswert. Dieser Wert wird an die aktuell besuchte URI angehängt und im späteren Verlauf der DispositionChain auf die ausgehenden Links angewandt, sodass die Wahrscheinlichkeit für eine Höherpriorisierung des ausgehenden Links größer ist, wenn die verlinkende Seite eine niedrigere Perplexität erzielt (abhängig von den Grenzwerten der Buckets).

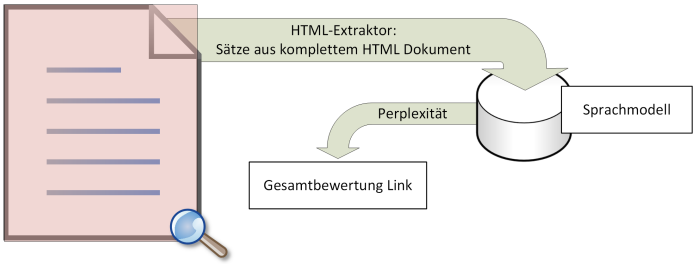


Abbildung 3.1.: Linkbewertung bei Remus und Biemann: Es dient stets das komplette Dokument zur Linkbewertung

Bei der neuen Methode für den Crawl mit Keywords, wird, anders als bei Remus und Biemann, zwischen Seite und Link unterschieden. Die Intention dahinter ist, dass eine Seite Links und Bilder enthalten kann, wovon die Bilder oder Links möglicherweise thematisch passend sind, aber die Seite selbst nicht zum Thema passt. So kann eine Internettageszeitung beispielsweise über verschiedenste Themen berichten, was für eine schlechte Perplexität sorgen würde. Eine Verlinkung auf einen themenrelevanten Artikel, wie z.B. eine Rubrik mit Tipps & Tricks für Haustiere, erhielte so trotz Themenrelevanz eine hohe und damit strafende Perplexität.

Als Konsequenz daraus ergibt sich die Einführung eines getrennten Bewertungsschema der ausgehenden Links und eingebetteten Bilder (s. Abb. 3.2), was ein wenig Ähnlichkeit mit der Kontextsensitivität in Shark-Search hat [Herscovici et al. 1998]. Da thematisch passende Seiten intuitiv angenommen auch bessere Links vorhalten werden, wird ein Link, wie im normalen fokussierten Crawl, zunächst mit dem Perplexitätswert der aktuell angesurften Seite versehen. Im Falle eines Links wird die Perplexität des Linktextes bestimmt. Weiterhin kommt bei der Linkbewertung eine Keywordgewichtung zum Einsatz, für welche Keywords automatisch vor dem Start des Crawls generiert werden. Ist bei einem Link die Perplexität des Linktextes niedriger als ein Schwellwert, wird die Linktextperplexität, anstelle der Dokumentperplexität verwendet und mit den automatisch generierten Keywords verrechnet.

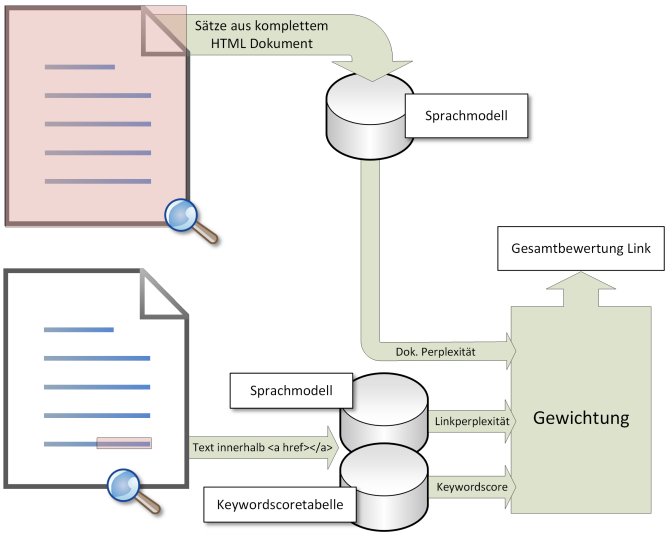


Abbildung 3.2.: Linkbewertung Thesis: Dokument und Link werden getrennt bewertet und am Ende zu einem Wert verrechnet, welcher in Kapitel 3.6 näher beschrieben wird.

3.2 Ansatzpunkt Extraktor mit Bildfunktion

Die selbstgeschriebenen Erweiterungen brechen teilweise aus der vorgegebenen Struktur von Heritrix aus, indem Elemente und Bilder bereits beim Erfassen auf der Festplatte abgelegt werden. Nach dem Kettenmodell müsste dies in der DispositionChain nahe der anderen Writermodule wie Mirrorwriter oder WARCWriter stattfinden (siehe Abbildung 3.3). URIs wandern durch die drei genannten Ketten und dabei durch die in den Ketten eingehängten Prozessoren (s. Kapitel 2.1 bzw. Abbildung 3.3). Möchte man nun z.B. nur Bilder herunterladen, würde es genügen, eine Mirrorwriter-Bean in die DispositionChain einzuhängen und über die Konfigurationsdatei Vorfilter zu definieren. Jeder Prozessor erbt eine Methode ”shouldProcess()” in der geprüft wird, ob der Prozessor das ihm vorgelegte URI Objekt bearbeiten soll oder nicht. Ein Dateifilter über die Konfigurationsdatei wäre so eine Möglichkeit dem Prozessor mitzuteilen, dass er nur Dateien vom Format jpg, jpeg, gif usw. auf die Festplatte schreiben soll - ohne den Quelltext des Prozessors zu verändern.

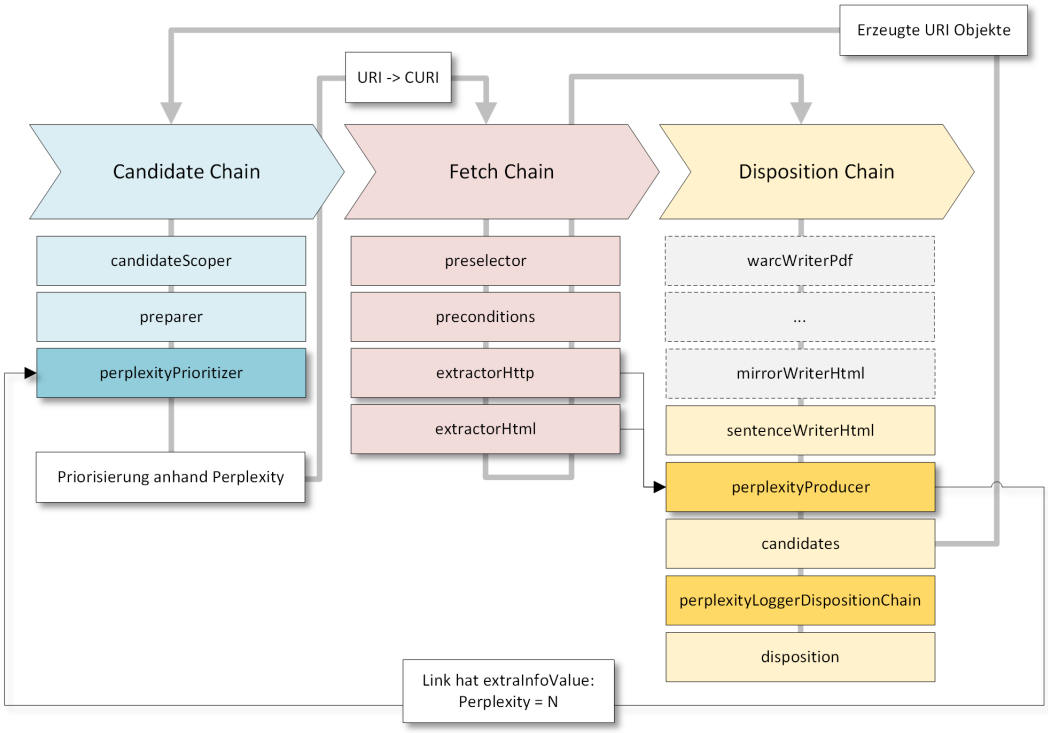


Abbildung 3.3.: Normales Heritrix Chainmodell mit den Modulen von Remus und Biemann (unpublished): In der FetchChain erzeugte Links werden mit der Perplexität des aktuellen Dokuments in der Dispositionchain versehen und in der Candidatechain zur Priorisierung herangezogen (Remus und Biemanns Module sind farblich kräftiger hervorgehoben, Deaktiverte grau).

In der Arbeit von Remus und Biemann (unpublished) werden URI Objekte mit einem ExtraInfo Feld versehen, in welchem neben dem ausgerechneten Perplexitätswert des HTML-Dokuments auch der Perplexitätswert des auf dieses Dokument verlinkenden Dokuments hinterlegt wird (s. Pfeil ”perplexityProducer - perplexityPrioritizer” in Abb. 3.3). Das ExtraInfoFeld füllen sie mittels eines eigens geschriebenen Prozessors namens ”PerplexityProducer” innerhalb der DispositionChain. Über eine einfache Methode können auch andere Key-Value-Paare an die bearbeitete URI mittels eines eigenen ExtraInfo Feldes angehängt werden. In der CandidateChain setzen sie nun einen weiteren, eigens geschriebenen Prozessor namens ”PerplexityPrioritizer” ein, welcher in einem URI Objekt gezielt nach dem Key-Value-Paar für die Perplexity der Parent-URI sucht und diese zur Priorisierung in der Warteschlange nutzt (s. Abb. 3.3 - nach dem PerplexityPrioritizer). Dazu wird sich der heritrixeigenen Buckets bedient, bildlich gesprochen ein paar Eimer, in denen URIs entsprechend ihrer Kategorisierung in ”normal”, ”medium” und ”high” landen. Die Trennung zwischen den Eimern erfolgt durch einfache Grenzwerte, die in der Konfiguration definiert werden. Die Kategorie ”highest” bleibt Spezialobjekten wie z.B. Seed-URIs (in der Konfiguration angegebene StartURLs bei denen der Crawl beginnt) vorbehalten.

Ein intuitiver Ansatz liegt nun zunächst darin, Informationen über ein Bild, welches ebenfalls als Datei-URI irgendwann durch die Verarbeitungsketten wandert, als Extrainformationen über Key-Value-Paare in der FetchChain zu extrahieren und zu hinterlegen. Im Anschluss könnte man diese analog zu Remus und Biemann (unpublished) über einen eigenen Postprozessor verarbeiten lassen. Die Implementierung dieser Arbeit geht einen anderen Weg, der wie erwähnt abseits des Ablaufplans von Heritrix liegt. Die Speicherung der Bilder und Bildinformationen werden in der Implementierung bereits in der FetchChain abgearbeitet. Nötig wird dies durch die Verarbeitungsweise der URI Objekte, wenn diese von der Candidate-Chain in die FetchChain übertreten. Es wird versucht, dies durch ein etwas umfangreicheres Beispiel zu zeigen:

An der Übertrittsstelle von Candidate- zur FetchChain erzeugt der CrawlController ein neues URI-Objekt und verliert die bereits angehängten Informationen aus dem vorherigen Lauf (s. Abb. 3.3 - Übergang Blau zu Rosa). Standardmäßig werden in den Extraktionsprozessoren alle erkannten Objekte als Link behandelt. Dabei stellt ein ausgehender Link die gleiche Linkobjektklasse dar wie ein eingebundenes Bild, sodass ein Bild durch die gleiche Verarbeitungskette läuft, wie eine HTML-Seite. Für einen normalen Crawl stellt das kein Problem dar, denn die einzige Information, die der Crawler über einen zu crawlenden Link benötigt, ist der Link selbst - und dieser ist im URI-Objekt enthalten. Das aktuelle URI-Objekt in einem Prozessor wird dabei immer als CURI bezeichnet. In der Arbeit von Remus und Biemann (unpublished) stellt es auch kein Problem dar, da die zu bewertende Seite bereits durch die FetchChain gelaufen ist und in der DispositionChain zur Bewertung kommt.

Für die Bildextraktion mit Informationgehalt ergibt sich aber designbedingt das Problem, dass ein Bildlink als nackte URL ohne Begleitinformationen bei einem Prozessor in der FetchChain ankommt. An dieser Stelle ist die Information über den Standort des Bildes innerhalb einer HTML-Seite bereits verloren. Es ist also nötig, bereits im Extraktor die Bewertung des HTML-Dokumentes vorzunehmen, in welchem der Link zum Bild eingebettet ist. Dazu ist es wichtig zu wissen, dass wenn in einem Extraktionsprozessor z.B. ein Link erkannt wird, der Prozessor ein Linkobjekt nach oben hin in der Hierarchie bekannt gibt. Der Crawlcontroller erhält dabei ein neues LinkObjekt, welches später als URI durch die Kette wandern wird. Der aktive Prozessor hat jedoch stets nur Zugriff auf die CURI und keinen Zugriff mehr auf das soeben erzeugte Objekt. Das bedeutet, dass Extrainformationen nur angehängt werden können, wenn der soeben neu erzeugte Link als CURI vorliegt. Man muss also, selbst um einfach im aktuellen Dokument gefundene Objekte direkt mit Zusatzinformationen zu versehen, die Klasse des LinkObjekts so erweitern, dass bereits bei der Erzeugung einer neuen Instanz die Zusatzinformation mitgegeben werden kann. Ein zweites Problem taucht auf, wenn das gleiche Bild auf zwei verschiedenen Seiten eingebunden wird. Da URL-Objekte wegen Redundanzvermeidung nur einmal vorgehalten werden, muss die Information direkt im Extraktor bezogen und auf dem lokalen Speicher abgelegt werden. Andernfalls könnte das erzeugte URI Objekt nicht durch die Verarbeitungsketten laufen und die zusätzlichen Informationen zum gleichen Bild auf andereren Seiten gingen verloren. Schriebe man den Extraktor so um, dass er sämtliche Informationen an eine Bild-URL anhängt wie im oben genannten Ansatz, wären die Informationen entweder verfallen, da die URL bereits abgearbeitet wurde, oder es wären möglicherweise viele Instanzen der gleichen URL in der DispositionChain verarbeitet worden. Die Mehrfachverarbeitung der gleichen URL mit anderen Informationen führte in diesem Fall zu Overhead und vervielfachtem Traffic für ein Bildobjekt.

Es kann jedoch wichtig sein, Informationen zu Bildern zu speichern, die im Crawl mehrfach vorkommen. So kann es z.B. wenn ein Bild auf mehreren Seiten eingebunden wurde, trotzdem anderweitig im ALT oder TITLE Tag beschrieben worden sein. Es kann auch in einem komplett anderen Zusammenhang verwendet worden sein. Diese mehrdeutigen Informationen sollen daher nicht verloren gehen.

Aus diesen genannten Gründen wird daher die Bewertung und die Speicherung der Bilddaten im in dieser Thesis geschriebenen Prozessor direkt in den Prozessor verlagert und somit im Chainmodell bereits in die FetchChain verschoben (siehe Abbildung 3.4). Dazu wurde u.a. ein eigenes Linkobjekt als Erweiterung des Standardlinkobjektes eingeführt, welches bereits im Konstruktor die Möglichkeit bietet, ein Key-Value-Paar mitzugeben. Damit ist gewährleistet, dass die Kompatibilität zu den von Remus und Biemann (unpublished) geschriebenen Modulen aufrecht erhalten werden kann. Weiterhin wird der Extraktor modifiziert: die Methode der URI-Objekterstellung wird einmalig um eine überladene Methode erweitert, sodass das Modul mit Heritrix auch bei Nichtnutzung der Bildspeicherung wie der normale Standard HTML-Extraktor fungieren kann. Die Bildspeicheroption ist bei Bedarf über Konfigurationsparameter in der Crawler-Beans.cxml abschaltbar (vgl. Listing A.8).

Die um das Bild herum gesammelten Informationen werden in eine Textdatei in einem XML-Format geschrieben (siehe Listing 3.3). Dazu wird der Extraktor um Schreibfähigkeiten erweitert, welche die Verzeichnisstruktur des Standardmirrorwriters nachahmen. Das Herunterladen des Bildes erfolgt innerhalb des Prozessors über die Bibliothek ImageIO. Das führt designbedingt u.a. dazu, dass die heruntergе-

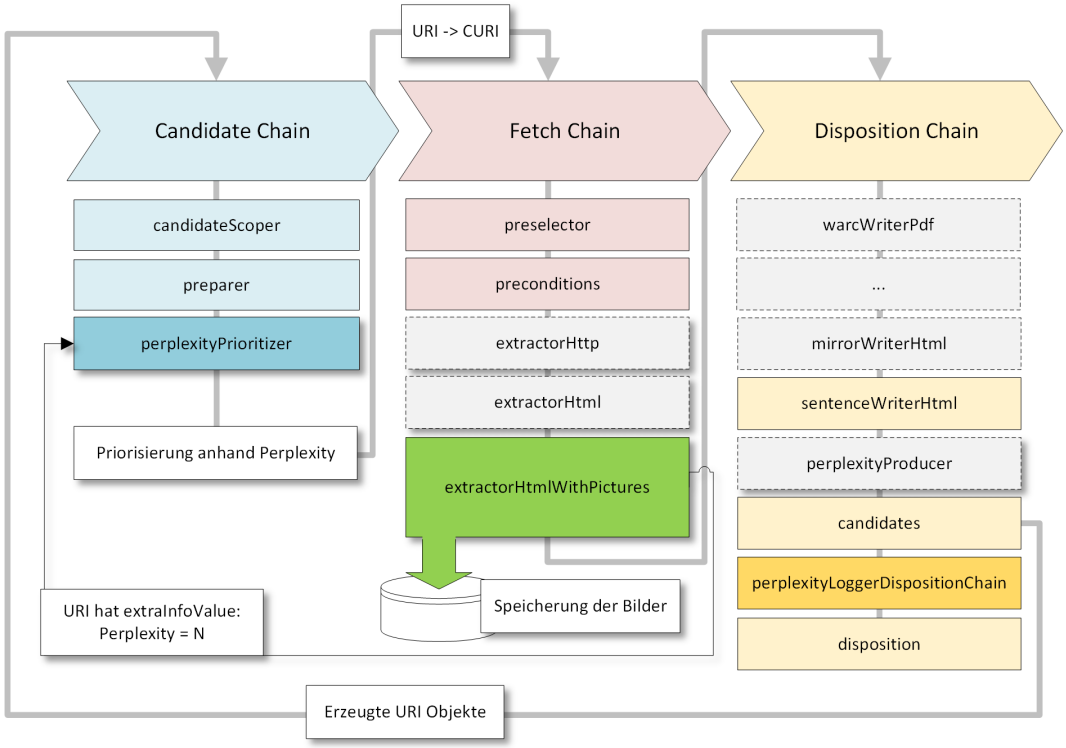


Abbildung 3.4.: Ausbruch aus dem Heritrix Chainmodell mit in dieser Thesis geschriebenem Modul: Link-URI-Objekte werden im Extraktor erstellt und direkt mit Perplexität versehen. Die Priorisierung ist von Remus übernommen (eigenes Modul grün hervorgehoben, Remus' Module farbig kräftiger). Die Bildspeicherung findet direkt statt und eine Bild-URI wird nicht als eigenes Objekt in die Verarbeitungskette gegeben (Vgl. Abb. 3.3).

ladene Bildmenge nicht als Traffic in der Statistik von Heritrix auftaucht. Im Falle eines erfolgreichen Herunterladens wird die XML-Datei mit dem gleichen Dateinamen wie das Bild ins gleiche Verzeichnis nach dem Schema *Crawljobname/Datum/Mirror-Speicherpfad/url-zerlegt-in-Verzeichnisse/Bilddatei* geschrieben (siehe Abbildung 3.5). Neben den aus dem HTML-Kontext erhaltenen Informationen werden weiterhin die echten Ausmaße des Bildes durch ImageIO bestimmt und gespeichert. So ist es möglich, noch im Nachgang Bilder nach echter Größe zu filtern, ohne die Bilddatei ein weiteres Mal auf der Platte auslesen zu müssen. Um keine Informationen zu verlieren, wird die XML-Datei erweitert, sollte das Bild mit identischer Bild-URL auf einer anderen Seite eingebunden worden sein. Durch das so gewählte Design kann das gleiche Bild mehrfach von Seiten gemappt werden.

3.3 Keywordgenerierung

Um in einem Crawlinglauf wichtigen Worten eine stärkere Gewichtung verleihen zu können, wird vor dem Lauf eine statische Keywordliste generiert. Diese Keywords erhalten während der Erstellung ein eigenes Gewicht und werden beim späteren Rating der HTML-Tag Inhalte genutzt und nehmen so entsprechend Einfluss auf die Bewertung eines Textes innerhalb einer HTML-Seite. Da versucht wird, kein Vorwissen über die genutzte Sprache zu verwenden, wird hier eine Positivliste generiert, d.h. es wird versucht einen Ansatz mit Stoppwortfilterung zu umgehen, da eine Stoppwortliste bei einer fremden Sprache möglicherweise nicht zu Verfügung steht. Es werden dazu initial zwei Seed-Korpora gebildet. Die Seiten, die das Thema des Crawls gut abdecken, werden in einer positiven Seedliste gespeichert. Als Gegenpol wird eine neutrale Seedliste mit Webseiten gebildet, die das Thema in der gleichen Sprache nicht enthalten. Als Beispiel wurden hier in der Positivliste Seiten gewählt, die vornehmlich das Thema

Abbildung 3.5.: Vergleich Bilddatei und Infodateiname (Bild: pictures-of-cats.org)

Katzen behandeln. In diesem Fall die Wikipedia Artikel zu: *Cat*, *Cat behavior*, *Cat body language*, *Kitten*, *Maine Coon*, sowie die URLs des *Domestic Cat Bereichs auf Nationalgeographic* und die *Liste aller Katzenrassen auf Petfinder.com* (siehe Listing A.1). In der neutralen Liste werden Seiten gewählt, die z.B. Autos, Personen oder Tagesgeschehen thematisieren. Es ist hierbei wichtig, dass genug neutrale Seiten enthalten sind. Als Daumenregel sollten es mindestens 6-7 große Seiten oder Artikel sein, die auf der neutralen Seite stehen. So kommen hier die Wikipedia Artikel: *Car*, *John F. Kennedy*, *The Wall Street Journal*, *Sepp Blatter* und *Michael Jordan* zum Einsatz. Zusätzlich sind noch die URLS der *Washington Post* und *New York Times* enthalten (siehe Listing A.2). Im Anschluss werden diese Seiten initial gecrawlt und jeweils zwei Korpora gebildet. Da der Keywordgenerator in Java geschrieben ist, aber für die Menge an Seiten nicht unbedingt Heritrix erforderlich ist, wird die Aufgabe in diesem Rahmen von JSoup[1] übernommen. Weiterhin wird das Languagemodell zur Textvorverarbeitung, wie Tokenisieren und Filtern über die AbstractStringProvider-Klasse von Remus und Biemann angebunden. Das Modell kann zu diesem Zeitpunkt mit einem Dummy befüllt sein, da der Keywordgenerator an dieser Stelle lediglich die eingebauten Mechanismen des StringProviders nutzt und keine Abfragen an das Sprachmodell benötigt werden.

Über beide Korpora wird anschließend ein TF-IDF Score gebildet (TF: Term Frequency, IDF: Inverse Document Frequency). Danach wird über alle Wörter des positiven Korpus iteriert und die Differenz zum TF-IDF Score des neutralen Corpus gebildet. Worte die nicht im neutralen Korpus vorkommen, werden mit einem eigenen Gewicht bedacht, sodass deren Relevanz im Vergleich weiterhin erhalten bleibt. Weiterführend wird nun eine Rangliste der Begriffe, sortiert nach der TF-IDF Differenz, gebildet. Dabei zeigt sich, dass viele Tokens noch die gleiche Differenz aufweisen und daher nicht ersichtlich wird, ob z.B. ”cat” oder ”snowshoe” der wichtigere Begriff ist. Daher werden die Gewichte der einzelnen Tokens durch eine Multiplikation mit der korpuseigenen TF verstärkt.

Die TF-IDF Differenz ergibt sich somit wie folgt (vgl Quelltext in Listing A.3) wobei x_i als aktuelles Token definiert ist:

$$keywordscore_{amplified}(x_i) := tf_{positiv}(x_i) \times [(idf_{neutral}(x_i) - idf_{positiv}(x_i)]$$

Auf diese Weise erhält z.B. das Wort ”cat” einen Faktor von 1015,4. Da der Vorgang an das Verhalten eines Audio-Verstärkers angelehnt ist, wird der Parameter im Modul ”Amplifier” genannt. Als Beispiel ein Auszug aus der Keywordliste mit und ohne Amplifier (siehe Tabelle 3.1).

[1] `http://jsoup.org` (aufgerufen im Juli 2015)

Tabelle 3.1.: Keywordscore im Vergleich

Keywordliste ohne Amplifier	**Keywordliste mit Amplifier**
1. cat = 1.550	1. cat = 1015.400
2. snowshoe = 1.550	2. cats = 834.023
3. cats = 1.550	3. coon = 141.831
4. rex = 1.550	4. breed = 139.519
5. ...	5. ...
15. balinese = 1.509	15. catus = 42.146
16. ocicat = 1.509	16. veterinary = 42.146
17. eyed = 1.509	17. kittens = 42.146
18. sphynx = 1.509 ...	18. rex = 41.856 ...

Rang	Token	Differenzwert
455	they	4.484
4482	make	0.0
5156	have	0.0
5586	therefore	-0.250
6021	there	-0.487
6082	themselves	-0.560
6303	then	-0.828

Tabelle 3.2.: TFIDF-Differenzwerte für Stoppworte

Die oberen 50 Worte/Tokens werden für die statischen Keywords herangezogen (siehe Anhang A.5 linke Seite). Als positiver Effekt zeigt sich in verschiedenen Testläufen, dass Stoppworte meist eine sehr geringe Differenz, bis hin zu 0 haben, sodass diese Worte/Tokens nicht in die Keywords mit einfließen. Auszugsweise hier die Differenzwerte für die Stoppworte *they*, *make*, *have*, *therefore*, *there*, *themselves*, *then* der Katzenkeywords in Tabelle 3.2.

Tokens mit negativer Differenz hingegen deuten darauf hin, dass diese in den Kontext des neutralen Korpus gehören und können daher auch weggeschnitten werden. Weiterhin fällt in einigen Testläufen positiv auf, dass die Anzahl der themenrelevanten Seiten nicht all zu groß sein muss, um eine erste intuitiv nützliche Keywordliste zu erhalten. Als alternatives Thema wurde für diesen Beispielversuch ”Vulkane” gewählt und lediglich der Wikipediaartikel *Volcano* in das Positivkorpus aufgenommen. Das neutrale Korpus bleibt dabei gleich (Listing A.2). Ein Blick in die oberen 20 Keywords zeigt Begriffe wie *volcanoes*, *volcanic*, *volcano* auf Platz eins bis drei. Es folgt direkt *lava* auf Rang 4, auf den Folgerängen finden sich neben *eruption*, *ash*, *gases*, *magma* auch weitere Pluralformen und Adjektive. Ein Auszug von 11 Keywords eines etwas abwegigeren, weil fiktionalen Beispiels mit dem Wikipedia Artikel *Death Star* fördert eine Liste in der Reihenfolge *wars*, *jedi*, *vader*, *lucasfilm*, *star*, *darth*, *leia*, *superlaser*, *tarkin*, *universe*, *rebels* zu Tage. Selbst wenn das gesuchte Objekt dabei nicht so genau getroffen wird wie im Vulkanbeispiel, zeigt sich auch hier die Themenrelevanz der Keywords.

3.4 Rating Gesamtseite

Mit Bezug auf die Unterscheidung der Probleme, wird hier der Fokus auf den textlichen Inhalt der Seite gelegt, um eine weitere Maßzahl zur thematischen Zugehörigkeit zu finden, die dann in die finale Bewertung mit einfließt. Dazu wird analog wie bei Remus und Biemann (unpublished) zunächst der gesamte Text aus der Seite an das einkonfigurierte Textextraktormodul geleitet. Dieses Modul filtert nicht-UTF-8-konforme Zeichen aus und extrahiert aus dem HTML-Code zunächst reinen Text. Die Textextraktion erfolgt wahlweise über JSoup oder den Boilerpipeextraktor von Kohlschütter et al. (2010). Beide Extraktoren unterscheiden sich darin, dass JSoup sämtliche Texte aller Objekte unterhalb der <body> Umgebung einbezieht. Der Boilerpipeextraktor versucht dagegen, wiederkehrende Elemente wie Menüleisten, Headerbilder etc. über verschiedene statistische Ansätze wie z.B. geringere Textdichte in Navigationsbereichen innerhalb der HTML-Struktur zu erkennen und auszufiltern und so den Text von wiederkehrenden, nicht zum Inhalt beitragenden, Einträgen zu befreien. Im günstigen Fall erhält man so vom Umfeld der Webseite bereinigte Textblöcke, die maßgeblich den Inhalt einer Webseite darstellen. Ist dies abgeschlossen, wird der extrahierte Text über das Sentencemaker Modul in Satzform gebracht und zurückgegeben. Der so aufbereitete Text wird jetzt über den Stringprovider, welchen das Sprachmodell anbindet zur Bewertung vorgelegt. Der erhaltene Perplexitätswert steht dem Prozessor dann für die Dauer der Bearbeitung einer aktuellen CURI zu Verfügung.

3.5 Rating Bildumgebung

Wird ein Bild im DOM-tree (DOM: Document Object Model) erkannt, so wird zunächst geprüft, ob das Bild über einen Alt- oder einen Title-Tag verfügt. Ist dies nicht der Fall, wird das Bild verworfen. Verfügt es jedoch über einen der beiden Tags, so wird das Bild zunächst zum Herunterladen markiert. Desweiteren werden die Tags analysiert und es wird versucht, weitere Informationen über die Umgebung des Bildes zu sammeln. Die Alt- und Title-Tags werden extrahiert und an das Sprachmodell gereicht. Zusätzlich wird der Text auf Keywords geprüft. Der vom Sprachmodell erhaltene Perplexitätswert wird nun mit dem Keywordscore verechnet, wobei das Ergebnis den abschließenden, an Perplexität orientierten, gewichteten Scorewert ergibt.

Um die Umgebung zu bewerten, wird der DOM-tree jeweils einmal nach Parent- und nach Siblingknoten abgelaufen. D.h. es werden die Textinhalte der übergeordneten Elemente bewertet und die der direkten Nachbarn auf gleicher Ebene.

Listing 3.1: Beispiel leere Parenttags

```
...
<p id="parent3">I found this little stray this morning in my backyard. Isn't she beautiful?</p>
  <p id="parent2">
    <p id="parent1">
      <img id="target" src="somecat.jpg" alt="Some cat in my backyard" title=""/>
      <p id="siblingP"></p>
      <img id="siblingPic" src="navigation_arrow.gif" alt="Next picture"/>
    </p>
  </p>
</p>
...
```

Die maximale Entfernung wurde dabei auf fünf Elemente festgelegt, da dies in internen Experimenten als guter Wert abschnitt - sie ist aber in der Konfigurationsdatei frei anpassbar. Um unnötigen Speicherplatzverbrauch zu verhindern sollte diese Zahl jedoch nicht zu hoch gewählt werden. In vorangegangen Läufen zeigte sich, dass in einer flachen HTML-Hierarchie bereits der <body>-tag sehr schnell erreicht

werden kann und spätestens ab dort die komplette HTML-Seite in der XML-Datei gespeichert wird. Ein weiterer wichtiger Grund ist, dass eine eine höhere Zahl das Sprachmodell noch stärker belastet, da die maximale Entfernung gleichzeitig, mathematisch betrachtet, als Faktor auf die Zahl der Anfragen an das Sprachmodell wirkt. Wird die Zahl jedoch zu klein gewählt, ist der Informationsanteil der Bildumgebung möglicherweise zu gering und es fehlen womöglich im Nachgang Informationen, warum ein Bild wie eingestuft wurde. In die Bewertung gehen nur existierende und mit Inhalt versehene Elemente ein. Es wird davon ausgegangen, dass leere Tags der optischen Gestaltung der Seite dienen und weniger der Strukturierung. D.h. leere Parenttags wie im Beispiel mit <p id=”parent1”> und <p id=”parent2”> fließen nicht in die Bewertung mit ein, da weder das Bild mit der id ”siblingPic” noch der Paragraph ”siblingP” über Text verfügen (siehe Listing 3.1). Erst <p id=”parent3”> verfügt über Textinhalt und würde so als Parentelement erster Ordnung mit dem Text ”I found this little stray this morning in my backyard. Isn´t she beautiful?” in die Bewertung einfließen.

3.6 Rating Link

Für die Bewertung des Links wird als Basismaß der Dokumentperplexitätswert genutzt. D.h. kann die Bewertung keine Kriterien finden, die den Link interessanter machen, wird, wie bisher auch, die Dokumentperplexität genutzt, um den Link zu gewichten. Vorläufige Tests haben gezeigt, dass die weitere Umgebung eines Links oft kaum Text enthält, oder der Link oft in Navigationsebenen liegt, deren Umgebung ebenfalls wenig Hinweise auf das Ziel des Links liefern. Es wird daher zum aktuellen Stand nur der Text bewertet, den ein Link umschließt. Wenn ein Link Informationsgehalt aufweist, dann ist meist der vom <a>-Tag umschlossene Text ein guter Indikator für den Inhalt des Ziels. Sollte ein Link trotzdem themenspezifisch nutzlosen Inhalt wie ”click here” enthalten, greift als Fallback-Bewertung die Dokumentperplexität. Enthält der Text des Links jedoch Tokens die im Keywordverzeichnis stehen, werden diese gewichtet und mit der Perplexität des Links verrechnet. In Listing 3.2 wurde die Bewertung auf ”Verbose” gestellt und zeigt einen Lauf, der mit einem englischen Sprachmodell trainiert wurde und englischen Keywords einsetzt. Wie man erkennen kann, erwartet das in Englisch trainierte Sprachmodell keinen deutschen Text und straft den Text mit einem hohen Perplexitätswert ab. Dennoch wird das Keyword ”Maine Coon” erkannt und der finale dem Link zugeordnete ”Perpscore” gesenkt.

Listing 3.2: Bewertung eines Links - Debugausgabe auf die Konsole

```
ProcessGeneralTagJSOUP — embedding:
curi http://www.meine—katze.de/Tier%C3%A4rzte
value /Maine_Coon
context link/@href
hop EMBED
Attribut: href="/Maine_Coon" | Key href | Value: /Maine_Coon | Link Owntext: Rasseporträt Maine Coon
Attribut: title="Maine Coon" | Key title | Value: Maine Coon | Link Owntext: Rasseporträt Maine Coon
Rasseporträt Maine Coon
Adding Link: <a href="/Maine_Coon" title="Maine Coon">Rasseportr&auml;t Maine Coon</a>
Final Perplexity: 8324519.834708901
DocumentPerplexity: 1.2596494866605083E7
LinkOwntextPerplexity: 5.816842896896651
Keywordscore: 0.01863814855281087
```

Die entsprechende Formel der Linkbewertung sieht wie folgt aus, wobei PP für Perplexität steht und die Gewichtungsfaktoren γ angegeben sind):

$$\gamma_d := 0.7, \gamma_{otx} := 0.3, \gamma_{linkkw} := 3, T_{otx} := 3000$$

$$perpscore_{link}(link_i) := \begin{cases} \Big(\big(\gamma_d \times PP_d(D_j) + \gamma_{otx} \times PP_{otx}(link_i)\big) \\ \times \big(1 - (\gamma_{linkkw} \times keywordscore(owntext_{link_i}))\big) \Big) \\ \text{falls } \Big(PP_{otx}(link_i) \leq T_{otx} \wedge PP_{otx}(link_i) \leq PP_d \Big) \\ PP_d(D_j) \times \big(1 - (\gamma_{linkkw} \times keywordscore(owntext_{link_i}))\big) \text{ sonst} \end{cases}$$

Ist der so erhaltene Perpscore kleiner als 0, z.B. wenn viele Keywords den Faktor negativ werden lassen, wird er auf 1 gesetzt (vgl Quelltext Listing A.4), sodass die Priorisierung in der Candidate-Chain problemlos erfolgen kann. Die Gewichtung ergibt sich aus Anpassungen nach diversen Testläufen, da z.B. Werte wie γ_{linkkw} oder T_{otx} (”Threshold owntext”) abhängig vom Sprachmodell bzw. der Keywordtabelle gewählt werden müssen. Die Gewichte γ_d und γ_{linkkw} wurden mit 70 zu 30 so gesetzt, dass die Dokumentperplexität zu 70 Prozent gegenüber 30 Prozent Linktextperplexität ausmacht. Der erzeugte Wert wird, wenn er den Schwellwert unterschreitet und geringer ist als die Dokumentperplexität, nochmals über eine Multiplikation mit dem gewichteten Keywordscore gestaucht.

3.7 Finale Methode

Damit der erweiterte HTML-Extraktor mit Bildextraktion funktioniert, muss im Crawljob in Heritrix zunächst sichergestellt werden, dass das Modul PerplexityPrioritizer von Remus und Biemann (unpublished) in der Candidate-Chain aktiviert ist und das Modul PerplexityLoggerDispositionchain in der DispositionChain aktiviert ist. Das Modul PerplexityProducer wird im Gegensatz zum fokussierten Crawl von Remus und Biemann nicht benötigt, da die Funktion vom erweiterten HTML-Extraktor abgebildet wird. Weiterhin sollten die Standardmodule HTML-Extraktor und HTTP-Extraktor abgeschaltet sein.

Wird nun eine URI als CURI aus der Frontier gezogen und durchläuft die Priorisierung, wandert diese in den erweiterten HTML-Extraktor. An dieser Stelle wird zunächst der Perplexitätswert für die Gesamtseite wie in Kapitel 3.5 bestimmt. Anschließend wird das Dokument in JSoup geparsed und über alle gefundenen Image Einträge iteriert. Die erkannten Bilder in der HTML-Seite werden geprüft, ob sie die geforderte Mindestgröße und im Imagetag ein Alt- oder Titleattribut haben. Ist keine Größenangabe vorhanden, oder ist das Bild zu klein, wird es verworfen. Ist keines der geforderten Attribute vorhanden, wird das Bild ebenfalls verworfen. Bilder, die diese Hürde genommen haben, werden einer Bewertung der Alt- und Titletexte, sowie ihrer Umgebung, unterzogen. Dabei wird eine Informationsdatei zum Bild erstellt (siehe Listing 3.3). Die Informationsdatei soll später bei der Auswahl und der Informationsauswertung zu jedem einzelnen Bild Hilfestellung liefern. Daher werden nicht nur die Endwerte, sondern auch Zwischenwerte gespeichert, um im Nachgang mehr über die Wirkungsweise eventueller Änderungen am Bewertungsschema im Vergleich mit einem vorherigen Lauf sagen zu können. In vorläufigen Tests hat sich jedoch gezeigt, dass die Aussagekraft des ALT-Tags in Verbindung mit der Umgebung am größten scheint. Daher wird eine zusätzliche Perplexitylogdatei für Bilder geführt. In der Logdatei werden im TSV-Format Perplexitywerte und Perplexitätsscorewerte (pure/weighted) für Alttext, Titltetext und den rudimentär zerlegten Bilddateinamen, gefolgt vom Speicherort des Bildes und der Infodatei festgehalten. Weiterhin werden die Spalten vom extrahierten Alt-, Title- und zerlegten Dateinamentext

und wählbaren, k-fachen Anzahl an Parent- und Siblingtagperplexitäten befüllt. Die so gewachsene Datei wird später für die Suche nach geeigneten Bildern genutzt und kann wegen dem offenen Format bequem mit Programmen wie Open/Libreoffice oder Microsoft Excel gelesen werden.

Listing 3.3: Auszug einer Infodatei

```
<urlimagefound>http://pictures-of-cats.org/is-the-kitten-healthy.html</urlimagefound>
<imageurltext>http:/pictures-of-cats.org/wp-content/uploads/2015/02/is-the-kitten-healthy-XXX.jpg</imageurltext>
<width-real>840</width-real>
<height-real>519</height-real>
<src-guesstring>pictures of cats org wp content uploads is the kitten healthy xxx</src-guesstring>
<src-guesstring-perplexity-pure>15786.4669645997</src-guesstring-perplexity-pure>
<src-guesstring-perplexity-keywordscore>0.22980618902626848</src-guesstring-keywordscore>
<src-guesstring-perplexity-weighted>4902.983530628476</src-guesstring-perplexity-weighted>
<alt>Is the kitten healthy?</alt>
<alt-length>22</alt-length>
<alt-perplexity-pure>830.8076364534564</alt-perplexity-pure>
<alt-keywordscore>0.01032622190107631</alt-keywordscore>
<alt-perplexity-weighted>805.070324420075</alt-perplexity-weighted>
<width>840</width>
<height>519</height>
<class>size-full wp-image-81022</class>
<style>border: 2px solid slategrey;</style>
<data-perplexityid>77</data-perplexityid>
<parent-0>Is the kitten healthy? Checklist infographic.</parent-0>
<parent-0-length>45</parent-0-length>
<parent-0-perplexity-pure>27580.695644500007</parent-0-perplexity-pure>
<parent-0-perplexity-keywordscore>0.01032622190107631</parent-0-perplexity-keywordscore>
<parent-0-perplexity-weighted>26726.28249426654</parent-0-perplexity-weighted>
<parent-0-perplexity-weighted-distance>0.0</parent-0-perplexity-weighted-distance>
<parent-1>A straightforward infographic which may assist some visitors when ... of a healthy cat.</parent-1>
<parent-1-length>965</parent-1-length>
<parent-1-perplexity-pure>963.9019553397098</parent-1-perplexity-pure>
<parent-1-perplexity-keywordscore>0.3731991858313779</parent-1-perplexity-keywordscore>
<parent-1-perplexity-weighted>1.0</parent-1-perplexity-weighted>
<parent-1-perplexity-weighted-distance>500.0</parent-1-perplexity-weighted-distance>
<parent-2>Is the Kitten Healthy? A straightforward infographic which may assist some visitors when ... healthy cat.</parent-2>
<parent-2-length>988</parent-2-length>
<parent-2-perplexity-pure>960.3054754640884</parent-2-perplexity-pure>
<parent-2-perplexity-keywordscore>0.3835254077324542</parent-2-perplexity-keywordscore>
<parent-2-perplexity-weighted>1.0</parent-2-perplexity-weighted>
<parent-2-perplexity-weighted-distance>1000.0</parent-2-perplexity-weighted-distance>
...
<sibling-0>Is the kitten healthy? Checklist infographic.</sibling-0>
<sibling-0-length>45</sibling-0-length>
<sibling-0-perplexity-pure>27580.695644500007</sibling-0-perplexity-pure>
<sibling-0-perplexity-keywordscore>0.01032622190107631</sibling-0-perplexity-keywordscore>
<sibling-0-perplexity-weighted>26726.28249426654</sibling-0-perplexity-weighted>
<sibling-0-perplexity-weighted-distance>0.0</sibling-0-perplexity-weighted-distance>
```

Sind alle Bilder auf der Seite geprüft worden, werden Links im HTML-Dokument erkannt und ausgewertet. Dabei wird der erzeugte Link mit zwei Perplexitätswerten versehen. Zum einen erhält der Link den über die Gewichtungsmechanismen generierten Perplexitätsscore für die zu besuchende URI. Gleichzeitig erhält er zusätzlich den anfangs berechneten Perplexitätswert als Extrainformationen für sein Attribut ”ViaURI”. Die ViaURI kann sinnbildlich wie die ParentURI aufgefasst werden, also die Seite in der der Link gefunden wurde. Dieser Mechanismus stellt die Kompatibilität zu Remus und Biemanns Perplexity-Prioritizer sicher, da das Verhalten des PerplexityProducer in der DispositionChain nachgeahmt wird. Ein gefundener Link, der so mit einer niedrigen Perplexität ausgestattet wird, wird früher angelaufen als ein Link mit hoher Perplexität.

Ist der Crawl abgeschlossen, wird gegenwärtig die erzeugte Bildperplexitäts-logdatei eingelesen und das geometrische Mittel zwischen ALT-Tag Perplexitätsscore und Parent-1 Perplexitätsscore gebildet und nach diesem sortiert. In vorläufigen Tests stellte sich eine Kombination beider Werte als zuverlässigste Quelle für in diesem Fall ”katzenhaltige” Bilder heraus. Zum einen ist in den Läufen der ALT-Tag meist mit dem Title-Tag identisch, in den allermeisten Fällen ist jedoch der Title-Tag gar nicht vorhanden und nur ein ALT-Tag existent. Weiterhin bildet der ALT-Tag in seiner Funktion das Bild textuell zu beschreiben, falls das Bild nicht geladen werden kann, schon von seiner Intuition her eine maßgebliche Beschreibungsquelle. Da in Zeiten von Breitbandinternet Ladezeiten für eingebettete Bilder nur noch wenig Relevanz haben, wird der Tag vernachlässigt. Da an dieser Stelle, wenn vorhanden, auch viel automatisiert erzeugter Content von z.B. bekannten Contentmanagementsystemen wie Wordpress hin-

terlegt wird, bringt zum anderen die Zuhilfename der Perplexitätsscores des ersten Parenttags weitere Sicherheit, dass sich das Bild im Themenumfeld befindet.

4 Evaluierungsmethode

Da sich dieses Thesis stark auf die Erkenntnisse von Remus und Biemann (unpublished) stützt und die Frage im Raum steht, ob die Vorgehensweise auch auf Bild zu Text Korpora übertragbar ist, wird in dieser Thesis analog zur Arbeit von Remus und Biemann ein Kneser Kney Smoothing Modell mit 5-Grammen sowohl zur Klassifizierung als auch zur Evaluation genutzt. Auch an anderen Stellen werden die von Remus und Biemann aufgezeigten Mechanismen teilweise übernommen oder adaptiert und angepasst. Die Evaluation wird in dieser Thesis bei 30 Millionen chronologisch extrahierten Sätzen abgeschnitten. Gesammelt wurden jedoch ca. 130 Mio. (fokussierter Crawl mit Keywords), 80mio (fokussierter Crawl) und 32 mio Sätze (unfokussierter Crawl).

Tabelle 4.1.: Genutzte Hardware und Anbindung

Crawl	**Sprachmodell**	**Crawling**	**Anbindung**
Fokussiert, mit Keywords	AMD-FX 8350, 16GB RAM	I7-2600k, 16GB RAM	50 Mbit
Fokussiert, ohne Keywords	I7-3790k, 32 GB RAM	selber Computer	100 Mbit
Unfokussiert	2xOpteron 2384, 16GB RAM	Xeon E5450, 8GB RAM	6 Mbit

Zu Verfügung stehen fünf Systeme, die wie in Tabelle 4.1 aufgeteilt werden.

4.1 Das Sprachmodell

4.1.1 Themenwahl

Zunächsteinmal wird ein Thema benötigt, für das Bilder und Bildtexte gefunden werden sollen. Als breiter aufgestelltes Thema wird aus mehreren Gründen das Thema *Katzen* gewählt: was zunächst niedlich anmutet hat praktische Gründe unter anderem darin, dass Katzen als Tiere bereits von Biologen nach mehreren Kriterien systemisch eingeordnet wurden. Übertragen auf die Sprachtechnologie sind Word- und ImageNet Paradebeispiele einer verwendbaren Klassifizierung, sodass je nach Ergebnislage weitere Evaluierungen mit den erhaltenen Datensätzen auf diesen Datenbanken möglich sind. Weiterhin stellt die Thesis einen ersten Versuch dar, mehr Daten zu sammeln und zu Evaluieren, als dies im privaten Gebrauch von einer Person ohne weiteres möglich ist. Katzen sind daher für das geplante Minicrowdsourcing ein Thema, das eine höhere Motivation zur Fortsetzung der Bewertungsaufgabe beim Teilnehmer hervorruft, als z.B. Bilder von Insekten oder Käfern zu klassifizieren. Katzen sind nahezu rund um den Globus ein Bestandteil der Lebensumgebung von Menschen. Es ist daher ein Leichtes, eine Katze von einem Hund oder anderen Gegenständen zu unterscheiden, im Gegensatz zu einzelnen Katzenrassen oder anderen hierarchisch tiefergelegenen Klassen z.B. in der WordNet-Hierarchie [Deng et al. 2009]. Weiterhin erfordert die Bildmarkierung keine tiefe Sachkenntnis seitens des Annotators, außer Englisch zu verstehen [Rashtchian et al. 2010].

4.1.2 Sprachmodell: Erstellung und Vortests

Um ein Sprachmodell zu erstellen, welches in diesem Fall auf Katzen trainiert werden soll, werden eine Reihe handselektierter Seiten über Heritrix heruntergeladen, u.a. auch jene, die zur Keywordgenerierung in der Positivliste stehen. Um möglichst viele und neutrale Sätze zu erhalten, werden per Suchmaschine Katzenseiten mit möglichst viel Textinhalt und objektivem Schreibstil gesucht. Die per Suchmaschine gefundenen Seiten, werden zunächst über ein von Kohlschütter et al. (2010) bereitgestelltes Webinterface[1] darauf geprüft, ob diese mittels Boilerpipeextraktor extrahierbar sind. Der Boilerpipeextraktor wird genutzt, um unnützen Text zu reduzieren, das Modul ist jedoch problembehaftet, was die individuelle Prüfung einer Seite verlangt (siehe Kapitel 5.1.3 ”Stolpersteine Boilerpipeextraktor”). Ist der Aufbaustil der Seite für den Boilerpipeextraktor geeignet, wird die Seite in der Kandidatenliste geführt. Sind genug Seiten gefunden, wird die Kandidatenliste als Seed genutzt und Heritrix für den Download genutzt. Um aus den Seiten Sätze für das Modell zu gewinnen, werden die Textinhalte der Seiten für die Verarbeitung in mehreren Schritten angepasst:

```
...
<body>
  <p>This funny &amp;amp; cute cat hast more than 99 lives :) — THIS is awesome! I really like her.<p>
</body>
...
```

1. Die Seite wird in Heritrix durch einen HTML-Extraktor auf reinen Text reduziert:

```
This funny & cute cat has more than 99 lives :) — THIS is awesome! I really like her.
```

2. Der erhaltene Text wird in Heritrix durch das SentencewriterModul in Sätze aufgetrennt und abgespeichert. Ein Satz muss dabei aus mindestens 5 Tokens bestehen, sonst wird er verworfen:

```
This funny & cute cat has more than 99 lives :) — THIS is awesome!
I really like her.
```

```
This funny & cute cat has more than 99 lives :) — THIS is awesome!
```

3. In dieser Form bereits auf der Platte liegend, werden die erhaltenen Sätze über den LT.Segmenter aus der LT-Suite auf Github[2] normalisiert und gefiltert. Die gesetzten Einstellungen sehen dabei vor, dass sämtliche Buchstaben in lowercase gesetzt und Zahlen zu einem Symbol umgeformt werden. Im Anschluss werden Satzzeichen und die erstellten Symbole entfernt. Mehr als ein Leerzeichen und andere nicht wortzugehörige Zeichen wie Tabulator u.ä. werden auf ein Leerzeichen reduziert:

```
this funny cute cat has more than lives this is awesome
```

Die so aufbereiteten Sätze aus den heruntergeladenen Seiten werden für die Erstellung des Sprachmodells genutzt. In diesem Fall ergibt sich ein kleines Korpus von ca. 20K Sätzen, das zum Großteil aus Seiten erzeugt wurde, die von Katzenrassen, Katzen allgemein und Katzenverhalten stammen. Für die Evaluation werden die Sätze gefiltert und normalisiert, anschließend zufällig gemischt und in ein Trainingsset von 14K Sätzen und ein Testset von 6K Sätzen eingeteilt. Aus dem Trainingsset wird nun ein Sprachmodell erzeugt, welches für die drei geplanten Crawls zur Bewertung der Texte herangezogen wird.

[1] `http://boilerpipe-web.appspot.com` (aufgerufen Juli 2015)
[2] `https://github.com/tudarmstadt-lt/lt.kd` (aufgerufen August 2015)

Tabelle 4.2.: Verteilung der gecrawlten Seiten im Vortestlauf

Seed URL	Anzahl heruntergeladener Seiten
http://www.catchannel.com	924
http://www.cnet.com	71
http://www.meine-katze.de	3
http://www.heise.de	2

Um das vorläufig final erhaltene Trainingsmodell im kleinen Feld vorab zu testen, wird ein fokussierter Crawljob definiert, der als Seed Seiten fernab des Katzenthemas und Seiten bezogen auf das Katzenthema enthält. Zusätzlich werden weitere Seeds platziert, welche die gleichen Themen beinhalten, jedoch sprachfremd sind. Die maximale Anzahl von herunterzuladenen Dokumenten wird auf 1000 begrenzt. Der Crawl wird an die SeedURLs gebunden, sodass diese nicht verlassen werden dürfen und höchstens Unterseiten der gleichen Domain angesurft werden können. Der Crawler hat so nun die Möglichkeit im Lauf sich immer für einen der Seeds oder der erkannten Seiten anhand des Sprachmodells zu entscheiden. Es wird weiterhin Multithreading abgeschaltet, um Seiteneffekten vorzubeugen. Zur Parameteroptimierung wird der Crawl bei Bedarf nun wiederholt und das Trainingsmodell erweitert, sowie die Bucketgrenzen angepasst, bis ein zufriedenstellendes Ergebnis vorliegt. Um Overfitting einzudämmen wird der Crawl nun nochmal erweitert konfiguriert, sodass dem Crawler weitere potentiell themenfremde Seiten in seinem Scope vorliegen. Ist das Ergebnis mit gleichen Bucketgrenzen immernoch zufriedenstellend, kann zum eigentlichen Crawlen übergegangen werden. Anhand des Beispiels kann man gut sehen, dass zunächst einmal der Sprachwechsel die größte Barriere darstellt (siehe auch Perplexitätswerte in Listing 3.2) und im nächsten Schritt das Trainingsmodell recht gut greift (siehe Tabelle 4.2).

4.2 Die Crawlingläufe

Es werden drei Crawlingläufe mit unterschiedlichen Einstellungen gefahren:
a.) Der erste Lauf ”ufc” ist ein unfokussierter Lauf, der lediglich Bilder einsammelt, gewichtet und wertet wie in den anderen Läufen. Dabei wird die neu hinzugekommene Keywordamplifikation bereits benutzt. Der Crawl läuft jedoch ohne Priorisierung in der DispositionChain und kann somit einer Breadth-First-Strategie zugeordnet werden.
b.) Der zweite Lauf ”fcnokw” ist ein fokussierter Crawl, der sehr ähnlich zu dem in der Arbeit von Remus und Biemann (unpublished) beschriebenen Lauf konfiguriert ist. Eine Linkpriorisierung erfolgt in der DispositionChain, was eine Kategorisierung nach Chakrabarti et al. (1999) als Best-First-Strategie bedeutet.
c.) Der dritte Crawl ”fcwk” ist der erweiterte fokussierter Crawl, welcher Linkextraktion über das neu geschriebene Modul betreibt. Die Linkpriorisierung erfolgt bereits in der FetchChain, dennoch wäre auch hier wieder die Kategorie Best-First-Strategie zu wählen.

Gemeinsamkeiten: Um eine Basis für den Vergleich mit dem fokussierten Crawl und dem erweiterten fokussierten Crawl zu haben, werden Daten benötigt, die gegen das Testmodell geprüft werden können. Um Sätze auf den Seiten des Crawls festzuhalten wird das Modul SentenceWriterHTML in allen Läufen einkonfiguriert, sodass Sätze aus den Webseiten extrahiert- und als gepackte Textdatei im Jobverzeich-

nis getrennt durch einen Zeilenumbruch chronologisch abspeichert werden. Als HTML-Extraktor wird das in dieser Thesis erweiterte Modul eingebunden. Weiterhin werden alle Läufe an das Sprachmodell angeschlossen. Über die Konfigurationsparameter *examineImagesWithJsoup=true* wird in allen Läufen veranlasst, dass der Extraktor mittels JSoup Bilder erkennt und auf der Platte ablegt. Dabei werden nur Bilder mit einer Mindestbreite von 512 und einer Mindesthöhe von 384 Pixeln zugelassen. Weitere Einstellungen bleiben jedoch individuell.

4.2.1 Testumgebung

Um die gleiche Ausgangsbasis zu schaffen, erhalten alle evaluierten Crawlingläufe die gleiche Seeed Liste. Diese enthält fünf Seiten:

- www.dmoz.org/Recreation/Pets/Cats
- cattime.com
- www.catster.com
- user.xmission.com/~emailbox/catstuff.htm
- cats.about.com

Weiterhin werden sogenannte Reject-Surts definiert. Es handelt sich dabei um Filter, die verhindern, dass Seiten der in den Reject-Surts definierten URLs angelaufen werden. Hier sind in Probeläufen Seiten aufgefallen, in denen sich der Crawler öfters verfängt. Generell haben besonders große Seiten, die sehr viele Unterseiten mit unterschiedlichen URLs beinhalten, das Potential einen Crawler an sich zu binden und einen freien Lauf zu verhindern. Gleiches gilt für besonders oft verlinkte Seiten mit unterschiedlichen Parametern innerhalb der URL wie z.B. Google Maps. Alleine die Vorstellung, wieviele Anfahrtsbeschreibungen auf Webseiten mit einer Karte über die Google Maps API mit unterschiedlichen Koordinaten hinterlegt werden, kann dies verdeutlichen. Über die Reject-Surts werden daher zum Start ebenfalls fünf Seiten ausgesperrt:

- wikipedia.org
- amazon.com
- amazon.co.uk
- amazon.de
- maps.googleapis.com

4.2.2 Unfokussierter Crawl

Hier wird keine Perplexität für das HTML-Dokument an sich generiert. Weiterhin wird die Standardheritrix Linkextraktion genutzt und keine Priorisierung von Links durchgeführt.

4.2.3 Fokussierter Crawl

Der fokussierte Crawl verwendet zwei Module von Remus und Biemann (unpublished). Der Perplexity-Producer wird auf die Bucketgrenzen aus dem vorläufigen Testlauf eingestellt. Um die vom Perplexity-Producer angehängten Perplexitätswerte am URI-Objekt auszuwerten, wird In der Candidate-Chain der PerplexityPrioritizer aktiviert, sodass eine Priorisierung der Warteschlangen stattfindet (Fokussierung). Zur Fokussierung wird legidlich die Perplexität des Dokuments genutzt.

4.2.4 Erweiterter fokussierter Crawl mit gewichteter Linkextraktion

In diesem Lauf wird der PerplexityProducer abgeschaltet und die Linkextraktion im erweiterten HTML-Extraktor aktiviert. Bei der Linkextraktion erfolgt somit die Linkbewertung direkt innerhalb des Prozessors samt Keywordgewichtung. Die Priorisierung der Warteschlange wird weiterhin vom PerplexityPrioritizer in der Candidate-Chain vorgenommen.

4.3 Crowdsourcing Ansatz im privaten Umfeld

Nach Abschluss des Crawlings wird anhand des Bildperplexitätslogs eine Rangliste gebildet, sortiert nach dem geometrischen Mittel zwischen ALT-Tag Perplexitätsscore und Parent-1 Perplexitätsscore. Die N besten Ergebnisse werden den Annotatoren vorgelegt. In diesem Fall ist N=2500 und die obersten 2500 Ergebnisse werden zur Evaluation den Annotatoren vorgelegt. Dafür wird ein kleines Crowdsourcing im privaten Umfeld durchgeführt mit jeweils drei Annotatoren pro Bild. Um die Evaluierung zu unterstützen, steht eine selbstgeschriebene, simple, in PHP geschriebene Webplattform für die Annotatoren bereit. Um den Annotatoren die Aufgabe etwas zu erleichtern, wurden über JQuery[3] zusätzlich Hotkeys eingebaut, sodass der Vorgang per Tastatur beschleunigt werden konnte. Die Bildbreite wurde auf ein Maximum von 600 Pixeln festgelegt, damit die Seite auf einem Tablet komplett hochkant zu lesen ist und die Bequemlichkeitshürde sinkt. Da noch kein Goldstandard existierte, wurde zur Kompensation die Annotation per Skript und Cronjob überwacht, sowie die Apache access.log gemonitort. Bei Unsicherheiten wurde stets Kontakt mit den Annotatoren in der Anlernphase gehalten.

Um die Klassifikation eines Bildes vorzunehmen, bekommt der Annotator ein Bild gezeigt, wobei unterhalb des Bildes der dem Bild zugehörige Alttag angezeigt wird. Der Annotator soll nun das Bild und den Text in Relation bringen. Dabei kann ein Bild in die Themenbereiche:

1. Katze (Cat)
2. Haustier (Pet)
3. Säugetier (Mammal)
4. Lebewesen
5. Sonstiges

einkategorisiert werden. Und der Text in

1. Beschreibt das Bild
2. Passt thematisch zum Bild
3. Irrelevant
4. Kontraproduktiv

Im Idealfall ist das Bildthema einer Katze zugeordnet mit einem bildbeschreibenden Text.

Die erhaltenen Annotationen werden innerhalb einer Textdatei gespeichert und über ein Flag im Dateisystem sichergestellt, dass kein Annotator das selbe Bild zweimal bewertet oder ein bereits fertigannotiertes Bild angezeigt bekommt. Für die Evaluation werden die Daten per Pythonscript ausgelesen und über die im Dateinamen einkodierten Indizes zugewiesen und eine TSV-Datei mit den Ergebnissen erstellt. Anhand der so erstellten Daten wird die Auswertung nach Mehrheitsentscheid, sprich mindestens 2 Annotatoren haben zu einer Frage die gleiche Meinung, und nach Einstimmigkeit (drei Annotatoren gleicher Meinung) durchgeführt. Die in den Bildnamen enthaltenen Indizes werden weiterhin genutzt, um eine Übersicht über den chronolgischen Crawlverlauf der Bilder zu bekommen.

[3] `https://jquery.com` (aufgerufen August 2015)

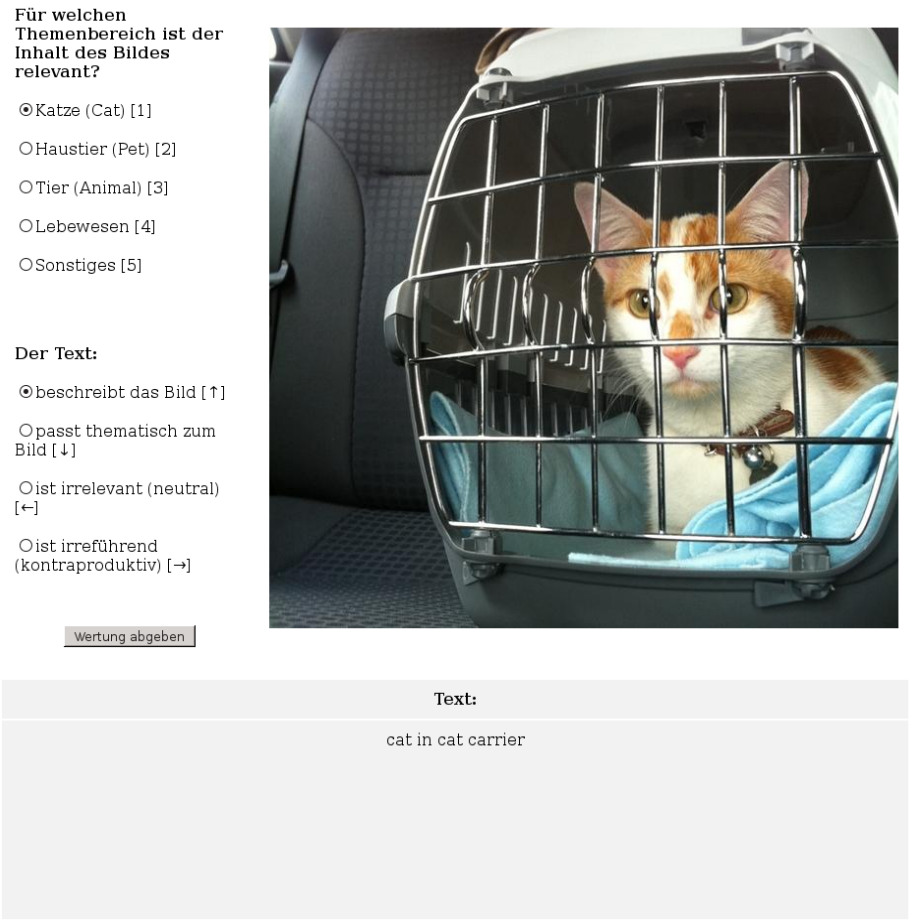


Abbildung 4.1.: Beispiel WebGUI Annotation (Bild: pawesomecats.com)

5 Ergebnisse & Diskussion

5.1 Statistiken, Beispiele

Nach rund einer Woche Crawling stehen die Daten auf den Systemen bereit. Neben der Datenbeschaffung dauerte die Bewertung der Sätze weitere zwei Tage. Im Vergleich zu Remus und Biemanns Lösung schneidet der Crawler jedoch deutlich langsamer ab, da bereits durch die Bildextraktion das Sprachmodell, trotz im neuen Modul einprogrammierter Cachingmechanismen, um ein vielfaches stärker belastet wird als das pro URI einmalige Abfragen der Perplexität wie bei Remus und Biemann. Die Linkextraktion im Keywordcrawler verstärkt diesen Effekt, sodass das Sprachmodell zum Flaschenhals im Crawling auf schwächeren Computern wird. Am schnellsten hingegen bleibt der unfokussierte Lauf, der zum einen die stärkste Maschine in Sachen Arbeitsspeicher, zum anderen die schnellste Anbindung inne hat. Sprachmodell und Crawler mussten sich zwar den selben Computer teilen, fehlende Priorisierung und der Wegfall der Netzwerklatenz beim Befragen des Sprachmodells wirken sich hier jedoch positiv aus.

Tabelle 5.1.: Crawlstatistiken

	Unfokussierter Crawl	Fokussierter Crawl	Fok. Crawl mit Keywords
Sätze	32.547.081	81.862.711	131.042.156
Tokens*	446.134.466	449.441.647	507.342.291
Hosts besucht	1.944.697	753.907	510.403
URIs processed	2.788.429	3.219.361	2.947.727
Dokumentvolumen gesamt**	100.022 MB	99.571 MB	100.004 MB
URIs/sec	16,79	3,35	6,38
KB/sec	661	171	254

* = nach Segmentierung bei Limit von 30 Millionen Sätzen
** = ohne Bilddateien

Für die Evaluation wurde ein Limit von 30 Millionen Sätzen gesetzt. Die Anzahl der extrahierten Sätze liegt jedoch weit darüber (siehe Tabelle 5.1). Ein erster Hinweis auf die Fokussiertheit findet sich in der Zeile ”Hosts” besucht. Der unfokussierte Lauf war gegenüber dem Fokussierten Crawl auf mehr als zweieinhalb mal sovielen Hosts unterwegs, gegenüber dem Keywordcrawl mehr als das Dreieinhalbfache.

5.1.1 Evaluation Crawling: Sätze

Um zunächst die Bilder außen vor zu lassen und den textuellen Einfluss auf die Crawlingläufe bewerten zu können, werden die aus jedem Crawl extrahierten Sätze durch den bereits bekannten LT-Segmenter

zu vorgefilterten und normalisierten Tokens aufbereitet. Nachdem durchängige Kleinschreibung sichergestellt ist und das Entfernen von Zahlen und Sonderzeichen durchgeführt wurde, werden Satzpakete zunächst in kleinen Tausenderschritten, dann immer größer, geschnürt. Die Sätze werden so angeordnet in den Sprachmodellclient eingelesen und gegen das auf den Testingsätzen trainierte Sprachmodell auf Perplexität geprüft. Da die Sätze chronologisch geordnet sind und als Folge daraus auch die Paketgrößen die Chronologie wiedergeben, kann anhand der Kurve nachvollzogen werden, wie gut die Fokussierung auf bestimmte Textstile während des Crawls funktioniert. Oftmals ist hierbei zu beobachten, dass ein unfokussierter Lauf in der Anfangsphase teilweise fokussierte Läufe in Perplexität unterbieten kann. Auch an dieser Kurve in Abbildung 5.1 sieht man gut, dass die eingebundenen Trainingsdaten bis ca. 20K Sätze einen nahezu identischen Lauf haben.

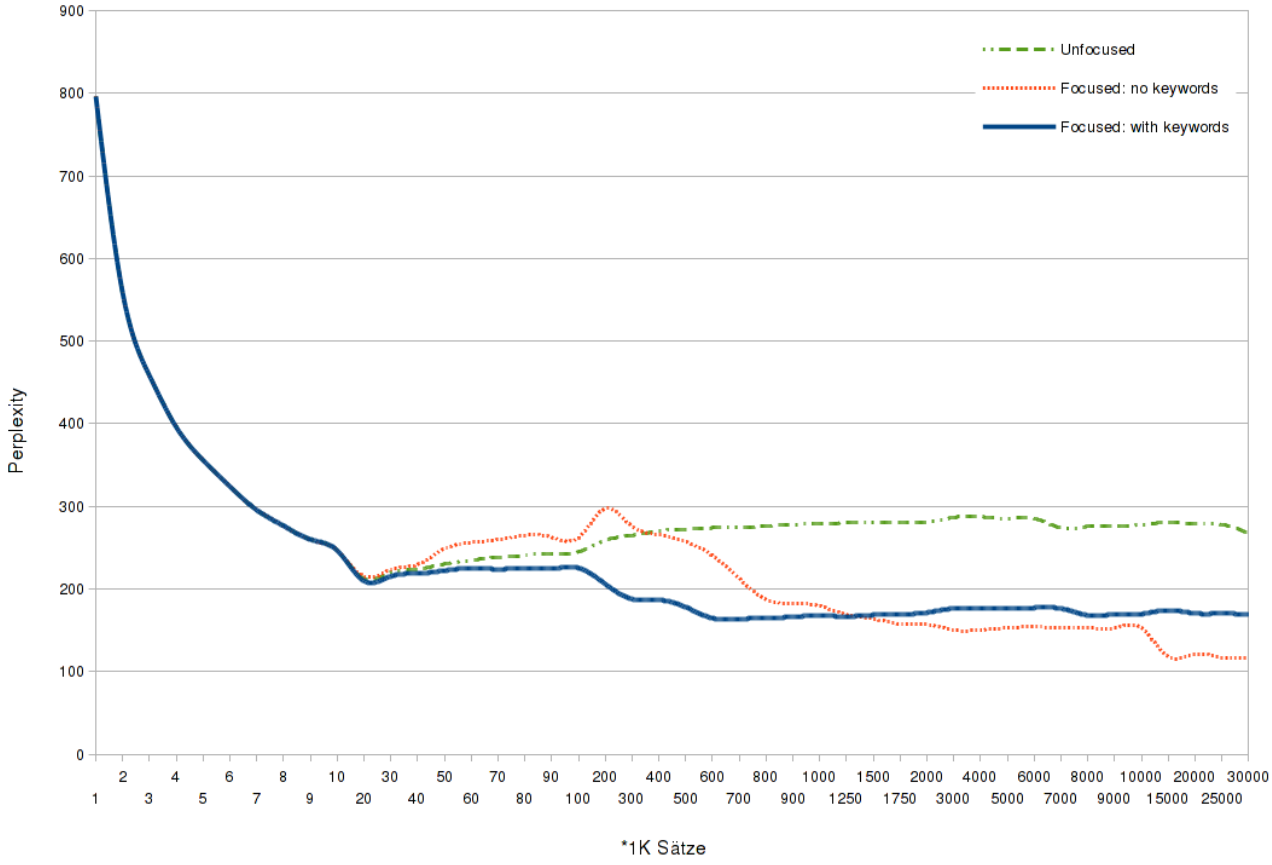


Abbildung 5.1.: Perplexitäts Plot: Vergleich der Satzperplexitäten für die verschiedenen Läufe

Zum einen liegt dies an den ca. 14K Sätzen im Trainingsmodell welche hier dazugegeben werden, zum anderen an der Seedliste, sodass beim Crawlstart alle Crawler erstmal den gleichen Links folgen müssen. Ab ca. 30K Sätzen treten erste Trennungserscheinungen auf. Was an den Graphen weiterhin auffällt, ist dass der fokussierte Crawler mit Keywordgewichtung offensichtlich etwas direkter den Katzenseiten folgt, sodass er bis eine Millionen Sätze in Führung bleibt und bis ca. zehn Millionen ein ähnlich gutes Niveau wie der normalfokussierte Lauf hält. Erst ab 10 Millionen Tokens scheint der fokussierte Lauf einige sehr relevante Seiten gefunden zu haben, die seine Perplexitätswerte deutlich verbessern. Die dort gefundenen Seiten weisen auch eine gewisse Kongruenz zur Bildextraktion auf, denn der fokussierte Crawl findet gegen Ende des Laufes noch einmal einige für gut befundene Katzenbilder. Insgesamt erscheint dies mit den Unterschieden in der Methode erklärbar. Der fokussierte Crawler mit Keywords arbeitet eher zukunftsorientiert, d.h. er sieht einen Link und bewertet diesen nicht nur aufgrund der aktuellen Seite, sondern auch vorwärtsgewandt auf den Verdacht hin, eine Katzenseite zu finden. Der

normale fokussierte Lauf hingegen bewertet den Link erst, wenn er ihn schon besucht hat und kann so entscheiden, ob jetzt gesehene Seite zum Thema passt. Die Kriterien ergeben sich hier eher aus der Stammbaum eines Links.

5.1.2 Evaluation Crawling: Bilder

Nach Abschluss der drei Crawlingläufe werden die extrahierten Bildeinträge für die Evaluation aufbereitet und alle Bilder weggeschnitten, deren berechnetes geometrische Mittel von Alt-Perplexitätsscore und Parent-1-Perplexitätsscore nicht zwischen den ausgwählten Grenzwerten liegen. In diesem Fall betragen die Grenzwerte nach unten 2 und nach oben hin 70.000. Bei vorläufigen Tests mit einem Limit innerhalb der formatbereinigten TSV-Dateien zeigte sich, dass die "Katzendichte" ab diesen Werten deutlich nachlässt. Die Werte sind jedoch in der Gesamtkette zu betrachten und müssten bei anderen Läufen und Themen entsprechend angepasst werden. Nach diesem Schnitt erfolgt eine Filterung der Bilder nach URL. In der Methode werden Bilder mehrfach vorgehalten, wenn andere Seiten die gleiche Bild-URL einbinden oder unterschiedliche Texte das Gleiche Bild begleiten. An dieser Stelle wird so gefiltert, dass das zuallererst erfasste Bild in der Liste verbleibt. Das soll zum einen Redundanz in der Bildmasse verhindern, weiterhin aber auch die Annotatoren vor doppeltem Content schützen. Durch die Sortierung der Bildliste nach dem erzeugten Geometrischen-Perplexitätswert wären die gleichen Bilder oftmals direkte Nachbarn in der Rangliste.

Tabelle 5.2.: Verarbeitung der extrahierten Bilder für die Evaluation

	Initial	Formatfilter	Schnitt	Unique
Unfokussiert	259801	201974	9762	6871
Fokussiert	402912	347871	48515	11347
Fokussiert mit Keywords	587182	553564	91947	39218

Die so erhaltenen Bild-zu-Text-Kombinationen werden für die Annotation über das Webinterface vorbereitet. Dazu werden die jeweils 5000 besten Ergebnisse eines Crawls herangezogen, indem die Rangliste einfach nach 5000 Elementen abgeschnitten wird. Die Bennenung erfüllt weiterhin den Zweck, die Zuordenbarkeit der Bilder zu erhalten und gleichzeitig die nötigen Informationen für verschiedene Evaluationsscripte im Dateinamen vorzuhalten. Die Annotation dauert ca. 12 Tage, wird im privaten Umfeld organisiert und nach ca. 2500 Bildern beendet. Die Annotation umfasst eine persönlich betreute Anlernphase und wird über eine Hilfeseite mit vielen Beispielbildern unterstützt. Jedes Bild wird von drei Annotatoren bewertet. Die Ergebnisse werden nach "Einstimmig" (alle Drei Annotatoren sind einer Meinung) und "Mehrheitsentscheid" (zwei Annotatoren überstimmen einen) ausgewertet.

Nach Abschluss der Annotation ergibt sich für den Bereich "Bildthema" folgendes Bild: Der unfokussierte Crawl bezieht die meisten Bilder aus der Kategorie "Sonstiges". Ähnlich viel lädt der fokussiete Crawl herunter, kann die Quote an Katzenbildern jedoch nahezu verdoppeln. Der keywordunterstützte fokussierte Crawl zieht jedoch mit 1757 Bildern (Mehrheitsentscheid, 1536 bei kompletter Einigkeit der Annotatoren) mit Abstand die meisten Katzenbilder in die Top 2500 der heruntergeladenen Bilder - das entspricht Faktor 2,08 zum Fokussierten und 3,52 zum Unfokussierten (siehe Abbildung 5.2).

Tabelle 5.3.: Anteil Katzenbilder auf ca. 2500 Heruntergeladene

	Einstimmig (3)	Mehrheit (>1)
UFC	0.1846	0.1993
FCNOKW	0.2990	0.3245
FCWK	0.6122	0.7002

Absolut betrachtet bedeutet dies eine Katzenquote von ca. 61% bei einstimmigen- und ca. 70% bei Mehr-

heitsergebnissen auf 2509 Bilder (siehe Tabelle 5.3). Es wird allerdings auch deutlich, dass je fokussierter der Lauf in Bezug auf Bilder ist, desto uneiniger sind sie sich die Annotatoren. So hat der unfokussierte Crawl 19 Bilder, bei dem sich die Annotatoren auf kein Bildthema einigen konnten, der Fokussierte dagegen 63 und der Fokussierte mit Keywords sogar 90 (siehe Tabelle A.1)).

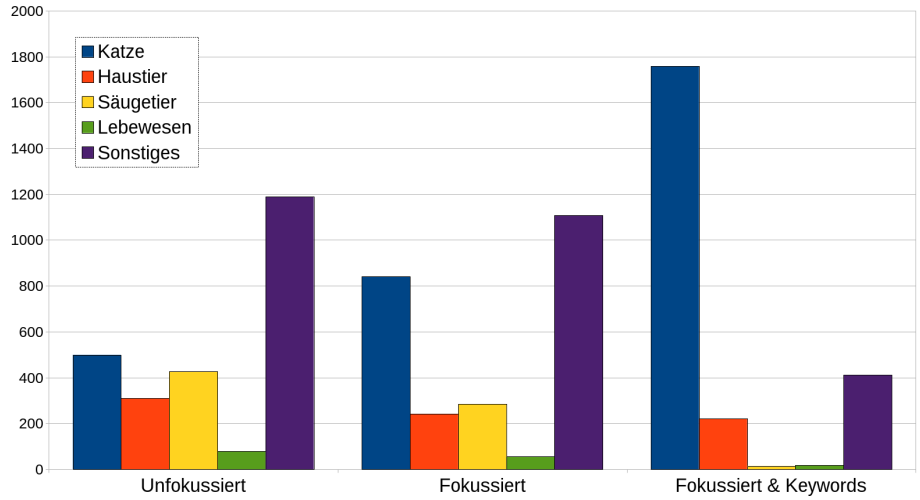


Abbildung 5.2.: Bildbewertungen in Summe der einzelnen Läufe (Daten siehe Tabelle: A.1)

Um die Einigkeit der Annotatoren zu Bewerten wurde ein Script geschrieben, welches allen Annotatoren eines Laufes alle möglichen Kombinationen aus Annotatorenpaaren bildet. Sofern in dem Lauf eine Annotation mindestens eines gemeinsamen Bildes stattgefunden hat, wird aus diesen Annotationen ein Cohen´s Kappa κ-Wert gebildet [Cohen 1968]. Alle so paarweise erzeugten κ-Werte eines Laufes werden im Anschluss gemittelt. Die auf diese Weise erhaltenen Werte untermalen die zunehmende Unstimmigkeit, je geringer die Unterschiede in den zu bewertenden Bildern sind. Wird der unfokussierte Crawl hier mit einem gemittelten κ-Wert von 0,85 für das Bildthema bewertet, bedeutet dies, dass die Annotatoren eine Übereinstimmungsrate haben, die deutlich von zufälligen Übereinstimmungen abweicht. Auch der fokussierte Crawl mit 0,7 befindet sich noch im guten Bereich, beim fokussierten Keywordcrawl hingegen nimmt mit 0,65 die Uneinigkeit noch etwas weiter zu (siehe Tabelle 5.4).

Ganz anders hingegen sieht es zunächst bei der Zuweisung der Relation von Text zu Bild aus. Entsprechend der Aufgabe, Freitext in Relation zum Bild zu bringen, sind sich hier die Annotatoren sehr uneinig, sodass die gemittelten κ-Werte in allen Läufen mit 0,27, 0,28, 0,23 sehr niedrig sind. Eine greifbare Aussage ist damit für die Text-zu-Bild-Zuweisung zunächst nicht zu treffen, denn die Übereinstimmungen könnten auch zufällig entstanden sein. Das mag mitunter auch in der Aufgabe begründet sein, dass die Vielfalt der Texte im Netz unendlich groß erscheint und ein Gold-Standard für eine offene Aufgabe zu finden komplex ist. Das Interannotatoragreement nimmt die Statistik der Textbewertungen bereits voraus. Hier zeigt sich ein uneinheitliches Bild, sodass davon auszugehen ist, dass möglicherweise die Aufgabe zu schwer war, oder ähnlich wie bei Deng et al. (2009) die Annotation schwieriger wird, je tiefer man sich in einer Hierarchieebene befindet. Werden die annotierten Daten genauer betrachtet, zeigt sich, dass die schlechten Werte tendenziell von der Hierarchieebene herrühren. Daher wurden zu den initialen κ-Werten zusätzlich gemittelte κ-Werte für zusammengefasste Kategorien erzeugt. Die Intention der Abstufung zwischen ”beschreibt das Bild” und ”passt zum Text” ist, eine Einordnung in die Textverwertbarkeit im Sinne der Bild-Text Korpusbildung zu ermöglichen. Die weiteren Abstufungen ”irrelevant” und ”kontraproduktiv” sollen Rückschlüsse auf das Crawlingverhalten anhand der Textbewertung zulassen. Für das angestrebte Ziel, lediglich Katzen zu finden, wären daher auch breiter gefasste Kategorien und damit eine flachere Hierarchie ohne Informationsverlust möglich.

Aus diesem Grund wurden für die angepasste Rechnung folgende Kategorien zusammengelegt:
a.) Bild reduziert 1: Katze, Haustier, (Säugetier + Lebewesen), Sonstiges
b.) Bild reduziert 2: Katze, Haustier, (Säugetier + Lebewesen + Sonstiges)
c.) Text reduziert 1: (Beschreibt das Bild + Passt thematisch zum Bild) + Irrelevant + Kontraproduktiv
d.) Text reduziert 2: (Beschreibt das Bild + Passt thematisch zum Bild) + (Irrelevant + Kontraproduktiv)
(siehe Tabelle 5.4)
Mit diesen zusammengelegten Kategorien bleiben die katzenrelevanten Bildinformationen erhalten und man kann erkennen, dass die Übereinstimmung zwischen den Annotatoren deutlich zunimmt. Die Werte spiegeln nebenbei gut aufgetretene Probleme bei der Bewertung wieder, in der z.B. einige Annotatoren Menschen als Lebewesen und andere Annotatoren als Säugetiere klassifizierten. Andere wiederum hielten manche Bilder trotz Menschen im Bild für Sonstiges. Für das Finden von Katzen in Bildern sind diese drei Kategorien jedoch irrelevant, sodass die jeweils zweite reduzierte Stufe (b und d) als Messpunkt genutzt werden kann. Es zeigt sich weiterhin, dass die Uneinigkeit der Annotatoren weniger von der Trennung zwischen ”beschreibt das Bild” und ”Passt thematisch zum Bild” herrührt, als von der Uneinigkeit darüber, ob ein Bild irrelevant oder kontraproduktiv ist.

	Unfokussiert	Fokussiert	Fokussiert mit Keywords
Annotatoren	11	14	13
Annotatorenpaare	25	31	34
κ-Wert Bild gemittelt	0.815	0.704	0.654
κ-Wert Bild reduziert 1	0.824	0.754	0.662
κ-Wert Bild reduziert 2	0.930	0.895	0.726
κ-Wert Text gemittelt	0.276	0.290	0.238
κ-Wert Text reduziert 1 gemittelt	0.400	0.295	0.280
κ-Wert Text reduziert 2 gemittelt	0.443	0.379	0.403

Tabelle 5.4.: Cohens Kappa Verteilung für die einzelnen Läufe bei der Bildthemabewertung

Für Uneinigkeiten bei der Bewertung von Text zu Bild kann z.B. bereits der Umstand ausschlaggebend sein, dass Annotatoren Szenen anders bewerten, wenn Text im Bild erscheint. Eine Grafik, deren Text eine Kopie eines Textes ist, der im Bild zu sehen ist, wird öfters unterschiedlich bewertet. Gleiches gilt für Text innerhalb eines Fotos: so sind sich die Annoatoren z.B. bei Abbildung 5.3 (c)[1], nicht einig, ob ”001 crazy cat lady dreamalittlebigger” ein thematisch passender Text, oder eher irrelevant ist. Ein ähnliches Beispiel bietet das Bild in Abbildung 5.3 (b)[2], denn obwohl die Katze eine Sprechblase hat, lautet der Subtext mit ”Cat hisses and bites for no reason” anders - auch hier ist kein einstimmiges Ergebnis zu erzielen. Manchmal ist auch nicht klar, wo der Schwerpunkt einer Illustration liegt: Die Annotatoren von Abbildung 5.3 (d)[3] sind uneins darüber, ob hier die Kralle der Katze oder doch nur ein anatomisches Erklärschema gezeigt wird. Die Mehrheit entschied sich für Letzteres. Bei den ”Flat Cats” aus Abbildung 5.3 (a)[4] ist bereits das Bild nicht übereinstimmend annotiert. Die Meinungen reichen von ”Cat” bis zu ”Sonstiges”. Der zum Foto angezeigte Text, der dieses als Pressefoto ausgibt, sorgt trotz Lernphase für Verunsicherung, ob es sich thematisch um Katzen (z.B anhand des Bandnamens oder der Katzensiluette auf der Basedrum) handelt, oder ob das gezeigte Bild nur ein Pressefoto darstellen soll.

[1] Bildlink: `http://www.dreamalittlebigger.com/post/crazy-cat-lady-shirt.html` (aufgerufen August 2015)
[2] Bildlink: `http://pictures-of-cats.org/My-Cat-Bites-and-Hisses-for-No-Reason-Advice-Please.html` (aufgerufen August 2015)
[3] Bildlink: `http://pictures-of-cats.org/the-basics/cat-anatomy-facts-for-kids.html` (aufgerufen August 2015)
[4] Bildlink: `http://www.flatcatsmusic.com` (aufgerufen August 2015)

(a) "The Flat Cats Press Photo"

(b) "Cat hisses and bites for no reason"

(c) "001 crazy cat lady dreamalittlebigger"

(d) "cats claw"

Abbildung 5.3.: Uneinigkeit bei Text Annotation, wenn Text im Bild vorhanden ist

Annotiert ein Mensch, wird der zur Bewertung gelesene Text individuell interpretiert werden, sobald kleine Ambivalenzen auftauchen oder Texte über mehreren Ebenen wie z.B. Ironie erfassbar sind. Der Text hat also u.a. auch Auswirkungen auf die Bewertung des Bildes. So ist die Verteilung der Textzuordnungen in Abbildung 5.4 in Anbetracht der ermittelten κ-Werte trotzdem nicht so klar und deutlich verwertbar wie die Ergebnnisse der Bildbetrachtung in Abbildung 5.2.

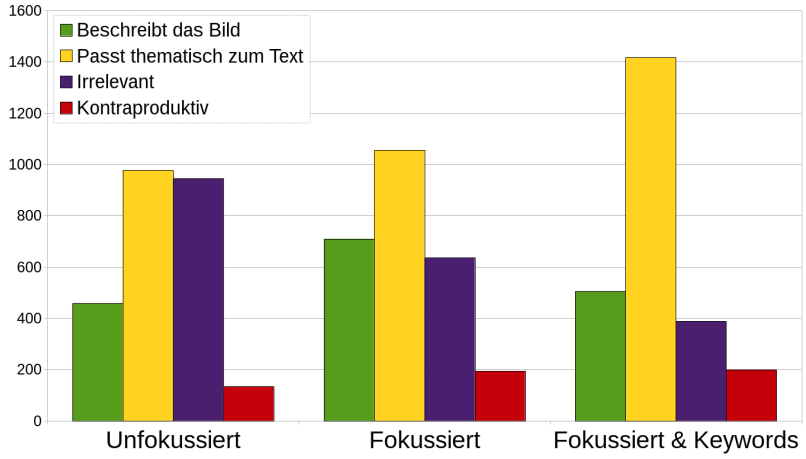


Abbildung 5.4.: Text- zu Bildbewertungen der einzelnen Läufe in Summe (Daten siehe Tabelle: A.2)

Betrachtet man nun den Gesamtlauf, stellt sich, neben der Annoatationsfrage, noch die Frage nach dem Verhalten des Crawlers bezüglich der Bildbeschaffung. Hierfür wurden die annotierten Bilder über den Downloadindex eines Bildes in Relation mit ihrer Bewertung gebracht. Abbildung 5.5 (quer) zeigt dazu das Crawlverhalten anhand der chronolgisch sortierten und mit Mehrheitsvotum annotierten Bilder. Die Skalen geben die einstimmige Bewertung der Annotatoren numerisch wieder. Der Wert 5 entspricht dabei "Katze", 4 "Haustier", 3 "Säugetier", 2 "Lebewesen" und 1 "Sonstiges".

Wie man erkennen kann, bezieht der unfokussierte Crawl über den gesamten Zeitraum das Gros seiner Bilder aus *Sonstiges*, die Kategorie *Katze* wird verstärkt zu beginn des Crawls bedient. Stellt man die Anzahl von 200K heruntergeladenen Bildern den fast 2 mio. besuchten Hosts gegenüber, so erscheint die Ausbeute nicht sehr groß. Anders sehen die beiden fokussierten Läufe aus: hier kann der normal fokussierte Crawl gegenüber dem unfokussierten Lauf mehr Bilder in der Kategorie *Katze* gewinnen, hat aber trotzdem noch eine gewisse Affinität zu *Sonstiges*. Viel wichtiger ist jedoch, dass man sehen kann, dass der fokussierte Lauf im späten Verlauf noch einmal auf eine gut mit geeigneten Bildern befüllte Menge von Seiten gestoßen ist (vgl. Perplexitätsknick bei 10 mio. Sätzen in Abbildung 5.1). Das zeigt, dass der fokussierte Ansatz neben der textuellen Auswertung auch bei den Bildern funktioniert und sich

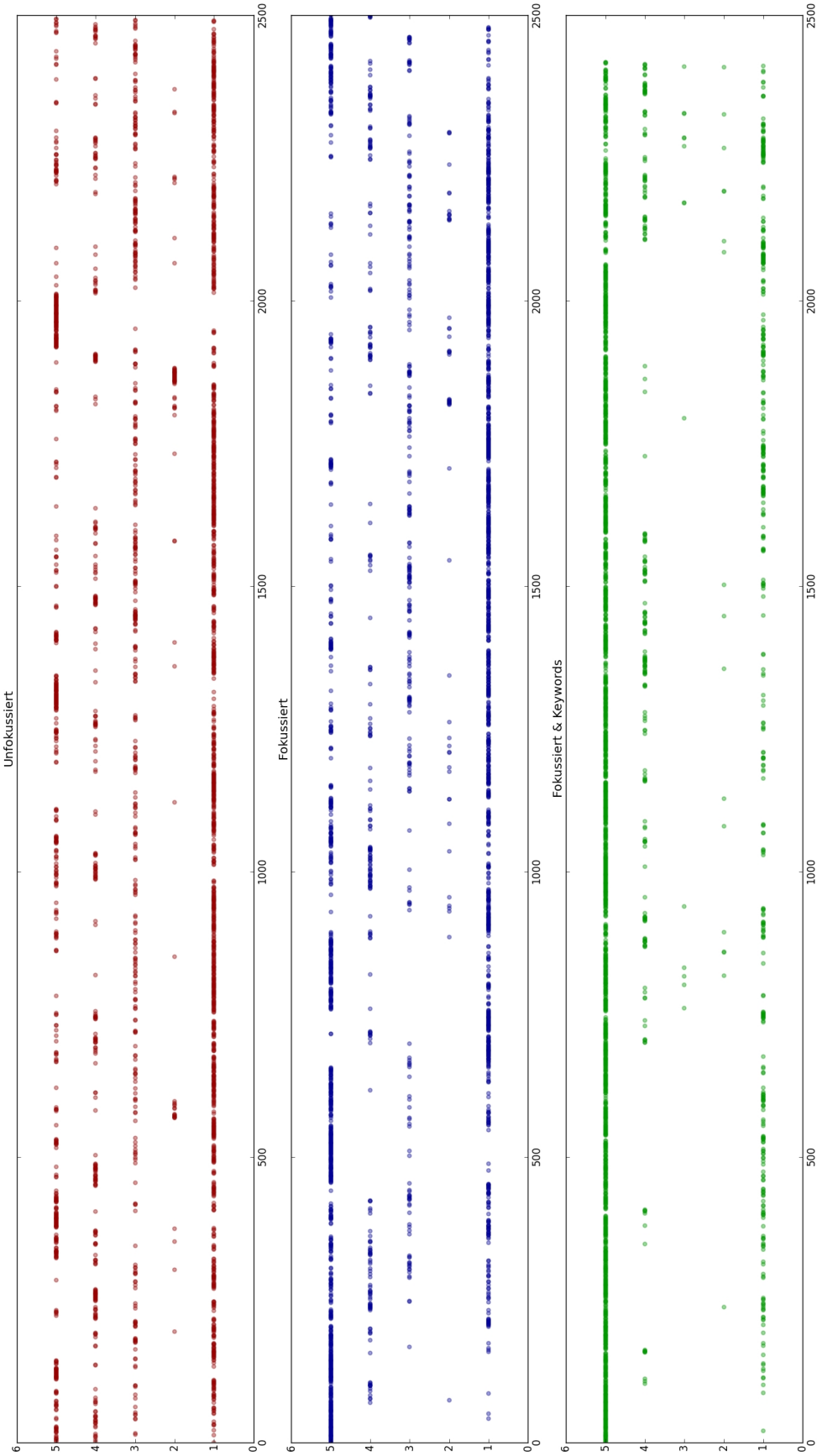


Abbildung 5.5.: Laufvergleiche innerhalb der Kategorien. X-Achse relativ auf 2500 Einträge gemappt.

nicht nur auf die Umgebung der Seed-URLs beschränkt. Gleiches gilt für den fokussierten Crawl mit Keywords. Hier ist gut zu sehen, dass dieser Lauf durchgängig Bilder aus der Klasse *Katze* und *Haustier* bezieht, jedoch weniger *Sonstiges* als die anderen Beiden.

Zwei weitere alternative Darstellung findet sich im Anhang in den Abbildungen A.2 und A.3. Als Achse wurde dazu die größte Spannweite an Bildindizes eines Laufes genutzt (vgl. Tabelle 5.2). Der fokussierte Lauf bestimmt damit in Abbildung A.2 mit seinen mehr als 600.000 Einträgen im Perplexitätslogfile die X-Achse der Indizes. Je größer der Index eines annotierten Bildes, desto später wurde es heruntergeladen. Abbildung A.3 ersetzt noch einmal die Anzahl der annotierten Bilder der X-Achse in Abbildung 5.5 durch die Indexwerte.

Tabelle 5.5.: Mögliche Korpuskandidaten

	Katze & beschreibt das Bild	Katze & passt thematisch zum Bild
Unfokussiert (3)	79	37
Unfokussiert (>1)	121	200
Fokussiert (3)	276	36
Fokussiert (>1)	407	275
Fokussiert mit Keywords (3)	120	553
Fokussiert mit Keywords (>1)	414	1074

Bleibt noch die Betrachtung der eingangs gestellten Frage nach der Realisierbarkeit eines Text- Bildkorpus durch fokussiertes Crawling: Möchte man dieses Ziel komplett automatisiert erreichen, braucht man größer angesetztes Crowdsourcing. Aus gut 39.000 möglichen Kandidaten wurden in dieser Thesis lediglich ca. 2500 annotiert. Das Interannotatoragreement bezüglich der Texte ist jedoch recht gering, sodass konkrete Aussagen mit Vorsicht zu treffen sind. Betrachtet man nun Tabelle 5.5 und sieht die einstimmig annotierten Katzenbilder, deren Text das Bild beschreibt, fällt auf, dass hier sogar der fokussierte Lauf im Vorteil sein könnte. Dagegen spricht allerdings die Menge der mit Mehrheit annotierten Bilder mit 414 Einträgen und 1074 themenrelevanten Texten und legt die Vermutung nahe, dass die Annotation stärker kontrolliert und gebunden werden muss. Bereits das Verhältnis an themenrelevanten Kandidatenbildern in Tabelle 5.2 zeigt, dass hier noch Potential steckt. Betrachtet man den Pool an Bildern auf der Platte, der für die Annotation bereitgelegt wurde, so erkennt man stichprobenartig sehr schnell, dass die Häufigkeit von Katzenbildern mit tiefer liegendem Rang (numerisch höher) besonders schnell im unfokussierten und im normal fokussierten Lauf nachlässt.

Auch wenn derzeit die Ausbeute für beschreibende Texte zu Bildern bei ca 23% der Katzenbilder bei Mehrheitsabstimmung liegt (16% aller annotieren Bilder), schlägt für die Themenrelevanz der Bilder das Pendel zugunsten der durch Keywords verstärkten Methode aus. Die Wirksamkeit dieser Methode demonstiert nachfolgendes Beispiel:
Das in Abbildung 5.6 gezeigte Bildmotiv ist in ähnlicher Form ein zweites Mal im Lauf vorhanden. Das eine Bild wird mit *"Athena and Elsa Clair play with their new electronic toy"* (Abbildung 5.6 (b)[5]) im Alt-Text auf der Webseite eingebunden, das andere mit *"Athena the cat pounces on string from FroliCat FLIK automatic cat teaser"* (Abbildung 5.6 (a)[6]). Die Bilder sind auf der selben Seite in identischer Seitenumgebung untergebracht. Die Infodatei beider Bilder zeigt den selben Parenttag (siehe Listing 5.1). Es ist also das Keywordgewicht, das über die Platzierung entscheidet.

[5] Bildlink: `http://lifewithdogsandcats.com/good-to-know/pet-product-and-book-reviews/technology-pets-chance-win-frolicat-flik` (aufgerufen August 2015) - "Athena and Elsa-Clair"

[6] Bildlink: `http://lifewithdogsandcats.com/good-to-know/pet-product-and-book-reviews/technology-pets-chance-win-frolicat-flik` (aufgerufen August 2015) - "Athena the cat"

(a) "Athena the cat"

(b) "Athena and Elsa-Clair"

Abbildung 5.6.: "Athena" oder "Athena the *cat*" - mehr als 4000 Plätze Rangunterschied

Listing 5.1: Athena the cat

```
Gleiche Parentwerte:
<parent-1-perplexity-pure>2009.2610579687182</parent-1-perplexity-pure>
<parent-1-perplexity-keywordscore>1.1639956458824765</parent-1-perplexity-keywordscore>
<parent-1-perplexity-weighted>1.0</parent-1-perplexity-weighted>

Alt-Tags:
<alt>Athena and Elsa Clair play with their new electronic toy</alt>
<alt-perplexity-pure>15076.108324361443</alt-perplexity-pure>
<alt-keywordscore>0.0</alt-keywordscore>
<alt-perplexity-weighted>15076.108324361443</alt-perplexity-weighted>

<alt>Athena the cat pounces on string from FroliCat FLIK automatic cat teaser</alt>
<alt-perplexity-pure>578.4404971047898</alt-perplexity-pure>
<alt-keywordscore>0.534421481289966</alt-keywordscore>
<alt-perplexity-weighted>1.0</alt-perplexity-weighted>
```

5.1.3 Stolpersteine: Web 2.0 & Boilerpipeextraktor

Während der Testläufe bei der Generierung der Sätze für das Trainingsmodell zeigt sich, dass besonders bei einer größeren Seedliste die Ausbeute an Sätzen erheblich höher ist, wenn der JSoup-Extraktor genutzt wird. Verfolgt man das Crawllog kommt man zum Schluss, dass einige Seiten scheinbar keine Sätze liefern. Der Umstand lässt sich so entschlüsseln, dass vom Boilerpipeextraktor nur wenig, bis gar kein Text geliefert wird. Dieser erweist sich an dieser Stelle als Stolperstein. Bei näherem hinsehen zeigt sich: besonders modernere Seiten ab der sog. Web 2.0 Ära können problematisch sein, sodass der Boilerplateextraktor Inhalte wegschneidet und vom Inhalt der Seite meist nur noch die Überschriften und ein paar Textzeilen übrig bleiben. Vermutlich liegt dies am Trend, immer mehr Themenbilder und sog. responsive Design für mobile Geräte und deren Bedienkonzepte einzubinden. Ein Blick in die heruntergeladenen Bilddateien für die Evaluation zeigt dies ebenfalls - viele Bilder enthalten nicht ausschließlich ein Motiv, sondern sind gleichzeitig Aufmacher eines Artikels. Die Erkennungsrate anhand der Textdichte und der Umgebung scheint darunter zu leiden. Ein Testinterface[7] wird zum aktuellen Zeitpunkt von Kohlschütter bereitgestellt. Auf diese Weise ist es möglich, ohne großen Aufwand zu testen, ob sich eine Seite für die Extraktion durch das Boilerpipemodul in Heritrix eignet. Diese Vorgehensweise ist in die Methode zur Trainingsmodellgenerierung eingeflossen. Jedoch böte sich Potential für ein besseres Crawling, wenn

[7] `http://boilerpipe-web.appspot.com`

die Zuverlässigkeit des Boilerpipeextraktors erhöht werden kann, sodass dieser auch im Lauf eingesetzt werden kann. Ganz allgemein lässt sich sagen, dass sehr javascriptlastige Seiten und Trends wie responsive Design problematisch für die Herangehensweise der Extraktion werden können, wenn der Ort des Inhalts keine Rückschlüsse auf den Anzeigeort und damit auf die Nähe bzw. Relevanz von einem Bild zu einem Text zulässt. Letzteres sind glücklicherweise Ausnahmen, unter dem Trend dorthin leidet aber bereits die Genauigkeit des Boilerpipeextraktors.

6 Ausblick

Nach der Auswertung der Läufe zeigt sich, dass der Ansatz des fokussierten Crawls für die Bildsuche bereits vielversprechend anmutet. Für die Generierung eines echten Korpus ist das System derzeit jedoch noch zu unausgereift. 120 garantierte- und 410 mutmaßlich gültige Kandidaten für einen bildbeschreibenden Text aus rund 2500 Bildern sind derzeit noch eine geringe Quote und für ein Korpus ist das eine zu kleine absolute, Zahl. Doch das System zeigt durch die Bildsuche bereits Ansätze, die dazu einladen, mehr mit dem Thema zu arbeiten. Betrachtet man den Bilderpool aus dem fokussierten Crawl, so ist selbst bei den abgeschnittenen 5000 Bildern noch nicht Schluss mit Katzencontent. Die Werte deuten auf deutlich mehr Katzenbilder hin, als in der Evaluation betrachtet wurden. Der Kategorisierungsverlauf der Indizes der heruntergeladenen Bilder zeigt z.B., dass selbst am Schluss des Crawls bei 100GB Limit weiterhin thematisch passende Bilder gefunden wurden. Eine größere Skalierung des Crawls fördert daher möglicherweise noch viele weitere Bilder mit niedriger Text- und Umgebungsperplexität zu tage und schließt das Ziel, einen Korpus zu erstellen, nicht aus.

Der hier gezeigte Lauf stellt ein ersten Testversuch dar, an dem viele Parameter noch verändert und optimiert werden können. Um diese Stellschrauben besser kennenzulernen, bietet es sich an, das gleiche Experiment mit dem gleichen und anderen Themenbereichen durchzuführen und dabei Erfahrungen zu sammeln, inwieweit man das System für z.B. das Katzenthema optimieren kann und dabei noch wichtiger - ob es auch mit anderen Themen funktioniert. Identifizierte Stellschrauben wie z.B. die Keywordextraktion und Keywordamplifikation können angegangen werden. Als Beispiel wäre hier zu nennen, dass man versuchen könnte aus der Keywordexktraktion zusätzlich eine Liste von vermeintlichen Stoppwörtern zu finden und diese beim Testen gegen das Sprachmodell zu nutzen. Erste Versuche in diese Richtung zeigten gute Ergebnisse, doch die Ausrichtung war zunächst weiterhin, beim Positivlistenansatz zu bleiben. Die anderen bisherigen Parameter können schärfer eingestellt werden - als Beispiel kann die Priorisierung über die Buckets auf niedrigere Perplexitätswerte getuned werden. Es ist zu prüfen, ob aus den in diesem Lauf gewonnenen Daten weitere Metriken für die Themenbewertung gewonnen werden können. Die gefundenen, bildbeschreibenden, Texte von Katzenbildern könnten als neues Trainingsmodell hergenommen werden, um ein neues Trainingskorpus an Sätzen daraus zu ercrawlen. Als erweitertes Featureset könnte z.B. Textdichteanalysen, wie die von Kohlschütter in der Boilerplateerkennung, in an das Thema angepasster Form geprüft werden. Um die Sprachunabhängigkeit zu wahren, ist beim statistischen Ansatz zu bleiben. Eine andere Möglichkeit die Ergebnisse zu verbessern, scheint eine Anpassung an das sogenannte Web 2.0 zu sein. Nicht nur durch den Stolpterstein mit dem Boilerpipeextraktor wurde während es Crawls deutlich, dass die Mehrheit der im Netz befindlichen Seiten nicht mehr HTML im ursprünglichen Sinne nutzt, sondern Communities wie Tumblr, Blogspot, Instagram oder Contentmanagementsysteme wie Wordpress intensiv genutzt werden um Inhalte zu publizieren. Dabei macht mittlerweile Javascript einen großen Teil der Funktionalität und Gestaltung einer Webseite aus. Mitte der 2000er noch als nicht Barrierefrei abgekanzelt, ist Javascript derzeit nicht wegzudenken. Handgeschriebene HTML-Seiten sind dagegen selten geworden. Das schlägt sich in der korrekten Nutzung von HTML-Attributen wie im konkreten Fall ALT-Tags nieder. Einer der häufig aufgetauchten Strings im Alttext sind von Wordpress automatisch scheinbar aus dem Pfad generierte Tags wie ”wp content uploads xyz”. Ein Filter für diese Autogenerierungen ist daher erstrebenswert.

Die methodischen Unzulänglichkeiten beim privaten Crowdsourcing können angegangen werden, indem aus den in diesem Lauf eindeutig annotierten Bildern ein kleiner Goldstandard gebildet wird. Damit wäre es möglich, große Crowdsourcing Plattformen zu nutzen, die ein besseres Innerannotatoragreement fördern. Zum einen stehen jetzt dafür Beispiele bereit, zum anderen ist über diese Plattformen ein wiederkehrender Goldtest und eine hohe Einstiegshürde für eine Verbesserung der Qualität der Annotationen leicht realisierbar. Um den Einfluss des Textes auf die reine Bildbewertung zu reduzieren, kann der Aufbau der Annotation in zwei Phasen unterteilt werden. In der ersten Phase werden ausschließlich

Bilder und deren Thema annotiert. In der zweiten Phase werden den Annotatoren zur Bewertung der Bild-Text-Relation nur noch die Bilder gezeigt, die bereits im ersten Lauf als thematisch relevant eingestuft worden. Auf diese Weise senkt man die kognitiven Anforderungen an die Annotatoren und filtert gleichzeitig untaugliche Bilder im ersten Durchgang aus.

Zunächst ist jedoch geplant, das geschriebene Modul weiter zu stabilisieren und anschließend als OpenSource zu veröffentlichen.

7 Danksagung

An dieser Stelle möchte ich mich bei Personen bedanken, die mich beim Fortkommen der Arbeiten an dieser Thesis unterstüzt haben. Zunächst möchte ich meinem Betreuer Steffen Remus für die Einführung in das Thema Crawler und vor allem für die gute Betreuung und die nützlichen Ratschläge bedanken. Weiterhin möchte ich mich bei meiner Familie für die Unterstützung bedanken, die ich während der Arbeit erfahren durfte. Dazu gehören neben der Annotation von Bildern auch Serverstellplätze, Strom und eine ausgelastete Leitung. Ein weiterer wichtiger Baustein waren meine Annotatoren: an dieser Stelle möchte ich mich namentlich bei allen bedanken, die die Zeit für die Annotation geopfert haben, sodass insgesamt mehr als 22500 mal Bilder angesehen und bewertet wurden:

Friedbert,
Brigitte,
Julia,
Iris,
Vivian,
Patrick,
Simon,
Jan,
Reinhard,
Michael,
Christopher,
Julian,
Doris,
Manfred,
Torsten.

Weiterhin möchte ich mich bei David für das Bekanntmachen mit Sprachtechnologie, Sebastian für den Hardwareverleih und Daniel für den intensiv genutzten Speicher auf seinem Server bedanken.

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Literatur

Bird, Steven (2006). “NLTK: The Natural Language Toolkit”. In: *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*. Sydney, Australia: Association for Computational Linguistics, S. 69–72.

Bird, Steven, Ewan Klein und Edward Loper (2009). *Natural Language Processing with Python*. O’Reilly Media, Inc.

Borin, Lars (2002). *Parallel Corpora, Parallel Worlds: Selected Papers from a Symposium on Parallel and Comparable Corpora at Uppsala University, Sweden, 22-23 April, 1999*. Rodopi.

Brin, Sergey und Lawrence Page (1998). “The Anatomy of a Large-Scale Hypertextual Web Search Engine”. In: *Seventh International World-Wide Web Conference (WWW 1998)*.

Chakrabarti, Soumen, Martin van den Berg und Byron Dom (1999). “Focused crawling: a new approach to topic-specific Web resource discovery”. In: *Computer Networks* 31.11–16, S. 1623–1640.

Chen, Stanley F. und Joshua Goodman (1996). “An Empirical Study of Smoothing Techniques for Language Modeling”. In: *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*. Santa Cruz, California, USA: Association for Computational Linguistics, S. 310–318.

Cho, Junghoo, Hector Garcia-Molina und Lawrence Page (1998). “Efficient Crawling Through URL Ordering”. In: *Seventh International World-Wide Web Conference (WWW 1998)*.

Cohen, Jacob (1968). “Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit.” In: *Psychological bulletin* 70.4, S. 213.

De Bra, Paul und Reinier Post (1994a). “Information Retrieval in the World-Wide Web: Making Client-based Searching Feasible”. In: *Selected Papers of the First Conference on World-Wide Web*. Geneva, Switzerland: Elsevier Science Publishers B. V., S. 183–192.

– (1994b). “Searching for arbitrary information in the www: The fish-search for mosaic”. In: *WWW Conference*.

Deng, Jia et al. (2009). “ImageNet: A Large-Scale Hierarchical Image Database”. In: *CVPR09*.

Dodge, Jesse et al. (2012). “Detecting visual text”. In: *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, S. 762–772.

Fellbaum, Christiane (1998). *WordNet: An Electronic Lexical Database*. Bradford Books.

Herscovici, Michael et al. (1998). “The Shark-search Algorithm. An Application: Tailored Web Site Mapping”. In: *Computer Networks and ISDN Systems* 30.1-7, S. 317–326.

Johnson, Rod et al. (2004). *The spring framework, reference documentation*.

Kneser, Reinhard und Hermann Ney (1995). “Improved backing-off for m-gram language modeling”. In: *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*. Bd. 1. IEEE, S. 181–184.

Koehn, Philipp (2005). “Europarl: A parallel corpus for statistical machine translation”. In: *MT summit*. Bd. 5, S. 79–86.

Kohlschütter, Christian, Peter Fankhauser und Wolfgang Nejdl (2010). “Boilerplate detection using shallow text features”. In: *Proceedings of the third ACM international conference on Web search and data mining*. New York, NY, USA, S. 441–450.

Kuznetsova, Polina et al. (2013). “Generalizing Image Captions for Image-Text Parallel Corpus”. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, S. 790–796.

McCallum, Andrew et al. (1999). “Building domain-specific search engines with machine learning techniques”. In: *Proc. AAAI-99 Spring Symposium on Intelligent Agents in Cyberspace*.

Menczer, Filippo, Gautam Pant und Padmini Srinivasan (2004). “Topical web crawlers: Evaluating adaptive algorithms”. In: *ACM Transactions on Internet Technology (TOIT)* 4.4, S. 378–419.

Mohr, Gordon et al. (2004). “Introduction to heritrix, an archival quality web crawler”. In: *Proceedings of the 4th International Web Archiving Workshop IWAW'04*. Bath, UK.

Ordonez, Vicente, Girish Kulkarni und Tamara Berg (2011). “Im2text: Describing images using 1 million captioned photographs”. In: *Advances in Neural Information Processing Systems*, S. 1143–1151.

Page, Lawrence et al. (1999). *The PageRank Citation Ranking: Bringing Order to the Web.* Techn. Ber.

Pinkerton, Brian (1994). “Finding what people want: Experiences with the WebCrawler”. In: *Proceedings of the Second International World Wide Web Conference*. Bd. 94. Chicago, S. 17–20.

Rashtchian, Cyrus et al. (2010). “Collecting Image Annotations Using Amazon's Mechanical Turk”. In: *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. CSLDAMT '10. Los Angeles, California: Association for Computational Linguistics, S. 139–147.

Remus, Steffen und Chris Biemann (unpublished). *Now Focus! N-Gram Language Models as Simple Means for Focused Web Crawling*. Darmstadt, Germany.

Teubert, Wolfgang (1998). “Korpus und Neologie”. In: *Studien zur deutschen Sprache: Neologie und Korpus* 11, S. 129.

Trosterud, Trond (2002). “Parallel corpora as tools for investigating and developing minority languages”. In: *Language and Computers* 43.1, S. 111–122.

Tschirsich, Martin und Gerold Hintz (2013). “Leveraging Crowdsourcing for Paraphrase Recognition”. In: *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. Sofia, Bulgaria: Association for Computational Linguistics, S. 205–213.

A Anhang

Listing A.1: Keywordgenerierung: Liste thematisch relevante URLs

http://en.wikipedia.org/wiki/Cat
http://en.wikipedia.org/wiki/Cat_behavior
http://en.wikipedia.org/wiki/Cat_body_language
http://en.wikipedia.org/wiki/Kitten
http://en.wikipedia.org/wiki/Maine_Coon
http://animals.nationalgeographic.com/animals/mammals/domestic-cat/
https://www.petfinder.com/cat-breeds/?see-all=1

Listing A.2: Keywordgenerierung: Liste thematisch neutrale URLs

http://en.wikipedia.org/wiki/Car
http://en.wikipedia.org/wiki/John_F._Kennedy
http://www.washingtonpost.com/
http://www.nytimes.com/
http://en.wikipedia.org/wiki/The_Wall_Street_Journal
http://en.wikipedia.org/wiki/Sepp_Blatter
http://en.wikipedia.org/wiki/Michael_Jordan

Listing A.3: Keywordgenerierung: Keywordscore

```
double neutraleIDF = 0;
double differenz = 0;

for (Entry<String, Integer> entry : final_positive_TFMAP.entrySet()){
String token = entry.getKey();

neutraleIDF = 0;
differenz = 0;


/* Fuer die binaere TF IDF Rechnung */
if(neutral_1malIDFMAP.containsKey(token)){
        neutraleIDF = neutral_1malIDFMAP.get(token);
}
else{

        double posdocumentcount = 0.1;
        double preidf = neutralTFListAndMap.size() / posdocumentcount;
        double customidf= Math.log10(1d+ preidf);
        neutraleIDF = customidf;
}

differenz = neutraleIDF - positive_1malIDFMAP.get(token);               // Verstaerker, der die Differenz erhoeht
differenz = differenz * final_positive_TFMAP.get(token);

idfdifferenzmap.put(token, differenz);

// Debug
System.out.println("Binaer: " + neutraleIDF + " - " + positive_1malIDFMAP.get(token) + " | Diff = " + differenz + " | " + token );

}
```

Listing A.4: Bewertung eines Links - Javaquelltext

```
double keywordscore_linktext = getKeywordScore(link.ownText());
double perplexity_weighted = documentPerplexity;

if (perplexity_owntext < 3000d && perplexity_owntext < documentPerplexity)
{
// Obviously the Linktext is great
perplexity_weighted = (((0.7d * documentPerplexity) + (0.3d * perplexity_owntext) ) * (1.0d - (linkkeywordscoreweight * keywordscore_linktext)));
}
else
{
// And if it carries keywords, also great
perplexity_weighted = documentPerplexity * (1.0d - (linkkeywordscoreweight * keywordscore_linktext));
}
/* Anchor for rating not toooooo good ;) */
if (perplexity_weighted < 0.0d){
perplexity_weighted = 1d;
}
```

Listing A.5: Keywordgenerierung: Top90 Katzen

```
cat;1015.3995712510865
cats;834.0228539436406
coon;141.83055278230566
breed;139.51856064451562
feral;75.57255368244596
species;72.66591700235188
maine;70.64408057738689
felis;63.94600696206965
breeds;58.844591047977886
shorthair;58.844591047977886
pet;56.679415261834464
purr;54.25799235692829
prey;46.5061868815052
fanciers;43.7562343690092
mongoose;42.47274415129784
catus;42.146231861364086
veterinary;42.146231861364086
kittens;42.146231861364086
rex;41.85616553248754
feline;40.738563033215456
fur;40.69291352131705
kitten;39.23959518127001
dogs;37.78627684122298
ears;36.21205602952485
pets;35.655252120267164
purring;34.353427349985324
birds;30.51968514098779
humans;29.715367226996076
pmid;29.391842526787396
males;29.06636680094075
hunting;27.90411035499169
claws;27.61304846089371
haired;27.61304846089371
mammals;26.108604785988845
adoption;26.108604785988845
domestic;25.263849024138388
kneading;24.706411780799638
lion;24.706411780799638
paws;24.706411780799638
genetics;23.2530934407526
dog;22.878279670462568
coons;22.13552679150601
urine;21.986193503990606
spotted;21.986193503990606
vocal;21.703196942771317
meat;21.123699350556166
bobtail;21.123699350556166
diet;21.123699350556166
tail;20.711149111116143
leopard;20.612056409991194
otter;19.987173718257807
grooming;19.614863682659294
communicate;19.614863682659294
civet;19.117045865391553
weasel;18.737975360866695
animal;18.665673459902216
felines;18.602740236661127
animals;18.185399219516615
siamese;18.106028014762426
domesticated;17.863782221992366
breeders;17.863782221992366
posture;17.863782221992366
conservation;17.488777003475583
meow;17.488777003475583
hz;17.488777003475583
eyes;17.439820080564452
scent;17.439820080564452
anatomy;17.439820080564452
communication;17.175099262876802
burmese;17.052511883606034
bengal;17.052511883606034
norwegian;16.597192346865558
tend;16.597192346865558
behavior;16.471861146516286
felidae;16.239578646084468
females;16.239578646084468
eating;16.239578646084468
reflex;15.986501740517413
skin;15.986501740517413
cardiomyopathy;15.986501740517413
hypertrophic;15.986501740517413
snowshoe;15.50228353055094
oriental;15.50228353055094
behaviors;15.115508033993542
glands;15.115508033993542
individuals;15.115508033993542
gentle;15.115508033993542
silvestris;15.092404630572279
genet;15.092404630572279
forest;15.088356678968688
```

Listing A.6: Keywords: Top46 - Vulkan

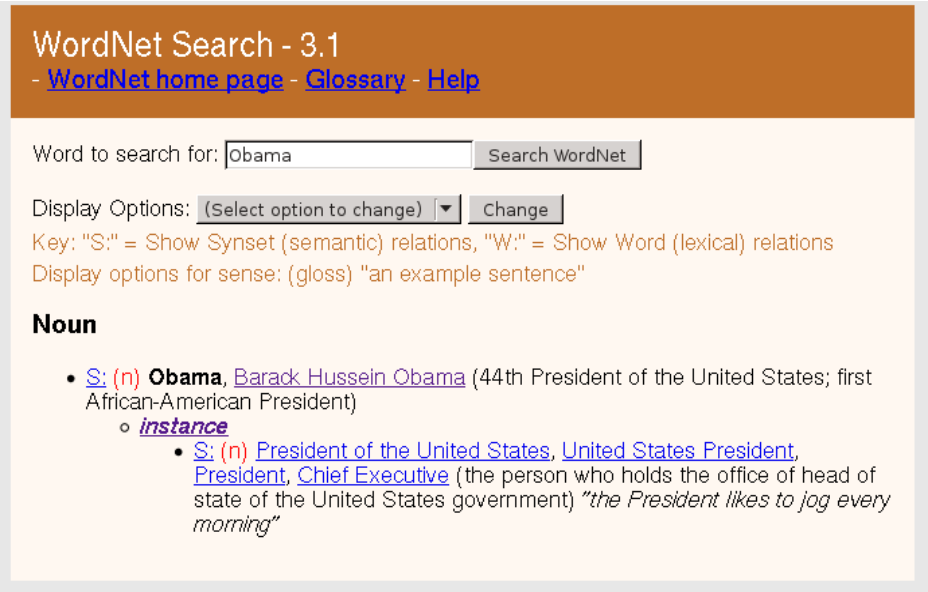
```
volcanoes;189.12785907272146
volcanic;142.62100848106866
volcano;131.76941000968299
lava;91.46347283025055
eruption;58.908677416093575
eruptions;51.157535650818105
ash;40.30593717943244
erupted;34.10502376721207
extinct;29.454338708046787
flows;26.353882001936597
cones;26.353882001936597
gases;24.803653648881504
plates;20.15296858971622
magma;20.15296858971622
geological;18.602740236661127
earth;17.459739748510906
cone;17.052511883606034
cinder;17.052511883606034
oceanic;17.052511883606034
lavas;15.50228353055094
ocean;15.50228353055094
dormant;15.50228353055094
vents;15.50228353055094
mud;15.50228353055094
submarine;13.952055177495845
sulfur;13.952055177495845
tectonic;13.952055177495845
stratovolcanoes;13.952055177495845
surface;13.24531980921517
io;12.401826824440752
domes;12.401826824440752
vent;12.401826824440752
silica;12.401826824440752
eruptive;12.401826824440752
crater;12.401826824440752
erupt;12.401826824440752
pyroclastic;12.401826824440752
volcanism;12.401826824440752
plate;11.439139835231284
volcanology;10.851598471385659
fissure;10.851598471385659
crust;10.851598471385659
caldera;10.851598471385659
mons;9.301370118330563
etna;9.301370118330563
stratosphere;9.301370118330563
```

Listing A.7: Keywords: Top40 Todesstern

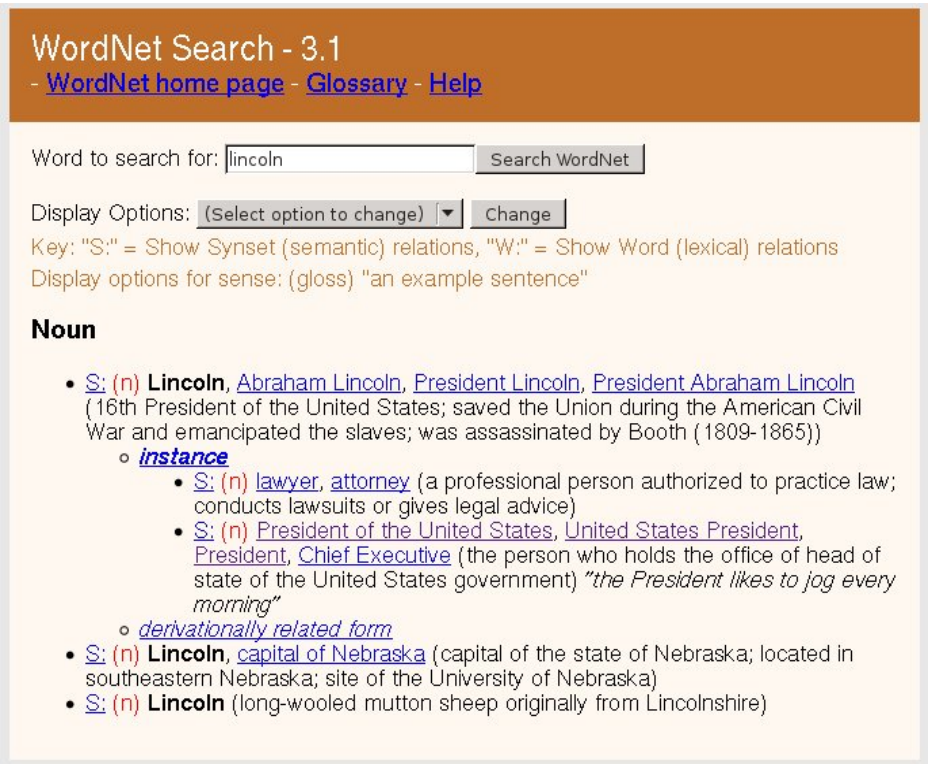
```
wars;33.715359514365886
jedi;17.052511883606034
vader;13.952055177495845
lucasfilm;12.401826824440752
star;11.243736938762726
darth;10.851598471385659
leia;10.851598471385659
superlaser;9.301370118330563
tarkin;9.301370118330563
universe;7.75114176527547
rebels;7.75114176527547
lego;7.75114176527547
death;7.04713089823861
sith;6.200913412220376
diameter;6.200913412220376
legends;6.200913412220376
obi;6.200913412220376
episodes;6.200913412220376
fiction;6.200913412220376
rebel;6.200913412220376
wan;6.200913412220376
databank;6.200913412220376
kilometers;6.200913412220376
lucasarts;6.200913412220376
novel;6.200913412220376
skywalker;4.650685059165282
landspeeder;4.650685059165282
vi;4.650685059165282
saturn;4.650685059165282
wikiproject;4.650685059165282
mimas;4.650685059165282
canon;4.650685059165282
revenge;4.650685059165282
rogue;4.650685059165282
schematics;4.650685059165282
stears;4.650685059165282
planets;4.650685059165282
alderaan;4.650685059165282
falcon;4.650685059165282
millennium;4.650685059165282
```

Listing A.8: Auszug der CrawlerBeans Konfiguration

```
...
extractorHtmlwithPictures.serviceID=catlm_training_70prozent_5gram_kneserney
extractorHtmlwithPictures.rmihost=192.168.0.200
extractorHtmlwithPictures.rmiport=1099
extractorHtmlwithPictures.assignmentBoundaries=6e2,2e3,1e4

extractorHtmlwithPictures.linkRazor=3e4
extractorHtmlwithPictures.obeyLinkLimits=true

extractorHtmlwithPictures.keywordFilePath=/home/dennis/workspace/lt.ltbot/resources/preparators/generated_keywords_50.txt
extractorHtmlwithPictures.minImgWidth=512
extractorHtmlwithPictures.minImgHeight=348
extractorHtmlwithPictures.maxImgWidth=9600
extractorHtmlwithPictures.maxImgHeight=7200
extractorHtmlwithPictures.obeyImageLimits=true
extractorHtmlwithPictures.downloadPicturesSynchronized=true
extractorHtmlwithPictures.kickUnspecifiedImages=true

extractorHtmlwithPictures.imgDistanceWeight=500
extractorHtmlwithPictures.useAmplifier=true

extractorHtmlwithPictures.examineLinksWithJsoup=true

extractorHtmlwithPictures.examineImagesWithJsoup=true
extractorHtmlwithPictures.ee=false

extractorHtmlwithPictures.processGeneralTagsByJsoup=true
...
<bean id="perplexityPrioritizer" class="de.tudarmstadt.lt.ltbot.postprocessor.DecesiveValuePrioritizer" />
<bean id="perplexityLoggerDispositionChain" class="de.tudarmstadt.lt.ltbot.postprocessor.DecesiveValueLogger" />
...
<bean id="extractorHtmlwithPictures" class="de.tudarmstadt.lt.ltbot.extractors.ExtractorHTMLwithPicturesPerp">
<property name="extractJavascript" value="false"/>
<property name="extractValueAttributes" value="false" />
<property name="ignoreFormActionUrls" value="true" />
<property name="sentenceMaker">
<ref bean="sentenceMaker" />
</property>
<property name="textExtractor">
<bean class="de.tudarmstadt.lt.ltbot.text.TextExtractor">
<property name="utf8Cleaner">
<bean class="de.tudarmstadt.lt.ltbot.text.UTF8CleanerExt" />
</property>
<property name="htmlTextExtractor">
<bean class="de.tudarmstadt.lt.ltbot.text.JSoupTextExtractor" />
<!-- bean class="de.tudarmstadt.lt.ltbot.text.BoilerpipeTextExtractor" / -->
</property>
</bean>
</property>
<property name="textExtractorTwo">
<bean class="de.tudarmstadt.lt.ltbot.text.TextExtractor">
<property name="utf8Cleaner">
<bean class="de.tudarmstadt.lt.ltbot.text.UTF8CleanerExt" />
</property>
<property name="htmlTextExtractor">
<!-- bean class="de.tudarmstadt.lt.ltbot.text.JSoupTextExtractor" / -->
<bean class="de.tudarmstadt.lt.ltbot.text.BoilerpipeTextExtractor"/>
</property>
</bean>
</property>
</bean>
...
```



(a) Wordnet - Barrack Obama - Instanz von President. Abgerufen am 20.8.2015



(b) Wordnet - Abraham Lincoln - Instanz von President. Abgerufen am 20.8.2015

Abbildung A.1.: Wordnet und Instanzen von Präsidenten

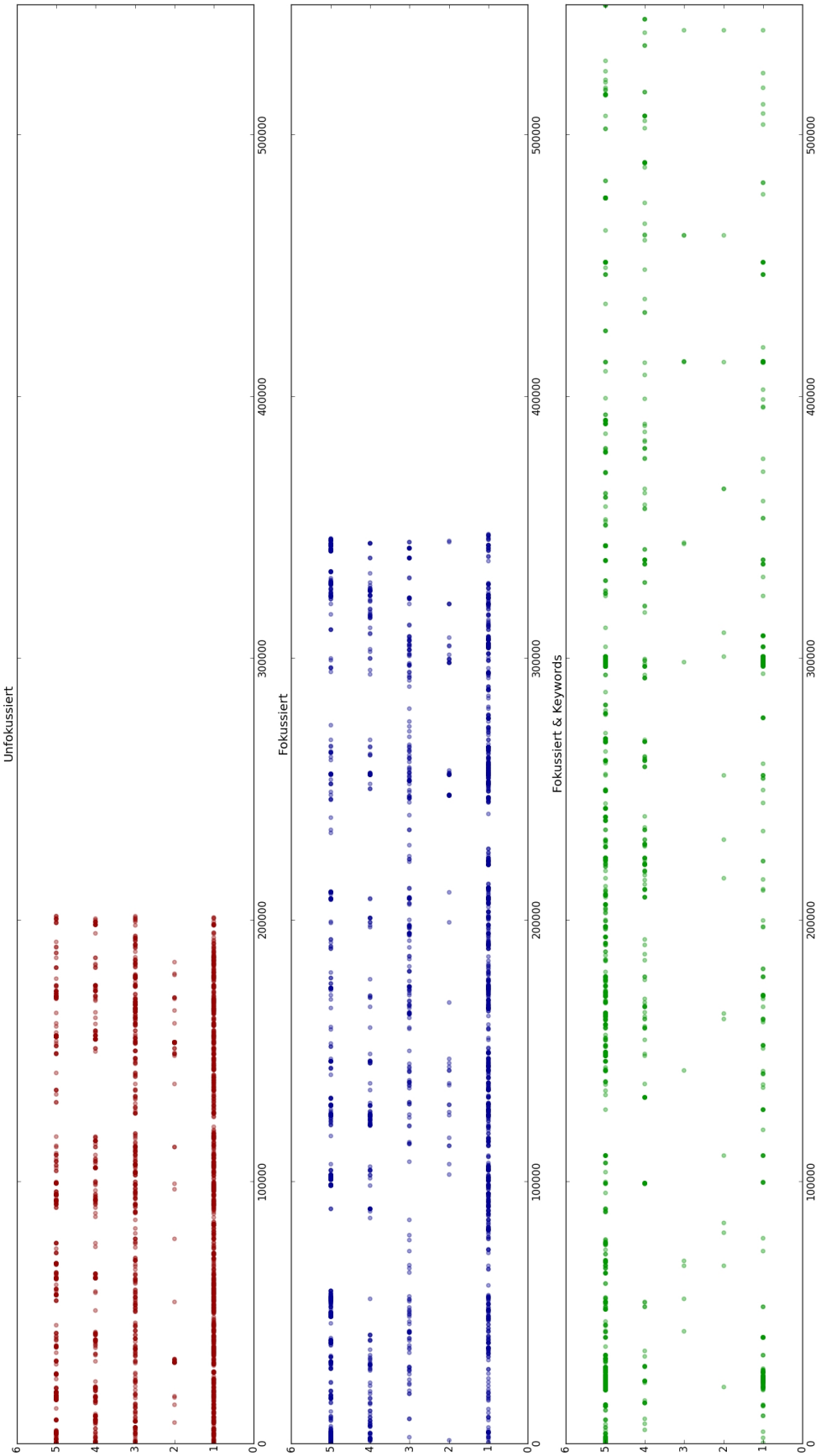


Abbildung A.2.: Laufvergleiche innerhalb der Kategorien, X-Achse auf Indizes gemappt absolut

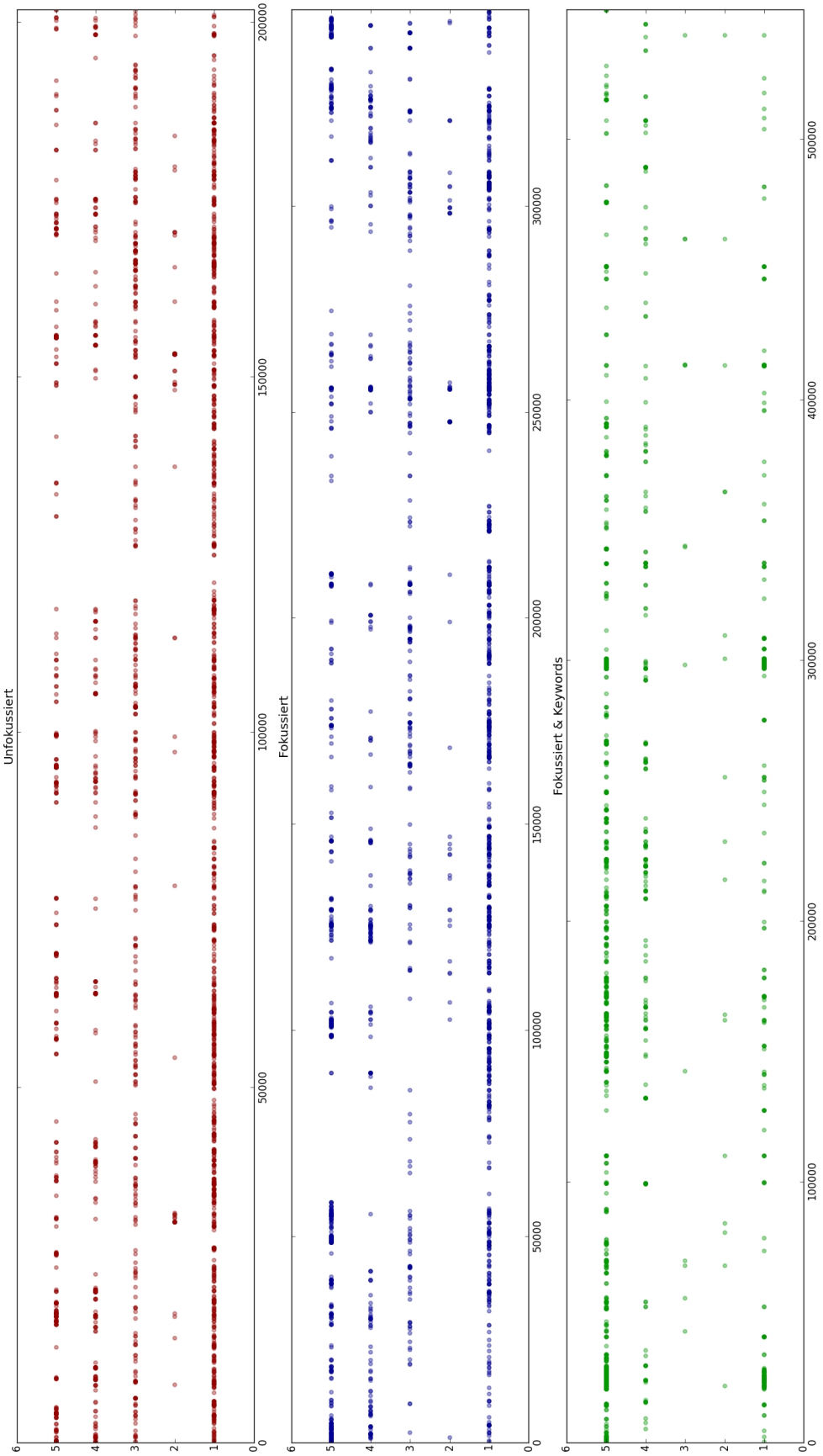


Abbildung A.3.: Laufvergleiche innerhalb der Kategorien, X-Achse auf Indizes gemappt relativ

Tabelle A.1.: Statistik der annotierten Bilder

	Unfokussiert	Fokussiert	Fokussiert mit Keywords
Katze (3)	464	775	1536
Katze (2)	37	66	221
Katze gesamt	**501**	**841**	**1757**
Haustier (3)	273	203	181
Haustier (2)	37	39	39
Haustier gesamt	**310**	**242**	**220**
Säugetier (3)	315	42	2
Säugetier (2)	111	242	12
Säugetier gesamt	**426**	**284**	**14**
Lebewesen (3)	49	19	3
Lebewesen (2)	20	36	14
Lebewesen gesamt	**79**	**55**	**17**
Sonstiges (3)	957	857	234
Sonstiges (2)	232	250	177
Sonstiges gesamt	**1189**	**1107**	**411**
Uneinig	19	63	90
Summe	**2514**	**2529**	**2419**

Tabelle A.2.: Statistik der annotierten Texte

	Unfokussiert	Fokussiert	Fokussiert mit Keywords
Beschreibt das Biild (3)	155	375	141
Beschreibt das Bild (2)	303	334	364
Beschreibt gesamt	**458**	**709**	**505**
Passt thematisch (3)	302	386	766
Passt thematisch (2)	675	669	651
Passt gesamt	**977**	**1055**	**1417**
Irrelevant (3)	316	112	33
Irrelevant (2)	629	523	355
Irrelevant gesamt	**945**	**635**	**388**
Kontraproduktiv (3)	4	8	0
Kontraproduktiv (2)	130	185	199
Kontaproduktiv gesamt	**134**	**193**	**199**
Uneinig	0	0	0
Summe	**2514**	**2592**	**2509**