

**Bachelorarbeit**

# **Maschinelles Lernen im Onlinehandel: Extraktion Produktspezifischer Daten**

**Content Extraction from Web Pages Using Machine Learning**

Leonardo Hübscher

B.Sc.

IT-Systems Engineering  
Fachgebiet für Informationssysteme

Betreuer:

Prof. Felix Naumann

Leon Bornemann

Stanislav Nowogrudski

20. Juli 2018

## Zusammenfassung

Durch die Vielzahl von Onlineshops und Fülle an Angeboten verliert der Onlinekäufer schnell die Übersicht. Preisvergleichsplattformen wie idealo helfen dem Kunden das günstigste Angebot im Netz zu finden. Die Gewährleistung der vollständigen Markttransparenz ist eine grundlegende Herausforderung für idealo. Das von uns entwickelte Softwaresystem *Scout* soll dabei helfen, den Produktkatalog von idealo auf Vollständigkeit zu überprüfen und fehlende Angebote aufzulisten. Ein sehr wichtiger Prozessschritt ist dabei die Extrahierung von Produktinformationen, wie der Produktname oder der Preis, aus den einzelnen Webseiten. Die Schwierigkeit der Extraktion liegt darin, dass jeder Shop einen individuellen Aufbau besitzt und unterschiedlich strukturiert ist.

Das entwickelte Parser-Modul löst dieses Problem, indem es für jeden Shop eigene Regeln für die Extraktion der Produktinformationen verwendet. Dabei ist es nicht erforderlich, dass diese Regeln manuell erstellt werden müssen. Durch die Nutzung der bereits vorhandenen Angebote aus dem Bestand von idealo kann die Extrahierung der Struktur mittels Maschinellen Lernens erfasst werden. Durch die Erhöhung der Flexibilität der Datenextraktion und durch die Einführung eines Bewertungssystems konnte der Extraktionsalgorithmus weiter verbessert werden. Messungen, welche auf 50 verschiedenen Shops basieren, haben ergeben, dass die Produktinformationen mit einer Genauigkeit von über 95 Prozent bei einer Trefferquote von etwa 50% extrahiert werden können.

# Inhaltsverzeichnis

<b>1 Die Welt der Preisvergleichsportale</b>	<b>4</b>
1.1 Der Onlinehandel von heute . . . . .	4
1.2 Das Preisvergleichsportal idealo . . . . .	4
1.3 Das Ziel des Bachelorprojektes . . . . .	4
1.4 Die Microservice-Architektur des Scout-Softwaresystems . . . . .	5
<b>2 Extraktion Produktspezifischer Daten</b>	<b>6</b>
2.1 Die technischen Anforderungen an den Parser . . . . .	6
2.2 Die Positionsbestimmung der Produktattribute . . . . .	6
2.3 Die Annahmen für die Umsetzung der shop-spezifischen Methode . . . . .	7
2.4 Die zwei Schritte der Datenextraktion . . . . .	7
2.5 Die Funktionsweise der Parser-Komponente . . . . .	8
2.6 Die Funktionsweise des Shop Rules Generators . . . . .	8
2.7 Der URL-Cleaner . . . . .	9
2.8 Die Erstellung von Selektoren . . . . .	10
2.9 Die Trimm-Funktion . . . . .	11
2.10 Das Bewertungssystem für die Selektoren . . . . .	11
<b>3 Die Genauigkeitsmessung des Extraktionsalgorithmus</b>	<b>12</b>
3.1 Die Testdaten der Evaluierung . . . . .	12
3.2 Die Messergebnisse . . . . .	12
3.3 Mögliche Fehlerquellen der Messungen . . . . .	13
<b>4 Der Abschluss</b>	<b>14</b>
<b>Das Literaturverzeichnis</b>	<b>15</b>
<b>Die Selbstständigkeitserklärung</b>	<b>16</b>

# 1 Die Welt der Preisvergleichsportale

Der Fernhandel ist bereits seit der Steinzeit ein wichtiger Bestandteil der Gesellschaft. Durch die rasante Entwicklung des Internets und der steigenden Anzahl der Onlinehändler vergrößert sich das Produktangebot.

Heutzutage kann ein Käufer aus einer Vielzahl von Artikeln wählen und muss sich nicht wie in der Steinzeit auf einen Händler oder auf die lokale Verfügbarkeit beschränken.

## 1.1 Der Onlinehandel von heute

In den letzten Jahren hat der Onlinehandel sowohl an Bedeutung für die Unternehmen, als auch für die Kunden gewonnen. Laut einer Statistik von Eurostat machte im Jahr 2017 der Onlinehandel 21% des Gesamtumsatzes deutscher Unternehmen aus und stellte somit einen nicht unerheblichen Anteil dar [4]. Aus einer weiteren Statistik geht hervor, dass 2017 zwei Drittel der Deutschen regelmäßig online einkauften [3].

Die Entwicklung bringt jedoch ein Problem mit sich: Mit der steigenden Auswahl an Produkten verliert ein potenzieller Käufer schnell die Übersicht. Preisvergleichsportale versuchen deshalb die Markttransparenz wiederherzustellen. Ein Käufer soll sich sicher sein das für ihn beste Angebot zu finden.

Das Angebot der Vergleichsportale wird von den Internetnutzern gut aufgenommen, wie eine Statistik aus 2017 zeigt: Für einen besseren Vergleich nutzen rund zwei Drittel Möglichkeit sich im Internet zum Produkt oder zum Preis zu informieren [1, 2].

In einem vom Nachrichtensender n-tv beauftragten Test<sup>1</sup>, hat das Deutsche Institut für Service-Qualität mehrere Preissuchmaschinen unter dem Aspekt des günstigsten Preises, der Preisaktualität und dem Nutzererlebnis verglichen.

Im Ergebnis hat idealo.de in allen Kategorien den ersten Platz eingenommen, gefolgt von billiger.de und preis.de. Es wurde jedoch bemängelt, dass selbst beim besten Preisvergleich nur für die Hälfte der Produkte der günstigste Shop angezeigt wurde.

## 1.2 Das Preisvergleichsportale idealo

Die Mission des Preisvergleichsportals idealo ist es, den Vergleich für den Kunden stetig zu verbessern. Je mehr Angebote idealo vergleicht, desto sicherer kann sich der Kunde sein, das tatsächlich günstigste Angebot zu finden.


Dazu schließt idealo Verträge mit mehreren Onlinehändlern ab. Diese verpflichten sich, Daten zu ihren Angeboten an idealo zu übermitteln und zu aktualisieren. Für jeden vermittelten Kauf zahlen die Shops an idealo eine Provision. Diese basiert auf CPC (Kosten pro Klick) oder CPO (Kosten pro tatsächlicher Bestellung).

Um auch zukünftig wettbewerbsfähig zu bleiben arbeitet idealo daran, auch für die letzten 50% der Produkte immer das beste Angebot liefern zu können. Dies erreicht es zum einen durch Vertragsabschlüsse mit weiteren Onlineshops und zum anderen durch die Herstellung, dass tatsächlich alle Angebote eines Vertragspartners gelistet werden.


## 1.3 Das Ziel des Bachelorprojektes

Es soll eine Software entworfen werden, welche eine automatisierte Bestandsanalyse für einen bestimmten Vertragspartner durchführt. Mit Hilfe des resultierenden Berichtes soll es möglich sein, herauszufinden welche Angebote des Vertragspartner im Produktkatalog von idealo fehlen.

<sup>1</sup><https://disq.de/2014/20141004-Preissuchmaschinen.html>

Der Bericht  Informationen darüber enthalten, welche Produkte nicht vorhanden sind, aus welcher Kategorie diese stammen und zu welcher Preisregion die Produkte gehören. Durch diese Übersicht soll ein Mitarbeiter von idealo dazu befähigt werden, die Ursachen für das Fehlen der Angebote herauszufinden.

idealo vermutet, dass ein unbeabsichtigtes Fehlen von Produkten durch einen fehlerhaften Importvorgang zu erklären ist. Zudem könnte es sein, dass ein Händler bewusst nicht alle Produkte bei idealo führen möchte.

Für die Entwicklung dieser Lösung  hatten wir als fünfköpfiges Team neun Monate Zeit. Zudem wurde uns ein Betreuer von idealo zur Verfügung gestellt, welcher die funktionalen Anforderungen an die Software kommunizierte und als technischer Berater diente. Er begleitete uns während des gesamten Entwicklungsprozesses und unterstützte uns bei Fragen bezüglich der Systemarchitektur.

## 1.4 Die Microservice-Architektur des Scout-Softwaresystems

Wir haben uns dafür entschieden, das Gesamtsystem als Microservice-Architektur zu konzipieren.

Die Microservice-Architektur ermöglicht es logisch gekapselte Komponenten zu entwickeln, welche sich sehr gut skalieren und erweitern lassen. Eine ausführlichere Begründung für diese Architekturentscheidung kann in der Bachelorarbeit von Dmitrii Zhmanakov nachgeschlagen werden.[7]


Für die Implementierung der Architektur haben wir die Programmiersprache Java gewählt und verwenden diese in Kombination mit dem Spring-Framework<sup>2</sup>.

Das entwickelte Gesamtsystem *Scout* besteht grob gesehen aus drei Komponenten: dem Crawler, dem Parser und dem Matcher.

Der *Crawler* ist für das Herunterladen jeder einzelnen Seite eines Shops verantwortlich. Jonas Pohlmann hat sich im Projektverlauf intensiv mit verschiedenen Crawling-Frameworks auseinandergesetzt und diese in seiner Bachelorarbeit verglichen.[5]

Die Funktionsweise der maschinenlernbasierten *Matcher*-Komponente wird in der Bachelorarbeit von Tom Schwarzburg näher beschrieben.[6] Der Matcher vergleicht die vom System gefundenen Angebote mit den Angeboten, die idealo bereits kennt.


Damit dieser die geladenen Angebote mit dem Katalog von idealo vergleichen kann, muss das heruntergeladene HTML-Dokument in ein Format gebracht werden, welches der Computer für den Vergleich nutzen kann.

Diesen Schritt erledigt der Parser, welcher zwischen Crawler und Matcher agiert. Der Fokus dieser Arbeit  in der Beschreibung der Funktionsweise und des Aufbaus des Parsers.


---

<sup>2</sup><https://spring.io/>


## 2 Extraktion Produktspezifischer Daten


Der Parser ist dafür verantwortlich die für den Vergleich relevanten Produktinformationen aus HTML-Dateien zu extrahieren und zu normalisieren. Dies ist ein sehr wichtiger Schritt, da die Qualität der extrahierten Werte die Ergebnisse der Matcher-Komponente stark beeinflussen könnte. 

### 2.1 Die technischen Anforderungen an den Parser


Die Herausforderung der Parser-Komponente besteht hauptsächlich darin, das heterogene Informationsschemata der verschiedenen Shops in ein homogenes, normalisiertes Schema zu bringen. 

Im Detail geht es darum, zu jedem Angebot den Titel, die Produktbeschreibung, den Preis, die Marke, die Kategorie, die Produktbilder sowie weitere eindeutige Merkmale im Format von idealo zu erfassen. Diese eindeutigen Merkmale sind zum Beispiel die standardisierte EAN (Europäische Artikelnummer), HAN (Händler Artikelnummer) und SKU (Stock keeping unit – eine shop-spezifische Kennung).

Da die Crawler-Komponente sehr viel Zeit benötigt um alle Seiten zu erfassen, spielt der Zeitfaktor für den Parser keine groSse Rolle. Eine schnelle Verarbeitung der Seiten ist dennoch wünschenswert. 

Es gilt sowohl eine hohe Trefferzahl als auch eine hohe Genauigkeit zu erzielen, damit die Ergebnisse des Parsers als zuverlässig eingestuft werden. Je genauer die Ergebnisse des Parsers im Format von idealo vorliegen,  desto einfacher sollte der Vergleich durch die Matcher-Komponente werden.

### 2.2 Die Positionsbestimmung der Produktattribute

Die eigentliche Schwierigkeit der Datenextraktion liegt in der Bestimmung der Stellen, an denen die gewünschten Informationen stehen.  gibt grundsätzlich zwei Möglichkeiten, wie man die Informationen aus den Angeboten extrahieren kann. Wir haben zwischen dem shop-unspezifischen und den shop-spezifischen Ansatz unterschieden.

Die Initiative Schema.org hat bereits 2011<sup>3</sup> eine erste Lösung für den *shop-unspezifischen* Ansatz entwickelt. Schema.org hat einen Standard entwickelt, den Webseitenbetreiber nutzen können, um bestimmte Daten zu markieren. Shopbetreiber können zum Beispiel die Produktrezensionen, den Preis oder auch den Produktnamen hervorheben. GroSse Suchmaschinenanbieter wie Google, Microsoft oder Yandex können dadurch sehr einfach relevante Informationen direkt in den Suchergebnissen anzeigen. Die Angebote der On-linehändler werden somit besser dargestellt.

Laut einer Schätzung von idealo verwenden rund 40% der Shops diesen Standard. Diese Herangehensweise bezeichnen wir als shop-unspezifischen Ansatz, da man generische Regeln verwenden kann, um die standardisierten Informationen zu erfassen. Eine Lösung zum Auslesen dieser Informationen ist recht einfach und schnell umsetzbar.


Alternativ zum shop-unspezifischen Ansatz gibt es die *shop-spezifische* Herangehensweise, d.h. dass für jeden Onlineshop individuell angepasste Spezifikationen für die Extraktion erstellt werden. Die Regeln des shop-spezifischen Ansatzes bilden eine Übermenge des shop-unspezifischen Ansatzes.

Die Umsetzung dieser Variante ist anspruchsvoller, da diese Spezifikationen zunächst erstellt werden müssten. Wir nehmen jedoch an, dass durch diesen Ansatz sowohl die Treffer-

---

<sup>3</sup><https://schema.org/docs/about.html>

quote als auch die Präzision des Parsers im Vergleich zu dem shop-unspezifischen Ansatz erhöht werden.

Zu Beginn haben wir erste Versuche basierend auf dem Schema.org-Standard unternommen. Leider  haben wir schnell feststellen müssen, dass die Spezifikation oft nicht richtig eingehalten wurden. Dies hat die Qualität der extrahierten Daten stark verringert. Auch bei anderen Standards, welche bei der Strukturierung von Produktdaten im Internet helfen sollen, wie zum Beispiel JSON-LD<sup>4</sup>(W3C) und das Open-Graph-Protokoll<sup>5</sup>(Facebook), konnten wir ähnliche Beobachtungen machen.

Wir haben uns deshalb gegen den shop-unspezifischen Ansatz entschieden.

## 2.3 Die Annahmen für die Umsetzung der shop-spezifischen Methode

Für die Umsetzung der Parser-Komponente haben wir zwei Annahmen getroffen, welche die Konzeption des Algorithmus beeinflusst haben.

Wir nehmen an, dass jeder Shop aufgrund der Vielzahl von Angeboten ein Content Management System (CMS) zur Verwaltung seiner Angebote verwendet. Daraus resultierend gehen wir davon aus, dass sich durch die Verwendung eines CMS die Struktur der Angebote eines Shops ähnelt und diese erlernt werden kann.

Außerdem erwarten wir, dass idealo aufgrund der Vertragsvereinbarungen für die zu untersuchenden Shops bereits eine gewisse Menge an Angeboten besitzt und die Produktattribute nicht manipuliert wurden.


## 2.4 Die zwei Schritte der Datenextraktion

Für die nachfolgenden Erklärungen werden die beiden Begriffe Regel und Selektor definiert. Eine *Regel* ist eine Sammlung von Selektoren für eine bestimmte Produkteigenschaft. Jeder *Selektor* dieser Regel stellt eine Wegbeschreibung durch das HTML-Dokument dar. Er führt zu dem gewünschten Element, aus dem das Produktattribut extrahiert werden soll.

Der grobe Ablauf der shop-spezifischen Datenextraktion kann in zwei Phasen untergliedert werden:

1. Die Generierung der shop-spezifischen Extraktionsregeln/ Spezifikation
2. Die Anwendung der Regeln auf die vom Crawler erzeugten Seiten

In der ersten Phase sollen die Regeln, welche für die Extraktion benötigt werden, mit Hilfe der Daten von idealo angelernt werden.

Die  generierten Regeln werden in der zweiten Phase angewendet, sodass zu jeder Produkteigenschaft genau ein Wert zugeordnet wird. Für jede gecrawlte Seite werden die extrahierten Produktattribute abgespeichert.

Die Logik der beiden Phasen spiegelt sich in der Architektur des Parsers wieder. Dieser besteht aus dem Shop Rules Generator (SRG), der Parser-Komponente und dem URL-Cleaner. Auf die Notwendigkeit des URL-Cleaners und dessen Funktionsweise wird in Kapitel 2.7 eingegangen. Die resultierende Architektur ist in Abbildung 1 abgebildet.

---

<sup>4</sup><https://json-ld.org/>

<sup>5</sup><http://www.ogp.me/>

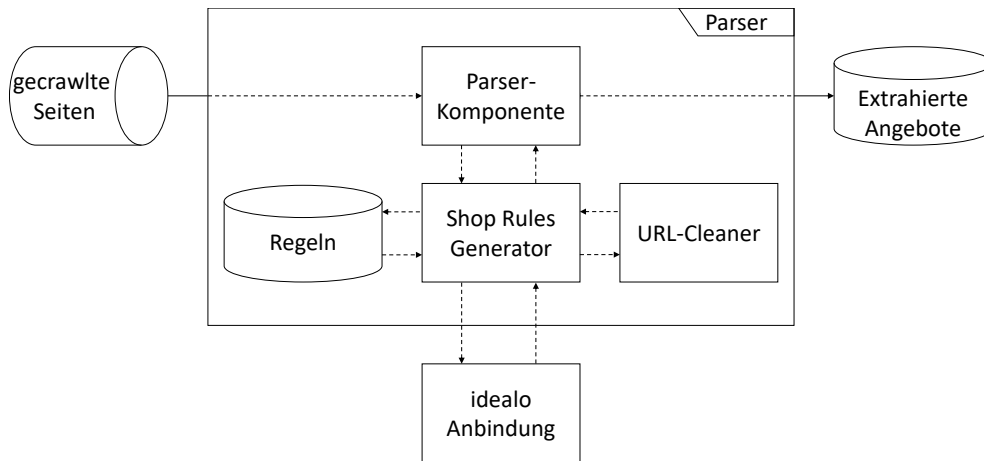


Abbildung 1: Architektur des Parsers

## 2.5 Die Funktionsweise der Parser-Komponente

Die Parser-Komponente erhält ihre Eingaben, indem sie Nachrichten aus einer Queue konsumiert. Eine Nachricht enthält eine vom Crawler heruntergeladene Webseite. Der Crawler sendet zusätzlich zu jeder Seite, die Webadresse und die Identifikationsnummer des zugehörigen Shops mit. Nach dem Erhalt einer Nachricht, lädt der Parser vom Shop Rules Generator (SRG) die Extraktionsregeln für den entsprechenden Shop.

Sollten die Regeln noch nicht existieren, wartet der Parser solange, bis diese vom SRG erstellt wurden. Sobald der Parser die Regeln empfangen hat, werden die Produktattribute aus der gecrawlten Seite extrahiert. Zum Schluss werden diese in einer Datenbank normalisiert abgespeichert.

Der Matcher greift später auf diese Datenbank für den Vergleich zu.

## 2.6 Die Funktionsweise des Shop Rules Generators

Der Shop Rules Generator (SRG) gibt auf Anfrage die Regeln für einen beliebigen Shop zurück. Sollten diese noch nicht existieren, werden sie generiert. Während des Generierungsprozesses durchläuft der SRG mehrere Schritte.

Zuerst werden eine bestimmte Anzahl von Angeboten aus der idealo-Datenbank geladen. Dies erfolgt über einen von idealo zur Verfügung gestellten API-Endpoint, welcher den Datenbankzugriff über eine REST-Schnittstelle kapselt. Dies hat den Vorteil, dass wir die Infrastruktur von idealo nutzen können und keine Kopie der Angebotsdaten lokal speichern müssen.

Zu jedem dieser Angebote liegen die Webadresse, sowie die Informationen über die in Kapitel 2.1 genannten Produktattribute vor. AuSSerdem wird für jedes Angebot das dazugehörige HTML-Dokument heruntergeladen.

Um die Server der Onlineshops nicht durch zu viele simultane Anfragen zu strapazieren, wird nach jedem Download eine fest definierte Zeit gewartet.


Für die Menge der heruntergeladenen HTML-Dokumente ist somit bekannt, um welche Angebote es sich handelt und welche konkreten Produktattribute erwartet werden.


Dieses Wissen wird für das Anlernen der Regeln genutzt. Dazu werden die Werte aller Produkteigenschaften in dem HTML-Dokument gesucht. Für jedes Vorkommen eines Produktattributs wird ein Selektor erstellt, der den Fundort referenziert und einer Regel zugeordnet.

Nachdem alle Regeln gesammelt wurden, wird eine finale Regelmenge bestimmt, welche in der Regeldatenbank gespeichert und bei zukünftigen Anfragen direkt zurückgegeben wird.



## 2.7 Der URL-Cleaner

Manche  onlinehändler manipulieren vor der Übermittlung ihrer Angebote an idealo die Links zu deren Angeboten. Sie fügen zu den regulären Webadressen Trackinginformationen hinzu. Mit Hilfe dieser Trackinginformationen können sie nachvollziehen, welche Kunden durch idealo auf deren Seite gelandet sind.

Dies ist für die Shopbesitzer sehr wichtig, da sie somit die CPC-Abrechnung von idealo kontrollieren können. Die im Rahmen der Anlernphase getätigten Webseitenaufrufe könnten diese Statistiken  noch verfälschen. Für idealo ist es deshalb sehr wichtig, dass die Trackinginformationen vor dem Aufruf der Website entfernt werden.

Dazu haben wir die URL-Cleaner-Komponente entwickelt, welche Adressen mit Trackinginformationen als Eingabe erwartet und bereinigt zurück gibt.

Wir haben zwei mögliche Verfahren entdeckt, wie die Trackinginformationen in Angebotlinks eingefügt werden können. Dies ist sowohl über Weiterleitungen als auch URL-Parameter möglich. Der URL-Cleaner wendet deshalb zwei Strategien sukzessive an, um die Trackinginformationen zu entfernen:

Die erste Strategie bereinigt die URL von Weiterleitungen. Dabei gibt es zwei mögliche Verfahren der Weiterleitungsdienste:



Beim ersten Verfahren leitet der zwischengeschaltete Server den Besucher anhand der mitgesendeten URL-Parameter auf die korrekte Seite weiter. Die ursprüngliche URL ist hierbei somit bereits in der Modifizierten enthalten.

Das zweite Verfahren ist etwas schwieriger, da anstatt der Ziel-Adresse lediglich ein kryptischer String mitgesendet wird. Die Weiterleitung ist erst nach der serverseitigen Zuordnung des kryptischen Strings zur tatsächlichen Adresse möglich.

In der Tabelle 1 sind zwei Beispiele abgebildet, wie solche Weiterleitungen aussehen könnten. Eine Mischform der beiden Varianten ist ebenfalls möglich.

URL-Parameter	cptrack.com/?redir= <b>www.shop.de/product1</b>
krypt. Identifikationsstring	bit.ly/ <b>2Kqyrz2</b>

Tabelle 1: Beispiele der Weiterleitungsverfahren

Für die Bereinigung von Weiterleitungen wird die übergebene URL zunächst encodiert. Anschließend wird die Root-Url  Shops in der encodierten URL gesucht und alles davor entfernt. Für die Weiterleitungen, welche kryptische Identifikationsstrings verwenden, haben wir  eine zufriedenstellende Lösung finden können. Allerdings wird dieser Fall durch das Fehlen der Root-URL des Shops in der encodierten URL erkannt.

 `http://www.shop.de/product?partner=idealo?pid=96`


Tabelle 2: URL mit parameterbasierten Trackinginformationen

Die Tabelle 2 zeigt eine Beispiel-URL mit parameterbasierten Trackerinformationen. Dabei können ebenfalls Parameter enthalten sein, welche nicht für das Tracking verwendet werden.

Um die parameterbasierten Trackerinformationen zu entfernen, werden alle Schlüssel-Wert-Parameterkombinationen gelöscht, deren Schlüssel in einer vorher angelegten Liste vorkommen. Diese Liste enthält alle Tracker-Schlüsselnamen, die bei einer manuellen Recherche über mehrere hundert Shops häufiger vorgekommen sind.

Nachdem beide Strategien angewandt wurden, wird die bereinigte URL zurückgegeben.

## 2.8 Die Erstellung von Selektoren

Um einen Selektor zu erstellen muss zunächst ein konkretes Element der DOM-Hierarchie bestimmt werden. Dieses wird von dem Shop Rules Generator (SRG) in einem vorherigen Schritt ermittelt und stellt den Fundort für ein gewünschtes Produktattribut dar. Es wird zwischen den folgenden drei Knotentypen unterschieden: Textknoten, Beschreibungsknoten und Datenknoten. 

*Textknoten* sind Elemente, bei denen das gewünschte Produktattribut innerhalb eines Tag-Paars steht. Das Attribut ist somit ein sichtbarer Bestandteil der Browservisualisierung. Zu den *Beschreibungsknoten* gehören die Elemente, bei denen das gesuchte Produktattribut innerhalb der Attributliste des Elementes vorkommt. Dieses Attribut ist im Gegensatz zum Textknoten kein sichtbarer Bestandteil der Visualisierung.

In dem Quelltext 2.8 ist jeweils ein kurzes Beispiel für beide Typen aufgeführt. Der gesuchte Wert ist in diesem Fall die Produkteigenschaft EAN mit dem Produktattribut 9332721000108.

---

```
<div id='product-details'>
  ① <span>9332721000108</span>
</div>
② <meta itemprop='gtin13' content='9332721000108'>
```

---

Quelltext 1: Beispiele für einen ① Textknoten und einen ② Beschreibungsknoten

Die Selektoren der beiden Knotentypen sind ähnlich aufgebaut und bestehen jeweils aus einem CSS-Selektor. Der Aufbau des CSS-Selektors erfolgt analog zu der "Copy selectorFunktion der Chrome-Entwicklertools. Der Selektor für einen Beschreibungsknoten speichert zusätzlich einen Schlüssel, um das korrekte Elementattribut auszulesen.

Wir haben festgestellt, dass viele Internetshops Javascript auf ihrer Webseite verwenden. Oftmals sind in diesem Fall die produktspezifischen Daten bereits in einer strukturierten Form im Javascript als Objekt in der Javascript Objekt Notation (JSON) enthalten. In dem Quelltext 1 ist ein solches Skript vereinfacht dargestellt.

---

```
<script>
  var gtmGuaranteeProduct1 = {
    "name": "Versicherung für 2 Jahre",
  };
  var product2400462 = {
    "name": "Product",
    "ean": "9332721000108"
  };
</script>
```

---

Quelltext 2: Javascript, dass produktspezifische Informationen enthält

Der Aufbau eines Pfades durch eine Hierarchie hat beim DOM sehr gut funktioniert. Wir wollen daher dasselbe Prinzip für das Auslesen des Script-Tags anwenden. Nachfolgend wird anhand der Bestandteile des Datenselektors erklärt, wie die Struktur des Javascripts in eine hierarchische Form gebracht werden kann.

Der resultierende *Datenselektor* wird aus drei Teilen gebildet. Der erste Teil besteht, genau wie bei den anderen Selektortypen, aus einem *CSS-Selektor*. Dieser CSS-Selektor zeigt zu dem entsprechenden Script-Tag im DOM.

Um innerhalb des Javascriptes das richtige JSON-Objekt zu finden, wird ein Pfad verwendet, welcher durch die verschiedenen Code-Blöcke navigiert. Ein Block ist jeweils durch { und } markiert. Der *Block-Pfad* entsteht durch eine Tiefensuche durch die verschachtelten Blöcke und zeigt auf einen Block, welcher als JSON interpretiert werden kann. Für die Navigation durch das JSON-Objekt wird ein *JSONPath* erstellt. Ein JSONPath ist ähnlich einem XPath aufgebaut ist somit standardisiert nutzbar. Der resultierende Datenselektor sieht wie in Tabelle 3 dargestellt aus.

CSS-Selektor	html > head > script:nth-of-type(1)
Block-Pfad	-> 1
JsonPath	\$['ean']

Tabelle 3: Datenselektor für Beispiel 1

## 2.9 Die Trimm-Funktion

Häufig stehen vor und nach den gesuchten Produktattributen andere irrelevante Informationen. Vor dem gesuchten EAN steht zum Beispiel der String 'EAN:\_', welcher entfernt werden soll. Eine generische Verbesserung, welche auf alle Selektoren angewendet werden kann, ist die Trimm-Funktion.

Diese fügt zu den generierten Selektoren Informationen hinzu, wie viele Stellen links oder rechts abgeschnitten werden sollen. Diese bezeichnen wir als Left-Cut und Right-Cut. Sollte vor der gesuchten EAN immer derselbe String stehen, so gibt es keine Streuung in den Left-Cut-Werten. Wenn es sich bei dem Produktattribut um die Beschreibung handelt und diese durch manuelle Änderungen von idealo von denen der Webseite abweicht, so gibt es eine starke Streuung der Left- und Right-Cut-Werten.

## 2.10 Das Bewertungssystem für die Selektoren

Durch die verschiedenen Methodiken Selektoren zu erstellen und durch die zuvor beschriebene Trim-Funktion wurde die Flexibilität des Parser erhöht. Dies hat jedoch den Nachteil das ein sogenanntes *Rauschen* in den extrahierten Daten entsteht.

Dieses Rauschen entsteht dadurch, dass nun potenziell sehr viele Daten extrahiert werden, bei denen nicht mehr bekannt ist, welche extrahierten Informationen korrekt oder falsch sind. Damit die Anzahl der falsch extrahierten Produktattribute minimiert wird, haben wir ein Bewertungssystem für Selektoren eingeführt.

Nach der Erstellung aller Selektoren, wird jedem Selektor ein Qualitätsindex zugewiesen. Für jedes Angebot werden alle Selektoren angewandt und die extrahierten Produktattribute mit denen von idealo verglichen. Je nachdem ob der extrahierte Wert mit dem von idealo übereinstimmt, wird der Index erhöht oder verringert.

Ein Ausnahmefall bildet der leere String: Wird dieser als extrahierter Wert zurückgegeben, so bleibt der Qualitätsindex unverändert. Wir haben uns dafür entschieden, da man den leeren String von einem unerwünschten bzw. falschen Wert unterscheiden kann.

Bevor die Regeln mit dem Qualitätsindex in der Datenbank abgespeichert werden, werden diese auf den Intervall [0; 1] normalisiert. Alle Regeln mit einem normalisierten Index unter einem bestimmten Schwellwert werden verworfen.

Die Parser-Komponente verwendet den Qualitätsindex, um aus allen gefundenen Werten den Besten zu ermitteln. Dazu gruppiert sie nach den extrahierten Werten und summiert den normalisierten Index. Für jedes Produktattribut wird somit nur das beste Ergebnis gespeichert.

## 3 Die Genauigkeitsmessung des Extraktionsalgorithmus

Der entwickelte Algorithmus deckt die grundlegenden Elementtypen für die Datenextraktion ab und ist in der Lage bis zu 70 Seiten pro Sekunde zu verarbeiten. Wie bereits eingangs erwähnt wurde, ist die Qualität der Extraktion von großer Bedeutung - doch wie genau ist der entwickelte Algorithmus? Genau diese Frage wird in diesem Kapitel beantwortet, um eine grobe Aussage für das weitere Matchingverfahren zu treffen.

### 3.1 Die Testdaten der Evaluierung

Sowohl für das Antrainieren der Extraktionsregeln als auch für die Evaluierung der Ergebnisse wurden die von idealo zur Verfügung gestellten Angebotsdaten verwendet. Für die nachfolgenden Messungen wurden 7500 Angebote von 50 Shops genutzt.

Je Händler wurden max. 50 Angebote als Trainingsmenge für die Erstellung der Regeln und 100 Angebote als Testmenge für die Evaluierung verwendet. Die Auswahl der Shops erfolgte basierend auf den ersten Listeneinträgen der Shopübersicht<sup>6</sup> von idealo.

Alle nachfolgenden Messungen basieren auf einem Schnappschuss der Angebotsdaten inklusive der verlinkten Webseiten. Die Links wurden nicht durch die URL-Cleaner-Komponente bereinigt, da sonst die Angebotsdaten von idealo möglicherweise nicht mit denen von der Webseite übereinstimmen. Damit die Trackingstatistiken der Shopbetreiber nicht verfälscht werden, wurden nicht mehr als 150 Angebote je Shop heruntergeladen.

Eine Analyse der gesamten Testdaten hat ergeben, dass für jedes Angebot die Angaben zum Titel, dem Preis und der SKU existieren. Am seltensten existieren hingegen die HAN (68%) und die Produktbeschreibung (77%) eines Angebots.

Das Verhältnis der fehlenden zu den vorhandenen Produktattributen der Trainingsmenge ähnelt dem Verhältnis der Testmenge und weicht um maximal 0.88% bei der Produkteigenschaft "Marke" ab.

### 3.2 Die Messergebnisse

Für jeden Shop wurden 21 Regelmengen basierend auf der Trainingsmenge erzeugt, welche sich aus der Kombination verschiedener Konfigurationen ergeben. Diese Konfigurationen umfasst zum einen die Anzahl der verwendeten Angebote für das Antrainieren (*SaS*) und zum anderen den in Kapitel 2.10 eingeführten Filterschwellwert (*F*).

In allen Statistiken wurden die Fälle ignoriert, bei denen idealo keine Produktattribute gespeichert hat. Für diese ist es unmöglich zu entscheiden, ob die extrahierten Angebotsinformationen korrekt sind.

Die Bestimmung der Genauigkeit erfolgte durch die Anwendung aller Regelmengen auf die Testmenge. Anschließend wurde ausgewertet, wie oft der extrahierte Wert dem von idealo entspricht. Zudem wurde für die Bestimmung der Precision erfasst, ob der Wert leer ist. In der Tabelle 4 sind die typischen Kennziffern Accuracy und Precision für die verschiedenen Konfigurationen aufgeführt. Die Accuracy entspricht in diesem Fall gleichzeitig dem Recall. In etwa die Hälfte der Nichtübereinstimmungen weisen eine Levenshtein-Distanz  $\leq 3$  zum erwarteten Wert von idealo auf und sind somit sehr ähnlich. Eine manuelle Prüfung hat ergeben, dass oftmals Kodierungsfehler oder unterschiedliche Trennzeichen eine Übereinstimmung verhindern.

Eine nähere Betrachtung der Kennziffern je Attribut liefert Abbildung 2.

Wie erwartet, wird die Produktbeschreibung und die Kategorie selten extrahiert. Dies hängt vermutlich damit zusammen, dass diese Informationen am stärksten von idealo

---

<sup>6</sup><https://www.ideal.de/preisvergleich/AllePartner.html>

F\SaS	Accuracy in %			Precision in %		
	10	20	50	10	20	50
<b>0</b>	50.59	50.78	51.07	72.73	70.81	69.62
<b>0.5</b>	52.12	53.50	<b>53.98</b>	88.66	90.39	91.22
<b>0.6</b>	51.87	53.07	53.15	94.15	94.03	94.93
<b>0.7</b>	50.15	51.37	52.02	96.15	96.39	95.86
<b>0.8</b>	47.92	49.69	50.05	97.84	97.81	97.76
<b>0.9</b>	44.61	46.92	46.57	98.16	98.14	98.42
<b>1.0</b>	41.48	39.27	36.00	98.17	97.64	<b>98.90</b>

Tabelle 4: Accuracy und Precision bei unterschiedlichen Konfigurationen

manipuliert werden. Interessanterweise wird die HAN ebenfalls selten extrahiert, was vermutlich durch das häufige Fehlen der HAN in den Testdaten zusammenhängt. Die in der Bild-URL enthaltene ID und die Marke werden häufig gefunden und somit vermutlich sehr nützliche Features für den maschinellenlernbasierten Vergleich des Matchers sein.

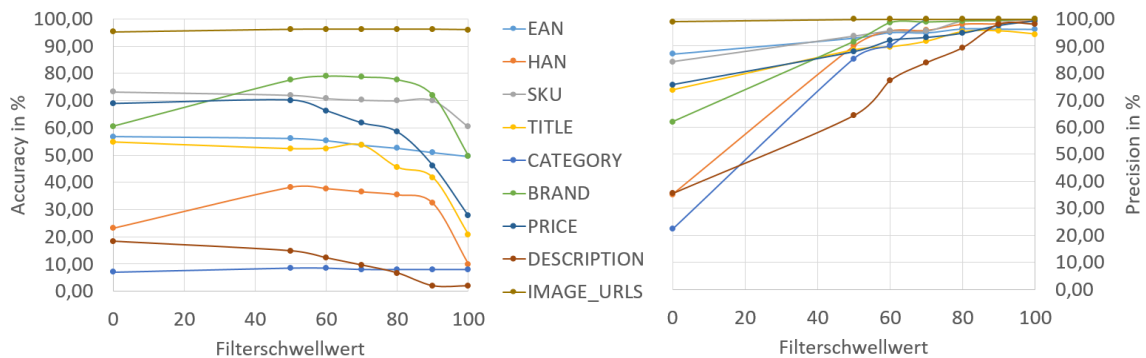


Abbildung 2: Accuracy (links) und Precision (rechts) pro Attribut für  $SaS = 50$

Zu guter letzt kann man dem Diagramm entnehmen, dass der Filterschwellwert ein gutes Mittel ist, um die Accuracy und Precision zu beeinflussen und ein höherer  $SaS$ -Wert tendenziell besser ist.

### 3.3 Mögliche Fehlerquellen der Messungen

Bei einer genaueren Analyse der Testdaten musste leider festgestellt werden, dass beim Herunterladen der Angebotsseiten eines Shops aufgrund einer Captcha-Absicherung keine nutzbaren Webseiten geliefert wurden. Für diesen Shop konnten folglich keine Regeln erstellt werden, was sich nachteilig auf die Accuracy auswirkt.

Des Weiteren trifft die in Kapitel 2.3 getroffene Annahme, dass die idealo-Daten nicht von denen der Shops abweichen, nicht zu. Die Angebotsdaten werden von idealo teilweise manuell oder durch Normalisierungsprozesse manipuliert, was die Regelerstellung erschweren kann. Zudem werden möglicherweise korrekt extrahierte Produktattribute fälschlicherweise als inkorrekt markiert.

## 4 Der Abschluss

Der Preisvergleich ist ein wichtiges Instrument, um die Markttransparenz im Internet zu gewährleisten. Damit ein möglichst objektiver Preisvergleich sichergestellt werden kann, ist es erforderlich, einen möglichst vollständigen Angebotskatalog zu vergleichen. Das Ziel des Softwaresystems *Scout* ist es, die Vollständigkeit des idealo-Angebotskatalogs zu untersuchen.

Für das Erfassen aller Produkte eines Onlinehändlers stellt die Datenextraktion einen wichtigen und schwierigen Schritt dar. In dieser Arbeit wurden zwei mögliche Herangehensweisen für das Extrahieren der Produktattribute vorgestellt. Der shop-spezifische Ansatz produziert im Vergleich zum shop-unspezifischen als zuverlässigere Lösung herausgestellt. Die Evaluierung hat ergeben, dass der Algorithmus je nach Anwendungsfall eine sehr hohe Präzision erreichen kann. Auf den Testdaten von idealo wurde zwar nur jedes zweite Attribut gefunden, dafür jedoch auch eine Präzision von über 95% erreicht.

Für die zukünftige Weiterentwicklung besteht noch Potenzial bei der Entwicklung weiterer Selektoren, um die Flexibilität des Parser weiter zu erhöhen. Des Weiteren kann man untersuchen, ob die Qualität der Selektoren durch eine gezielte Auswahl der Angebote, welche für das Anlernen verwendet werden, verbessert werden kann.

Abschließend möchte ich auf ein Problem bei der technischen Umsetzung im Zusammenhang mit der Bibliothek Jsoup<sup>7</sup> hinweisen: Für eine produktive Weiterentwicklung sollte man evaluieren, wie man eine höhere Flexibilität bei fehlerhaften HTML-Dokumenten gewährleisten könnte.

---

<sup>7</sup><https://jsoup.org>

## Das Literaturverzeichnis

- [1] ALLENSBACH, IfD: *Anteil der Online-Käufer an der Bevölkerung in Deutschland von 2000 bis 2016*. <http://de.statista.com/statistik/daten/studie/2054/umfrage/anteil-der-online-kaeufer-in-deutschland>. Version: 2016. – Zuletzt besucht: 2018-07-03
- [2] ALLENSBACH, IfD: *Anzahl der Internetnutzer in Deutschland, die das Internet nutzen, um Produktinformationen oder Preisvergleiche einzuholen, nach Häufigkeit der Nutzung von 2013 bis 2016 (in Millionen)*. <https://de.statista.com/statistik/daten/studie/171732/umfrage/nutzung-des-internets-fuer-produktinformationen-und-preisvergleiche/>. Version: 2016. – Zuletzt besucht: 2018-07-03
- [3] EUROSTAT: *Anteil der Online-Käufer in Europa nach ausgewählten Ländern im Jahr 2017*. <https://de.statista.com/statistik/daten/studie/153999/umfrage/anteil-der-online-kaeufer-in-europa-nach-laendern/>. Version: 2017. – Zuletzt besucht: 2018-07-03
- [4] EUROSTAT: *E-Commerce-Anteil am Gesamtumsatz der Unternehmen in ausgewählten Ländern in Europa im Jahr 2017*. <https://de.statista.com/statistik/daten/studie/73412/umfrage/e-commerce-anteil-am-gesamtumsatz-der-unternehmen-2008/>. Version: 2017. – Zuletzt besucht: 2018-07-03
- [5] POHLMANN, Jonas: *Hoch skalierbares fokussiertes crawling von Webshops*. 2018
- [6] SCHWARZBURG, Tom: *Unscharfer Angebotsabgleich im E-commerce mit maschinellem Lernen*. 2018
- [7] ZHAMANAKOV, Dmitrii: *Entwurf eines automatisierten Erfassungssystems für Produktangebote*. 2018

## **Die Selbstständigkeitserklärung**

Hiermit erkläre ich, die vorliegende Bachelorarbeit selbstständig erarbeitet und angefertigt, keine anderen als die angegebenen Quellen und Hilfsmittel genutzt und jegliche Zitate sowie Verwendungen fremder Inhalte kenntlich gemacht zu haben.

Potsdam, 20.07.2018

.....  
Leonardo Hübscher