Institut für Rechnergestützte Ingenieursysteme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3464

Internetgestützte Textanalyse zur Extraktion von Produktentwicklungswissen anhand von semi-strukturierten Dokumenten

Fan Zou

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. D. Roller
Betreuer/in:	M. Sc. Julian Eichhoff
Beginn am:	7. März 2013
Beendet am:	6. September 2013
CR-Nummer:	I.2.7, I.5.2, J.6

Kurzfassung

Mit der Popularisierung und Entwicklung des Internets in den letzten Jahrzehnten tauchen immer mehr elektronische Dokumenten im Internet auf. Zahlreiche Produktspezifikationen sind über das Internet z.B. in Form von Web-Seiten oder PDFs zugänglich. Diese Arbeit hilft den Unternehmen, die Produkte und das Produktentwicklungswissen aus den Webseiten automatisch zu extrahieren. In dieser Arbeit werden die Definition der Product Named Entity, die Konstruktion der Corpus, die Identifizierung von Product Name Entity und schließlich die Extraktion von Produktnamen und Produktentwicklungswissen erforscht. Die Arbeit betrifft die folgenden Aspekte:

1. Nach der Untersuchung von Produktenamen in Web-Seiten definieren wir die verschiedenen Komponenten von Produktnamen. Mit der Definition entwickelten wir eine Rechtlinie für die Markierung des Korpus. Danach erstellen wir einen Product Named Entity Korpus durch die Nutzung der halb-betreuten Lernmethode.
2. Nach den Merkmalen des Produktnames unterteilen wir die Indentifizierung des Produktnames auf zwei Phasen. Die erste Phase erkennt den Brandname, den Serienname und den Typenname eines Produkts. Basierend auf den ersten Ergebnissen wird der Produktname in der zweiten Phase erkannt werden. Für die Erkennung von diesen zwei Phasen können wir verschiedene Methoden verwenden. In der Arbeit werden das Hidden Markov Modell, Maximum Entropy Modell und das Conditional Random Field Modell diskutiert. Nach dem Vergleich der drei Metholden nutzen wir das Conditional Random Field Modell.
3. Nachdem die Produktnamen erfolgreich erkannt werden, werden die Produktnamen, die Produktmerkmale und die Restriktionen zwischen Produkten extrahiert.

Abstract

With the popularization and development of internet in the past few decades, more and more electronic documents appear on the Internet. Numerous product specifications are available via Internet, eg available in the form of web pages or PDFs. This dissertation helps the company to automatically extract the products, product sepecifications and product restriction from the web site. In this paper, We research on the definition of product named entity, the construction of the corpus, and the recognition technologies. This work concerns the following aspects:

1. After studying many of product names in web pages, we define the various compositions of product name entity. With this definition, we developed a rule for the corpus annotation. Then we create a product named entity corpus by using the semi-supervised method.
2. According to the features of the product names we divided the recognition of product names into two phases. The first phase detects the brand name, the series name and the type of a product. Based on the first results the product name will be recognised in the second phase. For the recognition in these two phases, many methods can be used. In this work we discuss hidden Markov model, maximum entropy model and Conditional Random Field model. After comparing these three models we decide to use conditional Random Field Model to do the recognition.
3. After the product names are successfully detected, the products, the product features and the restrictions between products will be extracted.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1.

Einleitung

In der heutigen Zeit ist das Internet ein wichtiges Medium für den Zugang zu Informationen. Im Internet existieren momentan über 4,5 Billionen Webseiten [Kun][1]. Diese zahlreichen Webseiten bringen entscheidende Vorteile für den Benutzer wie z.B. bei der Produktauswahl können die Verbraucher die Spezifikationen und die Preise von mehreren Produkten vergleichen. Für Unternehmen spielen diese Vorteile sogar noch eine größere Rolle.

1.1. Hintergrund

Zahlreiche Produktspezifikationen sind über das Internet z.B. in Form von Web-Seiten oder PDFs zugänglich. Aus den Spezifikationen können Verbraucher herausfinden, mit welchen anderen Produkten das vorhandene kombinierbar ist. Etwa zu welchen Kameras ein „Canon EF mount Objektiv“ passt. Unternehmen, die existierende Produkte als Einzelteile, Baugruppen, Geräte oder Apparate in einer von ihnen zu konstruierenden Maschine oder Anlage verbauen möchten, benötigen ebenfalls solche Spezifikationen, um die Kompatibilität mit dem Gesamtsystem und die Erfüllung der Konstruktionsziele sicherzustellen.

Es ist für Unternehmen sinnvoll, relevante Produktdaten aus dem Internet möglichst automatisch zu extrahiert und zu identifizieren, um diese relevanten Informationen aus dem Internet effizient nutzen zu können. Aber bei der Extraktion ist es oftmals schwierig herauszufinden, ob die Webseite ein Produkt behandelt, um welche Produkte geht es, und handeln die Informationen von Restriktionen oder den Spezifikationen.

Der Schlüssel für die Lösung dieser Probleme ist die automatische Erkennung von Produkten. Diese Erkennung des Produkts gehört zur „Named Entity Recognition“ (Kurz NER), die eine Teilaufgabe der Extraktion von Informationen ist. NER lokalisiert und klassifiziert die Wörter und Sätze im Text in vordefinierten Kategorien [Kim+04] wie die Namen der Personen, die Organisationen, die Orte, die Geldwerte, die Prozentsätze usw. Im Vergleich zu anderen normalen NER ist „Product named entity recognition“ (Kurz PNER) ein relativ neuer Bereich von Erkennungssystem.

[1] `http://www.worldwidewebsize.com/`

NER ist eine wichtige Grundlage für die Technologien zur Verarbeitung natürlicher Sprache. Sie hat einen wichtigen Einfluss auf die syntaktische und semantische Analyse. Darüber hinaus ist die NER für die Informationsextraktion, Informationsfilterung, das Information Retrieval, maschinelle Übersetzung und automatische Antwort-Systeme von großer Bedeutung. Es folgen einige Anwendungsbereiche:

1. Informationsextraktion

   Für Informationsextraktion sollen spezifische Informationen aus dem Text automatisch extrahiert werden. Dann werden strukturierte Daten gebildet. Zum Beispiel können die Zeit, der Ort und die Personen von einem Ereignis aus einem Text extrahiert werden. NER ist der erste Schritt in der Aufgabe der Informationsextraktion. Sie ist die Grundlage für weitere Schritte. Zum Beispiel muss der Ausdruck der Zeit erkannt werden.

2. Information Retrieval

   Named Entity (kurz NE) ist die Grundeinheit der Informationen in dem Text. Durch die Extraktion und Analyse von NE im Text können die entsprechenden Unterlagen genauer lokalisiert werden. Dann wird die Informationsrückgewinnung schneller und genauer.

3. Maschinelle Übersetzung

   Für die maschinelle Übersetzung ist es oft notwendig, die Eigennamen wie Personennamen, Orte oder Organisationen genau zu übersetzen. In dieser Situation gibt es eine große Anzahl von Eigennamen, die nicht manuell übersetzt werden können. Deshalb ist die genaue und effiziente automatische Extraktion und Identifizierung der Named Entities (kurz NEs) für die Verbesserung der Genauigkeit und Nützlichkeit maschineller Übersetzungen von großer Bedeutung.

4. Automatische Antwort-Systeme

   Vorher basierten die Suche-Maschinen auf Schlüsselwörtern. Jetzt gibt es eine neue Suchmethode, die auf dem Antwort-System basiert. Sie ist eine erweiterte Form des Information Retrievals. Bei der Suche fragen die Suchenden anstatt nach der Angabe von Schlüsselwörtern einfach mit natürlicher Sprache und bekommen eine prägnante und präzise Antwort. Die Antworten sind meistens NEs. Zum Beispiel für die Frage „Welche neue Handys gibt es im Jahr 2013“ sind die Antworten „Nokia Lumia 920, Samsung Galaxy S4, ..“, alle NEs. Deswegen hat die NER auch einen großen Einfluss auf automatische Antwort-Systeme.

5. Automatisches Tagging für Web-Seiten

   Die NEs wie die Marke oder Personennamen können auf der Web-Seite markiert, verlinkt oder mit Werbungstags gelabelt werden, damit der Nutzer diese Webseite besser verstehen kann und das Unternehmen eine Werbewirkung bekommen kann. Zum Beispiel durch die Markierung der Produktnamen und ihrer Spezifikationen mit

verschiedenen Farben auf der Webseite kann der Nutzer die Produktinformationen klar auf einen Blick bekommen.

1.2. Aufgabenstellung

Das Ziel dieser Arbeit ist eine Methodik zu entwickeln, um die Produktnamen, die Spezifikationen der Produkte und die Restriktionen der Produkte aus den englischen Webseiten automatisch zu erkennen und zu extrahieren.

Die Arbeit ist in folgender Weise gegliedert:

1. Analyse der Struktur und Eigenschaften der Produktnamen auf den Webseiten.
2. Analyse der vorhandenen Technologien, Methoden und Werkzeuge zur Extraktion von Daten aus dem Internet.
3. Analyse der vorhandenen Methoden und Strategien der NER und dann PNER.
4. Erarbeitung eines Konzepts für PNER durch eine Umsetzung der analysierten Strukturen und Methoden.
5. Entwicklung einer prototypischen IT-Umsetzung.
6. Implementierung des Prototypen auf dem Google App Engine.
7. Bewertung der erarbeiteten Ergebnisse auf Basis von Beispielen.

1.3. Lösungsansatz

Um die semistrukturierten Daten am Beispiel von Produktname aus Web-Seiten zu extrahieren und zu klassifizieren bietet diese Arbeit eine Methodik an. In dieser Methodik für die PNER wird das Machine-Learning-Verfahren verwendet, das auf halb-betreuten Lernvorgängen basiert. Für den Lernalgorithmus benutzen wir die Technik der Conditional Random Fields (kurz CRFs). Die CRFs ermöglichen die Klassifikation der aus einer Webseite extrahierten Daten. Anhand dieser Klassifikation werden die Namen, Spezifikationen und Restriktionen von Produkten aus Webseiten erkannt und ausgegeben.

Für das Testen dieser Methodik wird eine Anwendung „productextraction“[2] auf der Plattform Google App Engine[3] (Kurz GAE) entwickelt.

[2] http://productextraction.appspot.com/
[3] https://appengine.google.com/

1.4. Geliederung

In dieser Diplomarbeit gibt es folgende Kapitel:

Kapitel 1 – Einleitung: Hier werden der Hintergrund und die Ziele dieser Arbeit dargelegt.

Kapitel 2 – Grundlagen: beschreibt die Aufgabe, Schwierigkeiten und die Klassen der Methoden von NER.

Kapitel 3 – Product Entity Named recognition im Bereich von Elektronik und Technik: entwickelt ein Konzept für PNER im Bereich von Elektronik und Technik.

Kapitel 4 – Prototypische Implementierung: Implementierung eines Prototypen auf der Google App Engine.

Kapitel 5 – Evaluation: evaluiert die Ergebnisse.

Kapitel 6 – Zusammenfassung und Ausblick: Zuerst wird zusammengefasst und dann ein Ausblick gegeben.

Kapitel 2.

Grundlagen

Im Kapitel 1.1 ist beschrieben, dass die Erkennung von Produktnamen zu NER gehört. Dieses Kapitel stellt wichtige Begriffe und Definitionen von NER dar, auch wird ein Überblick über die Grundlagen dieser Arbeit gegeben. Hier werden die Aufgaben, die Probleme und die Methoden von NER vollständig vorgestellt.

2.1. Named Entity Recognition

Der Begriff „Named Entity“, der heute weitgehend im Natural Language Processing verwendet wird, wurde für die sechste Message Understanding Conference (kurz MUC) [GS96] geprägt.

Damals konzentrierte sich MUC auf die Information Extraction (Kurz IE) Aufgaben, bei denen die strukturierten Informationen der Unternehmensaktivitäten aus unstrukturierten Texten extrahiert wurden. Beispielweise wie die Zeitungsartikel. Bei der Definition der Aufgabe bemerkte man, dass es unerlässlich ist, um die Informationseinheiten wie Namen zu erkennen, einschließlich Person, Organisation und numerische Ausdrücke wie Uhrzeit, Datum, Geld und die Ausdrücke von Prozent anzugeben. Das Identifizieren dieser Einheiten im Text wurde als eine der wichtigsten Teilaufgaben der IE anerkannt und wurde als „Named Entity Recognition and Classification (kurz NERC)“ [NS07] umgesetzt. Seitdem ist NER als wichtiger erster Schritt in der Anwendung für die Verarbeitung natürlicher Sprache, wie Textzusammenfassung, Filterung der Information, Extraktion der Beziehung und Beantwortung der Frage anerkannt worden.

Kategorien	Beispiele
Datum	10.06.2013 oder 10.Juli.2013
Zeit	13:32 oder 8 pm
Name	Mirko oder Laura Sonntag
Lokation	Stuttgart oder Hauptbahnhof
Phone Nr.	0711-1234567 oder +4917612345

Tabelle 2.1.: Beispiele der Kategorien von NE

2.1.1. Aufgabe der Named Entity Recognition

Die Aufgaben von NER wurden zuerst auf der MUC-7 veröffentlicht, die 1997 stattfand [CR97]. Einfach ausgedrückt besteht die Aufgabe meist aus zwei Teilen:

1. Identifikation der Grenze der Einheit.
2. Festlegen der Klasse der Einheit (Personenname, Ort oder anders).

Genauer gesagt kann NER als ein Problem betrachtet werden, das die Folge von Wörtern in einem Satz eines Textblocks in eine Folge von entsprechenden Kategorien zuordnet. Ein typisches NER-System nimmt als Eingabe ein Stück Text, wie

„The Chinese president Hu Jintao has visited the Hongkong Red Cross in Kowloon.“,

und gibt eine Folge von mit Namen markierten Wörtern aus, wie

„The /O Chinese /O president /O Hu Jintao /PER has /O visited /O the /O Hangkong Red Cross /ORG in /O Kowloon /LOC.“,

wo das Label PER sich auf den Personenname bezieht, LOC Ortsname, ORG Organisationsnamen und O kennzeichnet Wörter, die sich außerhalb der Menge von definierten Kategorien befinden.

Nach den Anforderungen der Anwendung oder der Unterschiede der identifizierten verschiedenen Objekte kann die NER in zwei Kategorien unterteilt werden:

1. Traditionelle NER wie die Erkennung von Personenname, Adresse, Organisation, Uhrzeit und Prozent.
2. NER für Eigenname im spezifischen Bereich. Beispielweise die Erkennung der Produktnamen im E-Commerce und die Erkennung von Gen-und Protein-Namen im Bereich der Biologie.

Derzeit ist die Technologie der traditionellen NER relativ reif. Für die Erkennung im Bereich der Biologie gibt es auch schon zahlreiche Studien. Allerdings ist die Technologie der Erkennung von Produktnamen noch in den Anfängen. In dieser Arbeit, basierend auf der traditionellen NER, werden wir eine Lösung finden, um den Produktnamen im E-Commerce zu erkennen.

2.1.2. Herausforderungen der Named Entity Recognition

Trotz der klaren Definitionen der Klasse der Einheit geschieht es sehr oft, dass eine Zeichenfolge mehrere Typen der Einheit darstellen kann, je nach Kontext. Zum Beispiel das Wort *„Hongkong“* im Satz *„ wir treffen uns in Kongkong“* ist ein Ort. Aber im *„Kongkong red cross“* bezieht es sich auf eine Organisation. Dieses Phänomen ist als Metonymie bekannt. Eine einfache Lösung ist, die Metonymie von Wörtern einfach zu ignorieren. In diesem Fall beispielsweise kann Kongkong immer als Organisation gelabelt werden. Allerdings für

praktische Anwendungen von NER ist dieser Ansatz nicht sehr nützlich. Auf der anderen Seite sind die idealistischen Lösungen nicht immer praktisch zu implementieren.

Ein weiteres wichtiges Thema ist, dass die Texte aus verschiedenen Textsorten (wissenschaftliche, journalistische, informelle, usw.) und Domains (Wirtschaft, Sport, E-Commerce, usw.) kommen. Der Stil eines Textes kann durch eine Reihe von Faktoren beeinflusst werden. Wie z.B. die Form von Medien (z. B. Web-Seiten , E-Mails, geschriebener Text), die Textsorte (z.B. Briefe, Bücher, Berichte), und der Autor. Zum Beispiel können die weniger formellen Texte der üblichen Großschreibung, dem Satzzeichen oder sogar der Rechtschreibung nicht folgen. Einige Studien haben deutlich gezeigt, dass obwohl jede Domain angemessen unterstützt werden kann, ist die Portierung eines Systems auf eine neue Domain oder ein textliches Genre noch eine große Herausforderung. [May+01] [MWC05]

2.2. Lernmethoden für NER

Bisher gibt es schon viele internationale Studien über NER. Im Anfang konzentrierten sich die Forscher auf manuell erstellte Regeln, um die NE zu erkennen. Jetzt verwenden die Forscher für NER die betreute Lernmethode (kurz BL) des maschinellen Lernens. Aber obwohl diese betreute Methode eine hohe Performance hat, braucht sie einen großen Textkorpus, der manuell markiert ist. Wenn es an einem manuell markierten großen Korpus mangelt, können die halb-betreute Lernmethode (Kurz HBL) oder die unbetreute Lernmethode (Kurz UBL) verwendet werden.

2.2.1. Betreute Lernmethode

Für NER ist derzeit die betreute Lernmethode die beliebteste Methode. Diese Methoden umfassen Hidden Markov Model (kurz HMM) [Bik+97], Maximum Entropy Model, (kurz ME) [Bor+98], Decision Tree Model [Sek98], Support Vector Machine(Kurz SVM) [AM03] und Conditional Random Fields [ML03].

Bikel et al. haben das Standard-NER verbessert. Basierend auf HMM haben sie eine Methode für NER vorgeschlagen. Sie haben ein gutes Erkennungsergebnis am MUC-6 Korpus bekommen[Bik+97].

Borthwick war der erste, der ME in NER angewendet hat. Er hat eine Erkennungsmethode für englische und japanische NE entwickelt[Bor+98]. Basierend auf den globalen Eigenschaften von ME haben Hai Leonq Chieu et al. [CN03] eine Erkennungsmethode vorgeschlagen und auf dem Korpus von MUC-6 und MUC-7 getestet. Ihre Testergebnisse haben gezeigt, dass die Verwendung von globalen Eigenschaften die Performance der Erkennung auf den Korpus von MUC-6 von 90,75% auf 93,27% und die Performance auf den Korpus von MUC-7 von 85,22% auf 87,24% verbessert hat.

Masayuki Asahara et al. [AM03] haben ein Schriftzeichen-basiertes Chunking Verfahren entwickelt. Zunächst wird der eingegebene Satz durch einen statistischen morphologischen

Analysator redundant analysiert, um mehrere (n-beste) Antworten zu produzieren. Dann ist jedes Schriftzeichen mit seinen Schriftzeichen-Typen und ihre möglichen POS-Tags der obersten n-besten Antworten kommentiert. Schließlich nimmt ein SVM-basierter Chunker einige Teile des eingegebenen Satzes als NEs. Die Autoren haben diese Methode zur Extraktion der IREX NE Aufgabe angewendet. Die F-Measure der Ergebnisse betrug 87,2%.

Andrew McCallum et al. [ML03] haben eine Erkennungsmethode für NER entwickelt, die auf CRFs basiert. Für die Aufgabe der Conference on Computational Natural Language Learning 2003 (Kurz CoNLL-2003)[1] [Con] haben die Autoren diese CRFs Methode verwendet. Sie haben einen F1 Wert von 84,04% für die Erkennung der englischen NEs im englischen Nachrichtenbereich bekommen. Für die Erkennung der deutsche NEs haben sie nur einen F1 Wert von 68,11% erhalten. Burr Settles [Set04] hat das CRFs für die NER im Bereich der Biologie eingesetzt. Wenliang Chen et al. [CZI06] haben eine auf CRFs basierende Methode für die Erkennung der chinesischen NEs vorgeschlagen. Ihr System hat einen F-Wert von 85,14% auf dem Korpus von Microsoft Research (kurz MSRA), 89,03% auf dem Korpus von City University of Hong Kong (kurz CityU) und 76,27% auf dem Korpus von Linguistic Data Consortium (kurz LDC) erreicht.

Alle oben erwähnten betreuten Lernmethoden brauchen einen großen Korpus, der manuell markiert ist. Die Größe und die Qualität des Korpus haben normalerweise einen Einfluss auf die Performance des Erkennungssystems. Deshalb: Wenn es an einem großen hochwertigen Korpus mangelt, können die halbe-betreute Lernmethode oder die unbetreute Lernmethode verwendet werden.

2.2.2. Halb-betreute Lernmethode

Die halb-betreute Lernmethode wird auch als schwach betreute Lernmethode bezeichnet. Die Grundtechnik der HBL ist das Bootstrapping und sie hat nur eine geringe Betreuung. Es ist wie eine Menge von Saatgut, die einen Lernprozess starten [NS07]. Zum Beispiel für die Extraktion der Namen des Albums können einige Albumnamen zunächst manuell als Beispielnamen gesammelt werden. Dann sucht das System nach Sätzen, die diese Beispielnamen enthalten und versucht, einige Hinweise aus dem Kontext zu identifizieren. Dann versucht das System andere NEs des Albums zu finden, die in ähnlichen Kontexten erscheinen. Die neu gefundenen Albumnamen werden erneut als Beispielnamen für die Erkennung angewendet, um neue relevante Hinweise aus dem Kontext zu entdecken. Durch Wiederholen dieses Vorgangs werden schließlich eine große Anzahl von Albumnamen und eine Vielzahl von Kontexten gesammelt.

Basierend auf regulären Ausdrücken hat Sergey Brin [Bri99] eine Methode für die Extraktion der NEs und der Beziehungen zwischen NEs entwickelt. Mit dieser Methode können die Buchnamen und die Namen der Autoren aus dem Internet automatisch extrahiert werden. Diese Methode gilt nur für Englisch.

[1] http://www.cnts.ua.ac.be/conll2003/ner/

Ellen Riloff und Rosie Jones [RJ99] haben eine Lernmethode eingeführt, die auf dem gegenseitigen Bootstrapping basiert. Mit dieser Methode werden zunächst einige NEs, die bereits mit bestimmten Typen markiert sind, manuell als Samen gesammelt werden. Dann werden alle Muster, die rund um diese Samen in einem großen Korpus gefunden werden, gesammelt und geordnet. Danach werden diese Muster genutzt, um neue NEs zu finden.

Der Nachteil der Methode von Riloff und Jones ist, dass die Performance des Erkennungssystems schnell verschlechtert werden kann, wenn ein Rauschen in die Liste der NEs oder in die Muster eingeführt wird. Deswegen haben Alessandro Cucchiarelli und Paola Velardi [CV01] die Methode von Riloff und Rosie verbessert. Zusätzlich zum Muster werden syntaktische Beziehungen benutzt, um genauere kontextuelle Hinweise rund um die NEs zu entdecken. Interessanterweise werden statt manuellen markierenden NEs die NEs, die vom existierenden NER System generiert wurden, als Samen genutzt.

Durch die Inspiration der Arbeit von Riloff haben Marius Paşca et al. [Pas+06] eine Methode entwickelt, in der „distributional similarity“ eingesetzt wird. Durch den Einsatz von „distributional similarity“ können Synonyme erzeugt und die Menge der Muster vergrößert werden. Die Autoren haben 10 Beispiele aus Fakten (Name, Geburtstag) als Samen verwendet. Ausgehend von diesem Samen haben sie eine Millionen Fakten aus 100 Millionen Web-Dokumenten extrahiert. Die Genauigkeit beträgt 88%.

Bald hat Marius Paşca [Pasimtech7] eine andere Methode für NER vorgeschlagen, die sich an anonymisierten Web Search Queries orientiert. Die Performance dieses NER Systems ist gut, aber nur wenn die zu extrahierenden NEs den gleichen Typ mit den Samen haben. Um diesen Mangel zu beheben, haben Jiafeng Guo et al. [Guo+09] eine auf WS-LDA (Weakly Supervised Latent Dirichlet Allocation) Model basierende Methode für NER entwickelt.

2.2.3. Unbetreute Lernmethode

Für unbetreutes Lernen wird hauptsächlich Clustering verwendet. Zum Beispiel nach der Ähnlichkeit der Kontexte kann man die NEs in verschiedenen Gruppen sortieren. Es gibt noch andere unbetreute Methoden. Grundsätzlich hängen diese Methoden von lexikalischen Ressourcen, von lexikalischen Mustern und von einem großen nicht markierten Korpus ab.

Basierend auf WordNet haben Enrique Alfonseca et al. [AM02] eine Methode für NER vorgeschlagen. Zunächst werden alle Synonyme im WordNet mit einer Klasse gelabelt. Dann wenn ein Wort von einem bestimmten Dokument eingegeben ist, wird sein Kontext analysiert. Dann wird sein Synonym herausgefunden, welches die maximale Ähnlichkeit mit dem Wort hat. Die Klasse dieses Synonyms wird als die Klasse der eingegebenen Worte ausgegeben.

Yusuk Shinyama u.a. [SS04] haben bemerkt, dass NEs oft synchron in mehreren Nachrichten erscheinen. Diese Eigenschaft ermöglicht die Erkennung der NEs. Durch den Vergleich von zwei verschiedenen Nachrichtenartikel, die gleichzeitig in verschiedenen Nachrichtenquellen erscheinen, können die seltenen NEs identifiziert werden. Die Genauigkeit ihres Experiments beträgt 90%. Aber die Trefferquote ist niedrig.

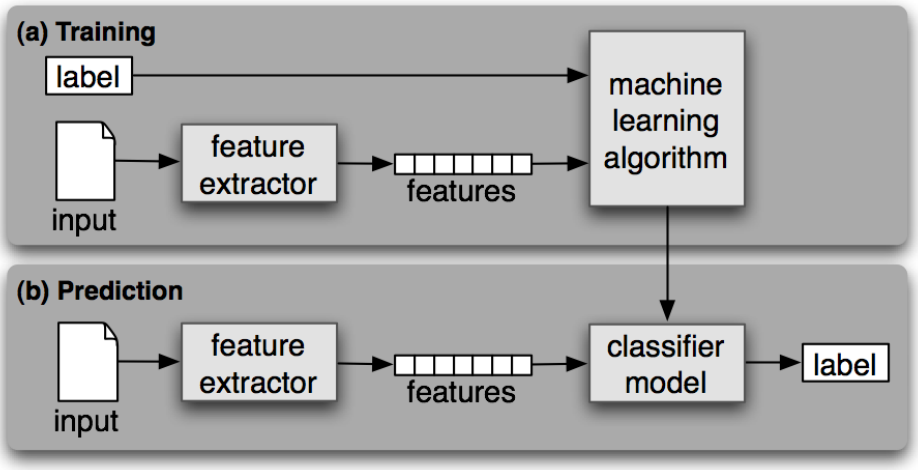

Abbildung 2.1.: Maschinelles Lernen

2.3. Maschinelles Lernen

Die betreute Lernmethode ist derzeit die beliebteste Methode für NER. Im Kapitel 2.2.1 auf Seite 15 sind fünf maschinelle Lernmethoden (HMM, ME, SVM, CRFs und Decision Tree) kurz beschrieben. Im diesem Kapitel werden die Grundlagen von HMM, ME und CRFs weiter dargestellt.

2.3.1. Mathematische Definition des Problems der NER

Die Aufgabe von NER kann als ein Problem der Markierung der Textsequenz behandelt werden. Diese Aufgabe lässt sich wie folgt beschreiben. Gegeben sind eine Sequenz der Beobachtungen

$$\vec{x} = (x_1, ..., x_n)$$

und eine Menge der Klasse Label Y. Wir finden die Sequenz der Klasse Label

$$\vec{y} = (y_1, ..., y_n) \in Y^2$$

mit der höchsten bedingten Wahrscheinlichkeit unter allen möglichen Sequenzen von Label:

$$\vec{y}^* = \underset{\vec{y}}{argmax}\ p(\vec{y} \mid \vec{x})$$

X	x_1	x_2	x_3	x_4	x_5	x_6
Sequenz der Beobachtungen	Hu	Jingtao	has	visited	Red	Cross
Y	y_1	y_2	y_3	y_4	y_5	y_6
Sequenz von Label	B	I	O	O	B	I

Tabelle 2.2.: Beispiel der Markierung der Textsequenz

Ein Beispiel für die Markierung der Textsequenz wird in Tabelle 2.2 gezeigt. Das Label „B“ (begin) steht für den Anfang einer NE und Label „I“ (inside) steht für innerhalb einer NE. Das Label „O“ (outside) bedeutet außerhalb einer NE.

Im Folgenden beschreiben wir einige verschiedene Ansätze, um die bedingte Wahrscheinlichkeit einer Sequenz von Labeln zu schätzen.

2.3.2. Hidden Markov Models

Das Hidden Markov Model ist eine leistungsfähige statistische Technik zum Maschinenlernen. Es ist ein generativ-globales Modell [BSW99] und wurde in vielen Informationsextraktionsaufgaben eingesetzt. Dieses Verfahren wurde von Baum und Pietre [BP66] entwickelt.

Nach der Arbeit von Jurafsky und Martin [JM08] besteht dieses HMM aus folgenden Elementen:

1. Eine Menge aus $|Q|$ Zuständen:

   $$Q = \{q_1, q_2, ..., q_{|Q|}\}$$

2. Eine Sequenz von $|O|$ Beobachtungen aus einem Alphabet V :

   $$O = o_1 o_2 ... o_{|O|}$$

3. Die Emissionswahrscheinlichkeiten (Beobachtungswahrscheinlichkeiten):

   $$B \in R^{|O|\times|Q|}, b_i(o_t) = P(o_t \mid q_i)$$

4. Die Transitionswahrscheinlichkeiten:

   $$A \in R^{|Q|\times|Q|}, a_{ij} = P(q_i \mid q_j)$$

   a_{ij} ist die Wahrscheinlichkeit von q_i in q_j überzugehen.($\sum_{j=1}^{|Q|} a_{ij} = 1, \forall i$)

5. Die Initialwahrscheinlichkeiten ($\sum_{j=1}^{|Q|} \pi_i = 1$):

   $$\Pi \in R^{|Q|}, P(q_i \mid START) = \pi_i$$

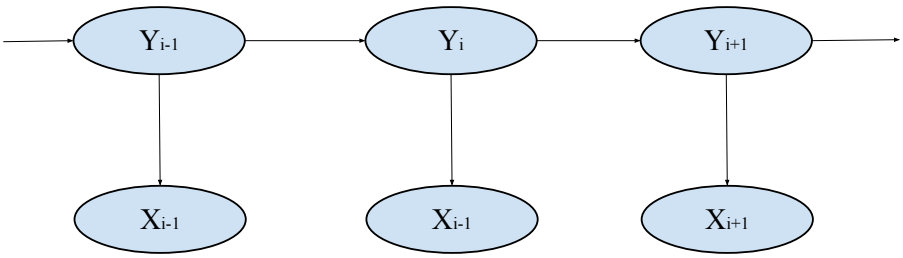

Abbildung 2.2.: Konzeptionelle Architektur des HMM

6. Eine Menge akzeptierender Endzustände:

   $\mathcal{F} = \{q_i, q_j ...\}$

Abbildung 2.2 zeigt die allgemeine Architektur eines instanziierten HMM. Jede ovale Form stellt eine Zufallsvariable dar, die eine beliebige Anzahl von Werten annehmen kann. Die Zufallsvariable Yi ist der versteckte Zustand an sequentieller Position i (d.h. das Label für das Token an Position i). Die Zufallsvariable Xi ist die Beobachtung an sequentieller Position i (d.h. das Wort-Token an Position i) der Sequenz der angegebenen Tokens. Die Pfeile in der Abbildung bezeichnen bedingte Abhängigkeiten.

Im HMM existieren drei fundamentale Probleme, die Rabiner (Rabiner 1989) in seiner Arbeit ermittelt hat:

1. Das Auswertungsproblem:

   Gegeben is eine Beobachtungssequenz $O = o_1 o_2 ... o_{|O|}$ und ein HMM $\lambda = (A, B, Q, \mathcal{F}, \Pi)$, ermittle die Wahrscheinlichkeit $P(O \mid \lambda) = \sum_S P(O, S \mid \lambda)$.

2. Das Decodierungsproblem:

   Gegeben ist eine Beobachtungssequenz $O = o_1 o_2 ... o_{|O|}$ und ein HMM $\lambda = (A, B, Q, \mathcal{F}, \Pi)$, berechne beste versteckte Zustandssequenz $S = s_1 s_2 ... s_{|O|}$ mit $\max_S P(O, S \mid \lambda)$.

3. Das Lernproblem:

   Gegeben ist eine Beobachtungssequenz $O = o_1 o_2 ... o_{|O|}$ und eine Menge von Zuständen Q, ermittle die Wahrscheinlichkeiten A , B und λ , die $P(O \mid \lambda)$ maximieren.

Für die oben genannten drei Probleme gibt es viele Algorithmen: Der Forward- und Backward-Algorithmus sind für das Auswertungsproblem. Das Decodierungsproblem lässt sich effizient mit dem Viterbi-Algorithmus lösen. Das Lernproblem ist auch lösbar mit Hilfe des Baum-Welch-Algorithmus.

Bisher gibt es schon viele Implementierungen von HMMs. Die bekanntesten Implementierungen sind die LingPipe[2] Pakete und Mallet[3] Pakete.

2.3.3. Maximum Entropy Model

Das ME Prinzip wurde von Edwin Thompson Jaynes [Jay57] eingeführt. Die Hauptidee ist, dass unter den Abwesenheiten der bekannten Informationen, das Verteilungsmodell keine Annahmen über den unbekannten Teil macht. Die Wahrscheinlichkeitsverteilung, die der Erkenntnis entspricht und welche die Entropie der A-priori-Wahrscheinlichkeit maximiert, werden gewählt[Nin+09].

Delta Pietra u.a. [DP+92] verwendeten ME in der Verarbeitung natürlicher Sprache (natural language processing, kurz NLP) erstmals im Jahr 1992. Viele Wissenschaftler verwenden ME in NLP, um das Problem der Text-Klassifikation, das Problem von Part-of-speech (Kurz POS) Tagging und das Problem der Identifizierung der Sätze zu lösen.

Mit dem Maximum Entropy Model, anstatt direkt Berechnen der Wahrscheinlichkeit $p(\overrightarrow{y_j} \mid \overrightarrow{x_j})$ zu berechen, zerlegen wir das Problem in eine Menge von lokalen Vorhersagen $p(y_j \mid x_j)$ an jeder Position in einer Eingabesequenz j, und dann kombinieren wir diese Vorhersagen in der endgültigen Lösung.

Lasse y das Label der Ausgabe und x die Kontextinformationen eines Wortes an einer bestimmten Position j in einer Sequenz $(\overrightarrow{y}, \overrightarrow{x})$ bezeichnen. ME ist ein Verfahren zum Schätzen der bedingten Wahrscheinlichkeit $p(y \mid X)$. Ist ein Kontexts x gegeben, wird der Prozess y ausgegeben. Dieser Ansatz basiert auf dem Prinzip der ME von E..T Jaynes [Jay57]. Nach seiner Meinung ist es die einzige unvoreingenommene Entscheidung, die Gleichverteilung angesichts der verfügbaren Informationen auszuwählen, wenn unvollständige Informationen über eine Wahrscheinlichkeitsverteilung verfügbar sind. Die Entropie ist ein mathematisches Maß dieser Gleichverteilung. Bei einer bedingten Verteilung $p(y \mid x)$ verwenden wir den Begriff der bedingten Entropie $H(y \mid x)$ definiert als

$$H(y \mid x) = -\sum_{(x,y)\in Z} p(y,x)\ log\ p(y \mid x).$$

Hier ist die Menge $Z = X \times Y$ aus X und Y. X ist die Menge aller möglichen Eingangsgrößen x und Y ist die Menge aller möglichen Ausgangsgrößen y. Die untere Grenze der Entropie ist null. Sie bedeutet, dass die Entropie eines Modells überhaupt ohne Unsicherheit ist (Das Ergebnis kann genau vorhergesagt werden). Die Entropie wird von oben durch $log\,|Y|$ begrenzt. In dieser Situation sind alle möglichen $|Y|$ werte gleich verteilt.

[2] http://alias-i.com/lingpipe/
[3] http://mallet.cs.umass.edu/

Das Prinzip ME hat vorgeschlagen, ein Model $p_*(y \mid x)$ zu verwenden, das nicht nur die höchste mögliche bedingte Entropie hat sondern auch in Übereinstimmung ist mit den Beweisen aus den Trainingsdaten:

$$p^*(y \mid x) = \underset{p(y|x) \in P}{argmax} H(y \mid x) \tag{2.1}$$

wobei P die Menge aller Modelle im Einklang mit den Trainingsdaten ist.

Wir stellen die Trainingsdaten in Bezug zu ihren nützlichen Tatsachen. Ein Beispiel dieser Tatsachen ist, dass die Frequenz, mit der „Hongkong“ als Lokation markiert ist, 0,1 beträgt. Diese Tatsache wird durch die stationären Funktionen $s_i(x, y) \in \{0, 1\} (1 \leq i \leq m)$ beschrieben, die von der Eingangsgröße x und dem Label y beide abhängen (zur Vereinfachung verzichten wir hier auf die Abhängigkeit der Eingangsfolge $\overrightarrow{x}$). Dann kann die Frequenz jeder derartiger Tatsachen als der Erwartungswert der entsprechenden Merkmalsfunktion s in Bezug auf die empirische Verteilungsfunktion $\tilde{p}(x, y)$ in den Trainingsdaten ausgedrückt werden:

$$\tilde{E}(s) = \sum_{(x,y) \in Z} \tilde{p}(x \mid y) s(x \mid y)$$

Jetzt haben wir die wichtigen statistischen inhärenten Tatsachen der Lernstichprobe formuliert. Wir verlangen, dass unser Modells des Prozesses auch ihnen entspricht. Wir realisieren das durch die Einschränkung des Erwartungswerts, das das Modell der entsprechenden Funktion zugewiesen wird. Der Erwartungswert einer Funktion s auf das Model ist:

$$E(s) = \sum_{(x,y) \in Z} \tilde{p}(x) p(y \mid x) s(x \mid y)$$

wobei $\tilde{p}(x)$ die empirische Verteilung von x in der Lernstichprobe ist.

Der Erwartungswert der jeweiligen Funktion s_i ist eingeschränkt (gleich wie der Erwartungswert der empirischen Verteilung):

$$E(s_i) = \tilde{E}(s_i) \ (1 \leq i \leq m)$$

Gleichung 2.1 kann dann als begrenztes Optimierungsproblem umgeschrieben werden:

$$p^*(y \mid x) = \underset{p(y|x)}{argmax} H(y \mid x)$$

vorbehaltlich $E(s_i) = \tilde{E}(s_i)$ für alle$(1 \leq i \leq m)$

Die optimale Lösung aus der Arbeit von Roman Klinger u.a. [KT07] hat folgende Form:

$$p^*(y \mid x) = \frac{1}{Z(x)} exp\left(\sum_{i=1}^{m} \lambda_i s_i(x,y)\right) \tag{2.2}$$

wobei $Z(x)$ definiert ist als

$$Z(x) = \sum_{y \in Y} exp\left(\sum_{i=1}^{m} \lambda_i s_i(x,y)\right)$$

.

ME Modell für Sequenz-Tagging

Gleichung 2.2 bietet die Vorhersage für ein einzelnes Wort. Im Zusammenhang mit dem Sequenz-Tagging brauchen wir auch den Index der Position von Wörtern zu verfolgen. Zu diesem Zweck wird die Gleichung 2.2 umgeschrieben :

$$p^*(y_j \mid \overrightarrow{x}, j) = \frac{1}{Z(x)} exp\left(\sum_{i=1}^{m} \lambda_i s_i(y_j, \overrightarrow{x}, j)\right)$$

wobei j eine Stellungsanzeige eines Wortes in einer Sequenz ist. Dann kann die bedingte Wahrscheinlichkeit einer Sequenz von Label wie folgt formuliert werden

$$p(\overrightarrow{y} \mid \overrightarrow{x}) = \prod_{j=1}^{n} p(y_j \mid \overrightarrow{x}, j)$$

2.3.4. Conditional Random Fields

CRFs sind ungerichtete grafische Modelle [LMP01]. Sie werden oft verwendet, um die sequentiellen Daten zu taggen. Das bedeutet, wenn eine Sequenz X eingegeben wird, gibt eine Sequenz Y mit gleicher Länge aus.

Im speziellen Fall werden die Ausgangsknoten des graphischen Modells durch die Kanten in einer linearen Kette verknüpft [Lu+07]. In diesem Fall machen CRFs eine first-order Markov Unabhängigkeits-Annahme. Man versteht das als bedingt trainierter endlicher Automat. Die Aufgabe der NER kann als ein Problem von sequentiellem Tagging dargestellt werden.

Sei $X = x_1.....x_T$ einige beobachteten Eingabedatenfolgen, wie eine Folge von Wörtern im Text in einem Dokument (die Werte für n Eingangsknoten des grafischen Modells). Sei L eine Menge der Zustände von einem endlichen Automaten. Jeder Zustand steht für ein Label $l \in L$ (zum Beispiel Produktname, Lokation oder Zeit). $Y = y_1.....y_T$ entspricht der Sequenz von Zuständen in L. Lineare Ketten CRFs definieren die bedingte Wahrscheinlichkeit einer Sequenz des Zustands in einer Gleichung:

$$p_\lambda(Y \mid X) = \frac{1}{Z_\lambda(x)} exp\left(\sum_{t=1}^{T}\sum_{k}\lambda_k f_k(y_{t-1}, y_t, x, t)\right)$$

In der oberen Gleichung ist f_k eine beliebige Feature-Funktion über den Zustand y_t mit der Eingangssequenz x an Position t. Eine Feature-Funktion kann den Wert 0 (in den meisten Fällen) oder 1 haben.

λ_k ist ein erlerntes Gewicht für jede Feature-Funktion, die zum Beispiel durch Gradientenverfahren oder das Quasi-Newton-Verfahren gelöst werden, wie durch den LBFGS Algorithmus.

$Z_\lambda(x)$ ist ein Normierungsfaktor über alle möglichen Sequenzen von Labeln, wie es in der Gleichung 2.3 gezeigt wird:

$$Z_\lambda(x) = \sum_{y} exp\left(\sum_{t=1}^{T}\sum_{k}\lambda_k f_k(y_{t-1}, y_t, x, i)\right) \tag{2.3}$$

Der Prozess der Berechnung $Z_\lambda(x)$ beinhaltet die Sequenz des Zustands mit enormer Größe. Es gibt viele Algorithmen für die Einstellung des optimalen Gewichts. Sobald das optimale Gewicht gefunden wird, dann kann man die Sequenz des Labels mit der höchsten Wahrscheinlichkeit für eine unmarkierte Eingabesequenz bestimmen. Dafür können wir den Viterbi-Algorithmus verwenden [ZK06].

Es gibt schon viele Implementierungen für CRFs. Fujii Yasuhisa hat ein sehr beliebtes CRF Paket CRF++[4] für viele Plattformen entwickelt. Außerdem gibt es Mallet[5] Pakete , das CRF Pakete von Sunita Sarawagi [6] ,minorThird[7] und LingPipe[8] .

[4] http://crfpp.googlecode.com/
[5] http://mallet.cs.umass.edu/
[6] http://crf.sourceforge.net/
[7] http://sourceforge.net/apps/trac/minorthird/
[8] http://alias-i.com/lingpipe/

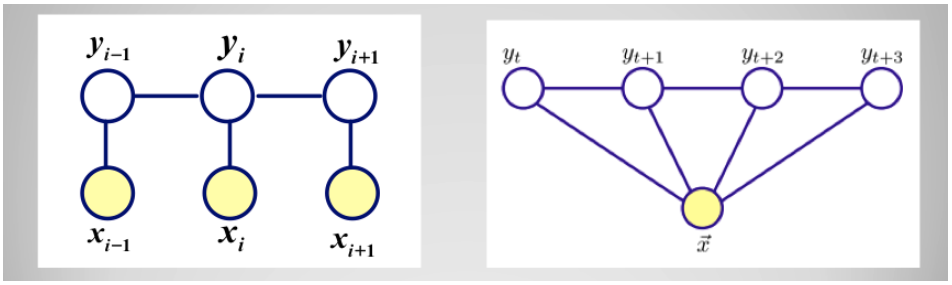

Abbildung 2.3.: Linear-Chain CRF

2.4. Evaluierung für NER

Die Evaluierung eines NER-Systems erfolgt in der Regel durch den Vergleich der Ergebnisse, die durch das NER-System automatisch ausgegeben werden, mit den Ergebnissen, die manuell durch menschliche Experten generiert werden. Diese Evaluierung kann auf der Ebene von Tokens oder Chunks durchgeführt werden.

Beim ersten Verfahren der Evaluierung wird ein Token nach dem anderen Token durchgeführt. D.h. jedes Token wird verifiziert (z.B. eine Lokation oder nicht). Beim zweiten Verfahren der Evaluierung wird ein Chunk nach dem anderen Chunk durchgeführt. D.h. ein Segment aus mehreren Token wird zusammen geprüft (z.B. derart geprüft, ob es ein zeitlicher Ausdruck ist).

Typischerweise verwendete Metriken für Evaluierung der NER die Korrektklassifikationsrate (Accuracy), die Genauigkeit (precision), die Trefferquote (recall) und das F-Maß (F-Measure).

Wir verwenden vier Notationen A, B, C, D für vier Fälle: Richtig Positiv (d.h. ein in einer Klasse durch System zugeordnetes Element gehört wirklich zu dieser Klasse), Falsch Positiv (d.h. ein in einer Klasse durch das System zugeordnetes Element gehört eigentlich nicht zu dieser Klasse), Falsch Negativ (d.h. ein Element, das wirklich zu einer Klasse gehört, ist aber nicht durch das System in diese Klasse zugeordnet worden) und Richtig Negativ (d.h. ein nicht in eine Klasse durch System zugeordnetes Element gehört wirklich nicht zu dieser Klasse). Die oberen vier Metriken können wie folgt definiert werden:

1. Korrektklassifikationsrate:

$$K = \frac{|A| + |D|}{|A + B + C + D|}$$

2. Die Genauigkeit P ist der Anteil der richtig zugeordneten Elemente bezüglich der Anzahl aller gefundenen Elemente einer Klasse:

$$P = \frac{|A|}{|A + B|}$$

3. Die Trefferquote R ist der Anteil der richtig zugeordneten Elemente bezüglich der Anzahl aller Elemente einer Klasse:

$$R = \frac{|A|}{|A + C|}$$

4. Das F-Maß ist das harmonische Mittel aus Genauigkeit und Trefferquote:

$$F = \frac{2 * (P * R)}{|P + R|}$$

2.5. PNER

Die Technologie der traditionellen NER ist derzeit relativ reif. Aber bis jetzt gibt es noch wenige Forschungen über PNER.

Pierre [Pie02] hat ein englisches NER-System entwickelt. Das System ermöglicht die Identifizierung der Produktnamen aus Produktreviews. Er verwendet einen einfachen „Boolean classifier“ für PNER, der ähnlich wie Token Matching ist. Basierend auf einem Constraint-Grammatik-Parser hat Bick [Bic04] für Dänisch ein System für PNER entwickelt. Dieser regelbasierte Ansatz hängt stark von der Leistung des dänischen Parsers ab. Chen Niu u.a. [Niu+03] haben in ihrer Arbeit die PNER im Bereich Auto, LKW, Flugzeug und Computer gemacht und haben ein F-Maß von 69,18% bekommen. Jun Zhao u.a.[ZL10] haben ein „hierarchical hidden Markov Model“ (kurz HHMM) für PNER verwendet. In ihrer Arbeit haben sie ein gutes Ergebnis bekommen. Aber ihr Verfahren ist nicht allgemein verwendbar. Deswegen haben, basierend auf CRFs, Zhang u.a. [ZG10] eine Methode für PNER vorgeschlagen. Für die Aufgabe TREC-2009[9] wird diese Methode verwendet und die Genauigkeit beträgt 93,6%. Die Trefferquote beträgt 92,4%.

Im Kapitel 3.2 werden wir CRFs verwenden, um die Aufgabe von PNER im Bereich Elektronik und Technik durchzuführen.

[9] http://trec.nist.gov/pubs/trec18/t18_proceedings.html

Kapitel 3.

Product Entity Named recognition im Bereich von Elektronik und Technik

Im Vergleich zur traditionellen NER hat PNER ihre eigenen Eigenschaften. Durch detaillierte Beobachtungen und Analysen werden die Eigenschaften der PNER in dieser Arbeit wie unten zusammengefasst.

1. Die Produktnamen aktualisieren und ändern sich schnell. Im Bereich Elektronik und Technik gibt es jeden Tag hunderte neue Produkte auf dem Markt. Zur Unterscheidung der verschiedenen Produkte geben die Unternehmen jedem Produkt einen Name. Der Modus der Benennung des Produktnamens kann entweder von der vorherigen Generation der Produkte erben oder neu entwickelt werden. Deswegen ist es eine neue Herausforderung, das bestehende System weiter an den neuen Modus der Benennung anzupassen und neue Produktnamen genaue zu erkennen.
2. Die Strukturformen der PNEs sind vielfältig. Im Vergleich mit traditionellem NE ist die Strukturform von PNE nicht einzig. Die Positionen der Komponenten im NE können relativ ausgetauscht werden. Zum Beispiel sind *„ Samsung S4 black edition“* und *„black Samsung S4“* das gleiche Produkt. Obwohl die Formen von *„Canon Rebel series digital camera T3i“* und *„Canon Rebel T3i digital camera“* unterschiedlich sind, beziehen sich die beiden auf das gleiche Produkt.
3. Der Produktname ist eine Mischung von Zahlen und Buchstaben. Zum Beispiel *„Nokia N900“*, *„Huawei Ascend P6“* und *„Creative M300“*. Diese Eigenschaft erhöht die Mehrdeutigkeit der internen Kategorie. Beispielweise kann „Apple“ ein Obst, eine Firma oder eine Produktmarke sein.
4. Der Produktname kann manchmal verkürzt geschrieben werden und manchmal hat ein Produkt mehrere Namen. Beispielweise schreiben die Benutzer *„Samsung Galaxy S4“* einfach mit *„S4“* oder *„Samsung Galaxy Note 2“* einfach mit *„Note 2“*. Diese Eigenschaft verursacht die Verstehensschwierigkeiten beim Lesen.
5. Die Grenzen von PNE sind unscharf. Im Vergleich zur traditionellen NER gibt es für PNER wenige Feature-Wörter im Kontext. Zum Beispiel *„discrict“* und *„sir“* sind gute Feature-Wörter, die traditionelle NER abzulösen. Aber für PNER mangelt es an solchen Feature-Wörtern. Deshalb ist das Auslösen der PNER schwieriger und die Bestimmung der Grenzen ist komplexer.

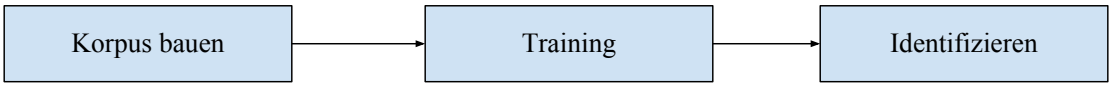

Abbildung 3.1.: Verlauf der PNER

Die regelbasierte Methode passt nicht für PNER in der oberen genannten Situationen. Wenn wir die regelbasierte Methode für PNER im Bereich Elektronik verwenden, brauchen wir für jeden Unterbereich von Elektronik eine separate Regel zu entwickeln. Deshalb ist die regelbasierte Methode für PNER nicht geeignet. In diesen Fällen verwenden wir die beliebte betreute Lernmethode des maschinellen Lernens, die auf statistischen Methoden basiert.

Statistische Systeme der NER können den Kontext und Wortart im Text ausnutzen. Durch die statistische Methode können viele NEs, die vorher noch nie in einem Bereich erschienen sind, korrekt identifiziert werden. Durch eine kleine Änderung kann das bestehende statistische System auch für andere Bereiche verwendet werden. Wir haben schon im Kapitel 2 erwähnt, dass die betreute Lernmethode einen großen Korpus braucht. Deshalb sammeln wir die Trainings-Texte aus dem Bereich von Elektronik und Technik. Dann markieren wir die PNEs manuell und bekommen einen Trainings-Korpus. Mit dem Trainings-Korpus wird das PNER-System durch ein statistisches Modell trainiert. Danach können wir das trainierte System wieder verwenden, um mehr PNEs zu erkennen. Obwohl diese betreute Lernmethode viele manuelle Arbeiten für die Filterung und Markierung der Trainings-Texte braucht, löst sie die Probleme, welche die regelbasierte Methode nicht lösen kann. Zum Beispiel das Problem der Erkennung von neuen PNEs, die mit neuer Struktur sind und vorher in einem Bereich noch nie erschienen sind. Außerdem hat diese betreute Lernmethode eine bessere Genauigkeit und Trefferquote.

In diesem Kapitel beschreiben wir ein Verfahren (siehe Abbildung 3.1) für PNER im Bereich Elektronik und Technik.

3.1. Der Bau eines Trainingskorpus für PNER

Für das auf betreuter Lernmethode basierende System von PNER brauchen wir einen großen Korpus, in dem die PNEs bereits markiert sind. Für den Bau eines solchen Korpus müssen wir zuerst PNE definieren.

3.1.1. Definition von Product Named Entity

Es ist schwierig, genau zu definieren, welche Arten von Ausdrücken als die Product Named Entities betrachtet werden können. Allgemein enthält ein PNE eine doppelte Bedeu-

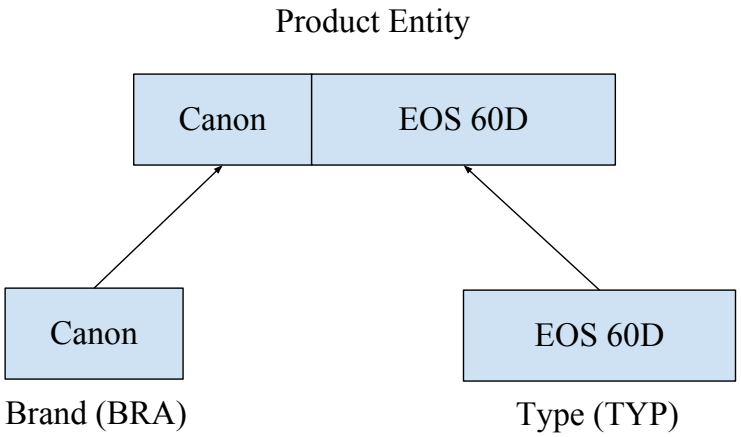

Abbildung 3.2.: Zwei Komponenten von PNE

tung[LMP01]. Einerseits muss es ein Ausdruck sein, der sich auf eine bestimmte Produktkategorie bezieht. Auf der anderen Seite muss es „Named Information“ geben.

Jun Zhao und Feifan Liu [ZL10] haben gemeint, dass das PNE normalerweise aus zwei Komponenten zusammengesetzt ist. Eine ist die Produktmarke, die andere ist der Produkttyp. (siehe Abbildung 3.2)

Obwohl es durch diese Aufteilung von Jun Zhao für die PNER einige Vorteile gibt, ist sie noch nicht für die Produkte geeignet, die im Bereich Elektronik und Technik sind. In dieser Arbeit unterteilen wir das PNE in drei Komponenten (siehe Abbildung 3.3). Die drei Komponenten sind Produktmarke, (kurz BRA) Produktserie (kurz SER) und Produkttyp (kurz TYP). Die Häufigkeiten der Aktualisierungen der drei Komponenten sind unterschiedlich im elektronischen Bereich. Das Aktualisieren der Produktmarke ist am langsamsten von den drei Komponenten. Auf dem zweiten Platz liegt die Produktserie. Der Produkttyp hat die maximale Aktualisierungsrate. Deswegen haben wir den Produkttyp von der Produktserie getrennt und sie werden separat erkannt.

Hier geben wir die Definitionen für Produktmarke, Produktserie, Produkttyp und die Definition für die Produkt Entity.

Die Produktmarke bezieht sich auf eine Klasse von Eigennamen (Brand von Produkten). Zum Beispiel „Nokia“, „HTC“ oder „Mido“. Produktserie ist ein allgemeiner Name von mehreren Produkten, die zur gleichen Produktmarke gehören. Beispielsweise in Produkte „Nikon Coolpix S9500“ und „Nikon Coolpix p100“ ist „Coolpix“ die Produktserie unter der Produktmarke „Nikon“. Produkttyp ist die Versionsinformation unter einer Produktserie. Normalerweise besteht der Produkttyp aus Buchstaben und Zahlen. Zum Beispiel für das Produkt „Samsung Galaxy s4“ ist „s4“ der Produkttyp. Dann definieren wir die Produkt Entity als ein Element, das aus Produktmarke, Produktserie und Produkttyp besteht. Aber in

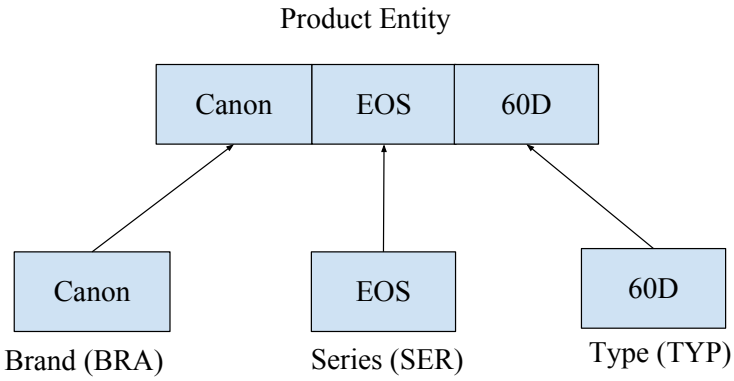

Abbildung 3.3.: Drei Komponenten von PNE

Text aus Internet	Produkt Entity	Produktmarke	Produktserie	Produkttyp
does anyone still use Nokia 3310?	Nokia 3310	Nokia	-	3310
is Canon EOS 60D good?	Canon EOS 60D	Canon	EOS	60D
how much costs Lenovo Thinkpad X230?	Lenovo Thinkpad X230	Lenovo	Thinkpad	x230

Tabelle 3.1.: Beispiele von PNEs

einem realen Text können einige Komponenten weggelassen werden. Deswegen vereinbaren wir einige Regelungen für PNE in einer Nominalphrase:

PNE enthält eine, zwei oder drei Produktmarken, Produktserie und Produkttyp. Zum Beispiel ist „Nokia Handy“ oder „ X230 Notebook“ eine PNE. Aber „Notebook Product“ ist keine PNE. Wir betrachten „Coolpix cameras“ als PNE, weil „Coolpix“ eine Produktserie unter der Produktmarke „Nikon“ ist. Aber „music handy“ ist keine PNE, da „music“ ist eine allgemeine Eigenschaft von fast allen Handys ist.

3.1.2. Bestimmen der Tags für den Trainingskorpus

Für das Training nutzen wir Tags im Trainingskorpus. Um die interne Struktur im PNE zu unterscheiden, definieren wir eine Menge von Tags für PNE in der Tabelle 3.2:

Um darüber hinaus zwischen verschiedenen Komponenten von PNE zu unterscheiden und später PNE zu erkennen, definieren wir die Menge Tags von NEs in Tabelle 3.3 :

Tags	Bedeutung
B	Dieses Wort ist der Beginn des entsprechenden NEs
I	Dieses Wort ist im entsprechenden NE intern
O	Dieses Wort ist kein NE

Tabelle 3.2.: Tags für PNE intern

Tags	Bedeutung
BRA	Produktmarke
SER	Produktserie
TYP	Produkttyp
PRO	Produkt

Tabelle 3.3.: Tags von NEs

Stellen wir die Tabelle 3.2 mit Tabelle 3.3 zusammen und bekommen wir $2 \times 4 + 1 = 9$ /acsNEs, wie in Tabelle 3.4 gezeigt wird:

Tags	Bedeutung
B-BRA	Dieses Wort ist der Beginn der PNE „Brand“
I-BRA	Dieses Wort ist in der PNE „Brand“ intern (kein Beginn)
B-SER	Dieses Wort ist der Beginn der PNE „Series“
I-SER	Dieses Wort ist in der PNE „Series“ intern (kein Beginn)
B-TYP	Dieses Wort ist der Beginn der PNE „Type“
I-TYP	Dieses Wort ist in der PNE „Type“ intern (kein Beginn)
B-PRO	Dieses Wort ist der Beginn der PNE „Product“
I-PRO	Dieses Wort ist in der PNE „Product“ intern (kein Beginn)
O	Dieses Wort ist keine PNE

Tabelle 3.4.: Tags für PNER

Wir verwenden die betreute Methode für PNER und möchten den Kontext und Wortarten im Text für die Erkennung ausnutzen. Deshalb verwenden die Tags für die Wortart wie Penn Treebank Projekt . Die Tags sind in Tabelle 3.5 und 3.6 gezeigt:

3.1.3. Regeln zum Markieren des Trainingskorpus

Zum Bau des Trainingskorpus markieren wir den Text nach den folgenden Grundsätzen:

1. Die markierte Informationseinheit von PNE sollte relativ konvergent sein. Sie sollte eine Klasse, eine Serie von elektronischen Produkten oder ein konkretes elektronisches Produkt unter einer Produktmarke bestimmen. Zum Beispiel *„Nokia N900“* ist ein PNE,

Tags	Bedeutung auf Englisch
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun
PRP$	Possessive pronoun (prolog version PRP-S)
RB	Adverb
RBR	Adverb, comparative

Tabelle 3.5.: Tags von Wortarten 1

da es ein spezifisches Handy bestimmt. *„Samsung Handy“* ist auch ein PNE, da es eine Serie von Produkten unter einer Produktmarke bezeichnet.

2. Das Markieren sollte die Integrität der Struktur des Syntaxbaums gewährleisten. Zum Beispiel bei diesem Markieren *„ How /WRB much /JJ is /VBZ the /DT in /IN Japan /NN produced /VBN [Leica /B-BRA D-LUX /B-SER 5 /B-TYP] /PRO . /.“* zerstört die grammatische Struktur diesen Satz, da *„in /IN Japan /NN produced /VBN“* und *„Leica /B-BRA D-LUX /B-SER 5 /B-TYP“* zusammen eine Einheit bilden. Um die Integrität der grammatischen Struktur zu gewährleisten, markieren wir *„in /IN Japan /NN produced /VBN Leica /B-BRA D-LUX /B-SER 5 /B-TYP“* als PNE. wie folgt: *„ How /WRB much /JJ is /VBZ the /DT [in /IN Japan /NN produced /VBN Leica /B-BRA D-LUX /B-SER 5 /B-TYP] /PRO . /.“*.

Im Folgenden geben wir die detaillierte Beschreibung des Markierens und einige spezifische Beispiele.

Tags	Bedeutung auf Englisch
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
PDT	Predeterminer
WP	Wh-pronoun
WP$	Possessive wh-pronoun (prolog version WP-S)
WRB	Wh-adverb
	VPound sign
’’	Close double quote
’	Close single quote
,	Comma
:	Colon, semi-colon
-RRB-	Right bracket
-LRB-	Left bracket
$	Dollar sign
„	Open double quote
,	Open single quote
.	Final punctuation

Tabelle 3.6.: Tags von Wortarten 2

1. Normalerweise erscheinen die Produktmarke, Produktserie oder Produkttyp geleichzeitig. In diesem Fall bilden sie zusammen ein PNE. Wir markieren nach den folgenden Regeln.

   ◇ Wenn die Produktmarke und der Produkttyp oder die Produktserie in Folge erscheinen, besteht das PNE aus ihr zusammen.

   Beispiel 1 *„The /DT original /JJ [Htc /B-BRA One /B-SER] /PRO works /VBZ with /IN HTC’s /BRA own /JJ Sense /NN user /NN interface /NN . /.“*

Beispiel 2 *„ The /DT new /JJ [iPhone /B-SER 4s /B-TYP] /PRO features /VBZ a /DT redesigned /VBN antenna /NN design /NN that /WDT allows /VBZ for /IN better /JJ reception /NN . /.“*

- ◇ Wenn es einige Wörter der Beschreibung von Produktattributen (z.B. Produktkategorie, Farbe) neben der Produktmarke, dem Produkttyp oder der Produktserie gibt, werden diese Wörter auch als eine Komponente von PNE gezählt.

  Beispiel 3 *„ Is /VBZ the /DT [Canon /B-BRA EOS /B-SER 60D /B-TYP Digital /NNP SLR /NNP Camera /NN] /PRO right /RB for /IN you /PRP ? /. “*

  Beispiel 4 *„The /DT [black /JJ iPhone /B-SER 5 /B-TYP] /PRO does /VBZ not /RB age /VB as /RB well /RB as /RB the /DT [white /JJ iPhone /B-SER 5 /B-TYP] /PRO . /.“*

2. Wenn es in einem Satz neben der Produktserie oder dem Produkttyp solche *„ .... series“* gibt, gehören diese *„..series“* auch zum PNE.

   Beispiel 5 *„It /PRP competes /VBZ primarily /RB with /IN the /DT [Nikon /B-BRA F /B-SER series /NN] /PRO and /CC ist /PR successors /NNS. /.“*

   Beispiel 6 *„The /DT [Canon /B-BRA EOS /B-SER 5D /B-TYP Series /NNP] /PRO group /NN is /VBZ dedicated /VBN to /TO one /CD of /IN the /DT finest /JJS digital /JJ SLR /NNP cameras /NNS made /VBD . /. “*

3. Wenn mehrere Produktkomponenten (BRA, TYP, SER) nebeneinander sind, markieren wir den Satz nach den folgenden Regeln.

   - ◇ Wenn zwei Produktkomponenten (BRA, TYP, SER) mit Konjunktionen verbunden sind, markieren wir sie getrennt als zwei PNEs.

     Beispiel 7 *„Nokia /B-BRA started /VBD rolling /VBG out /RP the /DT Portico /JJ update /NN for /IN the /DT [Lumia /B-SER 920 /B-TYP] /PRO and /CC [820 /B-TYP] /PRO . /.“*

   - ◇ Wenn mehrere Produktkomponenten (BRA, TYP, SER) nebeneinander sind und einige Komponenten (z.B. Produktmarke) weggelassen sind, markieren wir jedes PNEs getrennt.

     Beispiel 8 *„What /WP is /VBZ the /DT different /JJ between /IN [x100 /B-TYP] /PRO, [x-pro /B-TYP 1 /I-TYP] /PRO and [x10 /B-TYP] /PRO ? /. “*

4. Wenn die Produktekomponente (BRA, TYP, SER) allein erscheint, markieren wir den Satz nach den folgenden Regeln.

   - ◇ Die Produktmarke scheint allein und sie bezieht sich auf die Produkte unter dieser Marke. Dann kann sie als ein PNE markiert werden.

     Beispiel 9 *„I /PRP can /MD not /RB find /VB the /DT exact /JJ driver /NN for /IN your /PRP$ [Fujifilm /B-BRA Digital /NNP Cameras /NNPS] /PRO . /.“*

◇ Die Produktserie erscheint allein und sie bezieht sich auf eine Serie von Produkten unter einer Marke. Dann können wir sie als ein PNE markieren.

Beispiel 10 *„How /WRB often /RB you /PRP pick /VBP up /RP your /PRP$ [iPhone /B-SER] /PRO and /CC use /VB it /PRP to /TO perform /VB some /DT task /NN or /CC another /DT ? /.“*

◇ Der Produkttyp scheint allein und er kann ein bestimmtes Produkt bestimmen. Dann kann er als ein PNE markiert werden.

Beispiel 11 *„A /DT article /NN is /VBZ posted /VBN in /IN the /DT forum /NN indicating /VBG that /IN a /DT firmware /NN fix /NN for /IN the /DT [5D /B-TYP Mark /I-TYP III /I-TYP] /PRO is /VBZ being /VBG tested /VBN to /TO speed /VB up /RP autofocus /NN . /.“*

3.1.4. Der Bau des Korpus

Wir verwenden die halb-betreute Methode, um einen großen Trainingskorpus zu bauen. Das Verfahren ist in der Abbildung 3.4 dargestellt. Zuerst laden wir die entsprechenden Seiten aus dem Internet. Dann werden die Texte extrahiert und mit der Wortart gelabelt. Wir bekommen jetzt einen originalen Korpus. Dann taggen wir eine Menge von Texten aus dem originalen Korpus für das Klassifikator-Training. Durch diesen Klassifikator können mehrere Texte automatisch gelabelt werden. Dann können wir die durch den Klassifikator automatisch getaggten Texte noch mal manuell korrigieren, um einen standardisierten Korpus zu bauen. Dieser standardisierte Korpus wird wieder zum Klassifikator-Training verwendet werden. Diese Schleife läuft, bis wir einen großen standardisierten Trainings-Korpus bekommen.

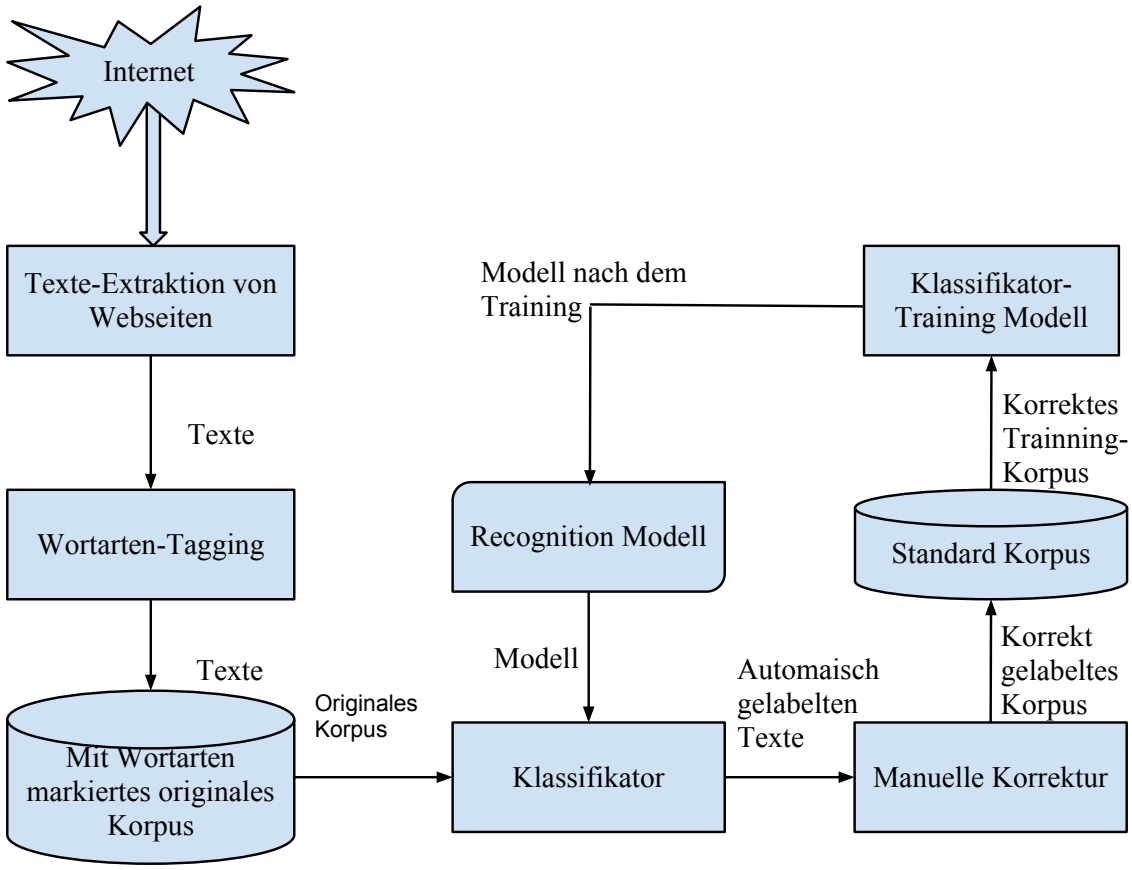

Abbildung 3.4.: Halb-betreute Methode für den Bau eines Trainings-Korpus

3.2. Product Named Entity Recognition

Im Kapitel 3.1 wird ein Trainings-Korpus gebaut. Jetzt müssen wir ein Modell für das Recognition-System auswählen und ein Verfahren für PNER bestimmen.

Vorher haben wir schon geschildert, dass wir die betreute Methode für PNER verwenden. Im Kapitel 2.3 werden drei Modelle (HMM, ME, CRFs) der betreuten Methoden vorgestellt. Alle drei Modelle können für das Sequenz-Tagging (z.B. NER oder Wortart-Tagging) verwendet werden. Aber das HMM hat einen großen Nachteil. Die geraden HMMs erster Ordnung können keine Abhängigkeiten zwischen weiter entfernten versteckten Zuständen berechnen. Deswegen können die Eigenschaften des Kontexts von Text nicht betrachtet werden. Die Auswahl der Feature-Funktionen ist begrenzt. Die Maximum Entropy Markov Modelle (kurz MEMM) können dieses Problem lösen. MEMM ist ein Modell, das die Eigenschaften von HMM und ME kombiniert. Das MEMM gestattet mehr Freiheit bei der Auswahl der Feauture-Funktion. Allerdings ist ein Nachteil, dass es beim MEMM „label bias problem“

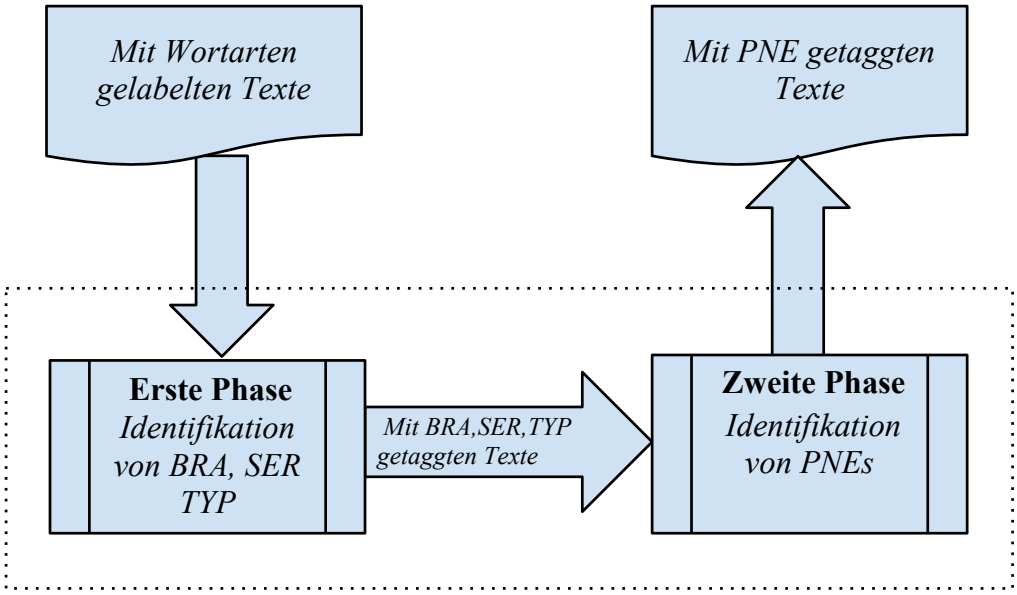

Abbildung 3.5.: Verfahren der PNER

gibt. d.h. Alle Situationen, die nicht im Trainings-Korpus erscheinen, werden alle ignoriert werden.

Das CRFs Modell kann alle oberen genannten Probleme lösen. Es kann das „label bias problem“ vermeiden [LMP01] und die beste Gesamtgenauigkeit erreichen. Das liegt daran, dass bei den CRFs nicht nur die Daten im jeweiligen Knoten berücksichtigt werden, sondern sind potentiell von allen Daten abhängig. Deshalb kann es den Kontext für die Recognition ausnutzen. Im Vergleich zum HMM hat das CRFs Modell dabei den Vorteil, dass es auch innerhalb des Modells überwacht trainiert werden kann, das heißt jedem Datentypen ist der korrekte Zustand zugeordnet. Das entsprechende HMM wird mit dem Baum-Welch-Algorithmus unüberwacht trainiert. Aus den obigen Gründen übertriff CRFs Modell die MEMM und HMM beide bei den Aufgaben des realen Sequenz-Taggings. [LMP01] [Pin+03] [SP03].

Im Kapitel 3.3 haben wir PNE in drei Komponenten (BRA, SER, TYP) geteilt. Deswegen unterteilen wir die PNER in zwei Phasen. In der ersten Phase werden die Komponenten (BRA, SER, TYP) von PNE identifiziert. Basierend auf der ersten Phase wird PNE in der zweiten Phase identifiziert. Die zwei Phasen sind in der Abbildung 3.5 dargestellt.

3.3. Extraktion der Restriktionen und Spezifikationen von Produkten

Nach der fertigen Identifizierung von Produktnamen werden die Restriktionen und Spezifikationen aus der Tabelle der Web-Seite extrahiert. Für die Identifizierung von Restriktionen und Spezifikationen machen wir zwei Annahmen:

1. Wenn in der ersten Zeile und in der ersten Spalte beide Produktnamen stehen, gehen ihre Treffpunkte um die Produktrestriktionen. (siehe ein Beispiel in Tabelle 3.7)

	Nikon D40 (Produktname)	Nikon D80 (Produktname)
Nikon AF-S NIKKOR 18-200mm objective (Produktname)	not suitable (Restriktion)	suitable (Restriktion)

Tabelle 3.7.: Produktnamen in der ersten Zeile und in der ersten Spalte

2. Wenn der Treffpunkt in einer Tabelle um Produktnamen geht, dann ist die Einheit in der gleichen Zeile oder Spalte des Treffpunkts eine Restriktion. Die anderen Einheiten können als Spezifikationen betrachtet werden (siehe ein Beispiel in Tabelle 3.8).

Features	
Megapixels (Spezifikation)	18 Megapixel (Spezifikation)
suitable lens Mount (Restriktion)	Canon EF Lens (Produktname)

Tabelle 3.8.: Produktnamen beim Treffpunkt

Kapitel 4.

Prototypische Implementierung

Im diesem Kapitel wird die Anwendung „productextraction“[1] beschrieben, welche die im Kapitel 3 beschriebene Methodik PNER prototypisch auf die Google App Engine[2] implementiert. Mit dieser Anwendung kann ein Benutzer eine englische Webseite (über Produkte im Bereich Elektronik und Technik) eingeben. Dann wird das Produktentwicklungswissen (z.B. Produktnamen, Produktrestriktionen oder Produktspezifikationen) aus dieser Webseite automatisch extrahiert und ausgegeben. Da es zu viele Unterkategorien und über Tausende verschiedene Produkte im Bereich Elektronik und Technik gibt, ist es schwierig, in einer kurzen Zeit eine Recognition-Anwendung für alle elektronischen Produkte zu entwickeln. Deshalb konzentrieren wir uns auf die Extraktion des Produktentwicklungswissens von Digitalkameras, die sich in einer Unterkategorie des Bereichs Elektronik und Technik befindet.

Diese Anwendung wird für eine englische Website entwickelt. Sie hat eine englische Benutzeroberfläche, um die Möglichkeit zur Nutzung der Anwendung nicht auf den Kreis der deutsch-sprachigen Benutzer zu beschränken.

4.1. Sammlung von Texten für den Trainings-Korpus

Abbildung 3.4 hat das Verfahren für den Bau des Trainings-Korpus dargestellt. Für das Training benötigen wir viele Texte über Produkte. Wir verwenden „yahoo shopping“ [3] Webseiten, welche die von Dritten zur Verfügung gestellten Informationen über die Preise, die Produkte, die Dienstleistungen und die Händler haben, als eine Quelle für die Sammlung des Korpus. (siehe Abbildung 4.5)

[1] `http://productextraction.appspot.com/`
[2] `https://developers.google.com/appengine/`
[3] `http://shopping.yahoo.com/`

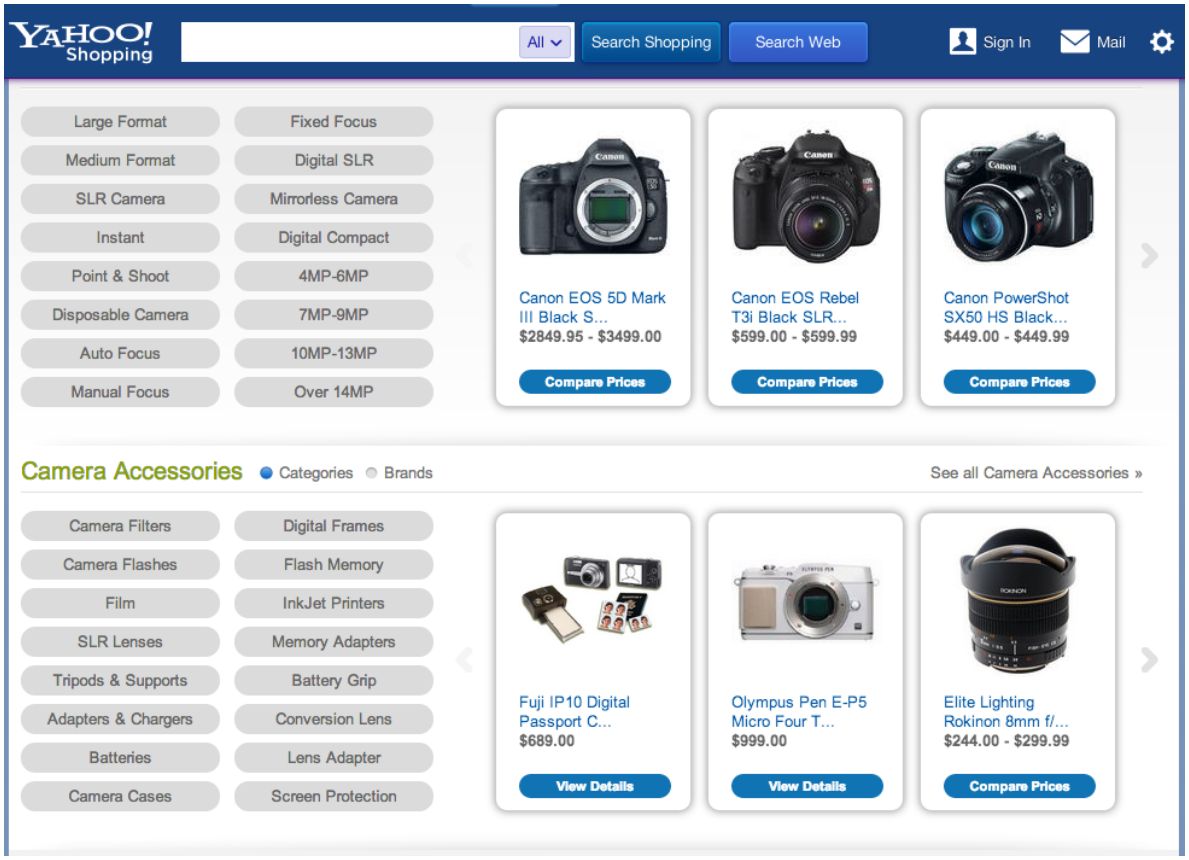

Abbildung 4.1.: Webseite von Yahoo Shopping

4.1.1. Auswahl eines geeigneten Tools für die Datenextraktion aus den Webseiten

Die meisten Webseiten sind HTML-Dokumente. Um die Texte über Produkte aus den HTML -Dokumenten für den Trainings-Korpus zu extrahieren, brauchen wir einige Tools. Für die Datenextraktion gibt es bis jetzt schon viele Produkte für die Endbenutzer. Es folgen einige aktuell vorhandene Tools.

WebSundew

WebSundew[4] ist ein Tool zur Datenextraktion. Mit Hilfe eines integrierten Browsers können die Daten aus den Webseiten effizient extrahiert werden. Es ist auch möglich,

[4] http://www.websundew.com/

den gesamten Prozess der Extraktion und die Speicherung der Ergebnisse mit WebSundew zu automatisieren. Die Ergebnisse können in einem beliebigen Format gespeichert werden.

Diese Software ist plattformabhängig und unterstützt nur das Windows System. Sie ist auch kostenpflichtig.

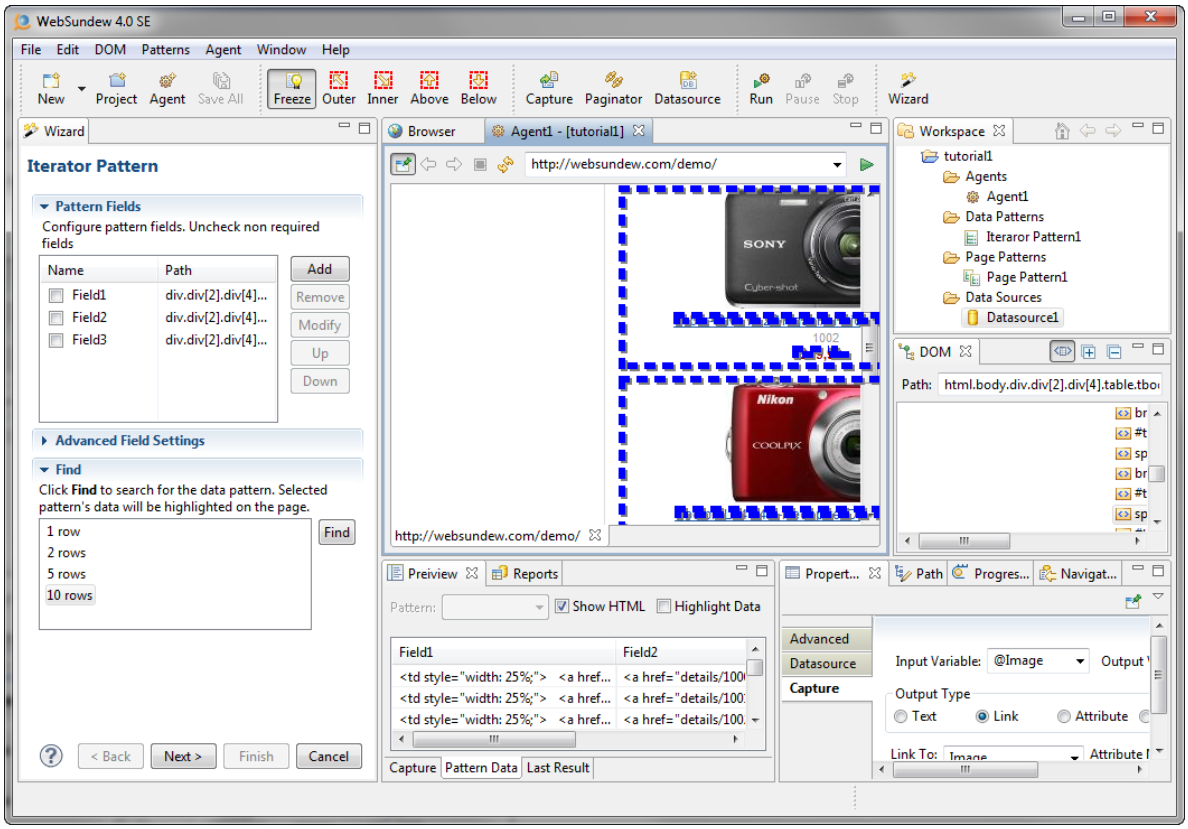

Abbildung 4.2.: WebSundewg

Der große Vorteil von WebSundew ist der integrierte Browser. Mit diesem Browser kann anhand von XPath-Ausdrücken der Benutzer die Datenbereiche definieren. Außerdem kann er im Browser sofort sehen, welche Informationen extrahiert werden (siehe Abbildung 4.2[5]). Aber mit dem Browser sind die Interaktionsmöglichkeiten eingeschränkt. Das ist ein Nachteil.

[5] `http://www.websundew.com/screenshots/`

Screen-Scraper

Das Tool Screen-Scraper[6] ist eine kostenpflichte Software. Die Basisversion ist kostenlos. Aber der Funktionsumfang ist eingeschränkt. Diese Software ist in Java geschrieben und ist plattformunabhängig. Sie bietet die Möglichkeit, mit anderer Software über eine Anzahl von unterstützten Programmiersprachen (z.B. .NET, Java, Ruby, PHP usw.) zu interagieren.

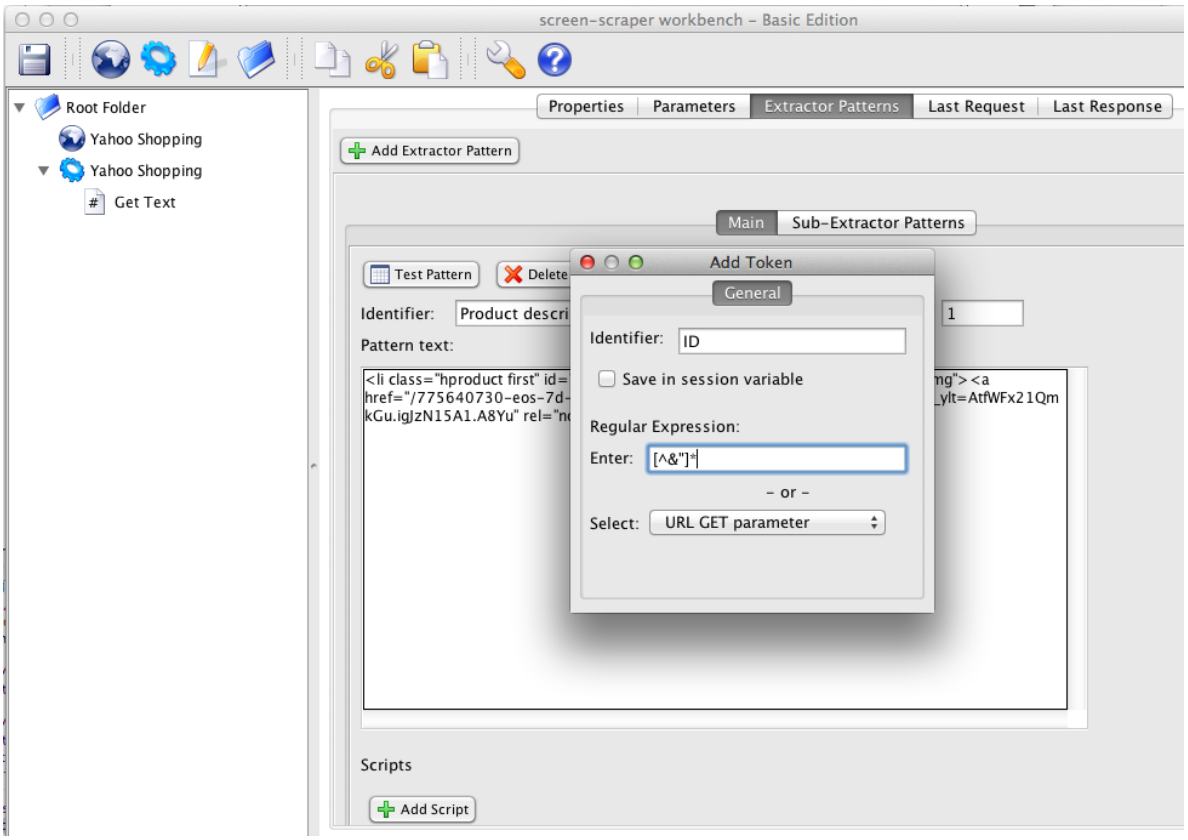

Abbildung 4.3.: ID Properties Set in Screen-Scraper

Durch die Einstellung von Skripten und Patterns können die Daten aus den Webseiten extrahiert werden. Die Skripte helfen, zur Extraktion zu den gewünschten Webseiten zu gelangen. Die Patterns dienen dazu, dass die Informationen gezielt extrahiert werden können (siehe Abbildung 4.3).

Der Vorteil von Screen-Scraper ist, dass er durch die Einstellung von XPath, Patterns und Zeitplan gute Konfigurationsmöglichkeiten für die Datenextraktion hat. Er hat

[6] `http://www.screen-scraper.com/`

allerdings einen Nachteil. Bei Screen-Scraper ist es nicht möglich, mit dem Browser zu interagieren.

Mozenda

Mozenda[7] ist eine kostenpflichte Software. Der Hauptunterschied der Software Mozenda zu der anderen Software zur Datenextraktion ist, dass ihre Extraktionsprojekte (Agenten) in der Cloud ausgeführt werden können. Zunächst bauen sie ein lokales Projekt mit einer Windows-Anwendung und dann führen Sie es auf dem Cloud-Server aus. Mit dieser verteilten Eigenschaft funktioniert Mozenda gut für den Bereich der großen geleichzeitigen Webextraktionen.

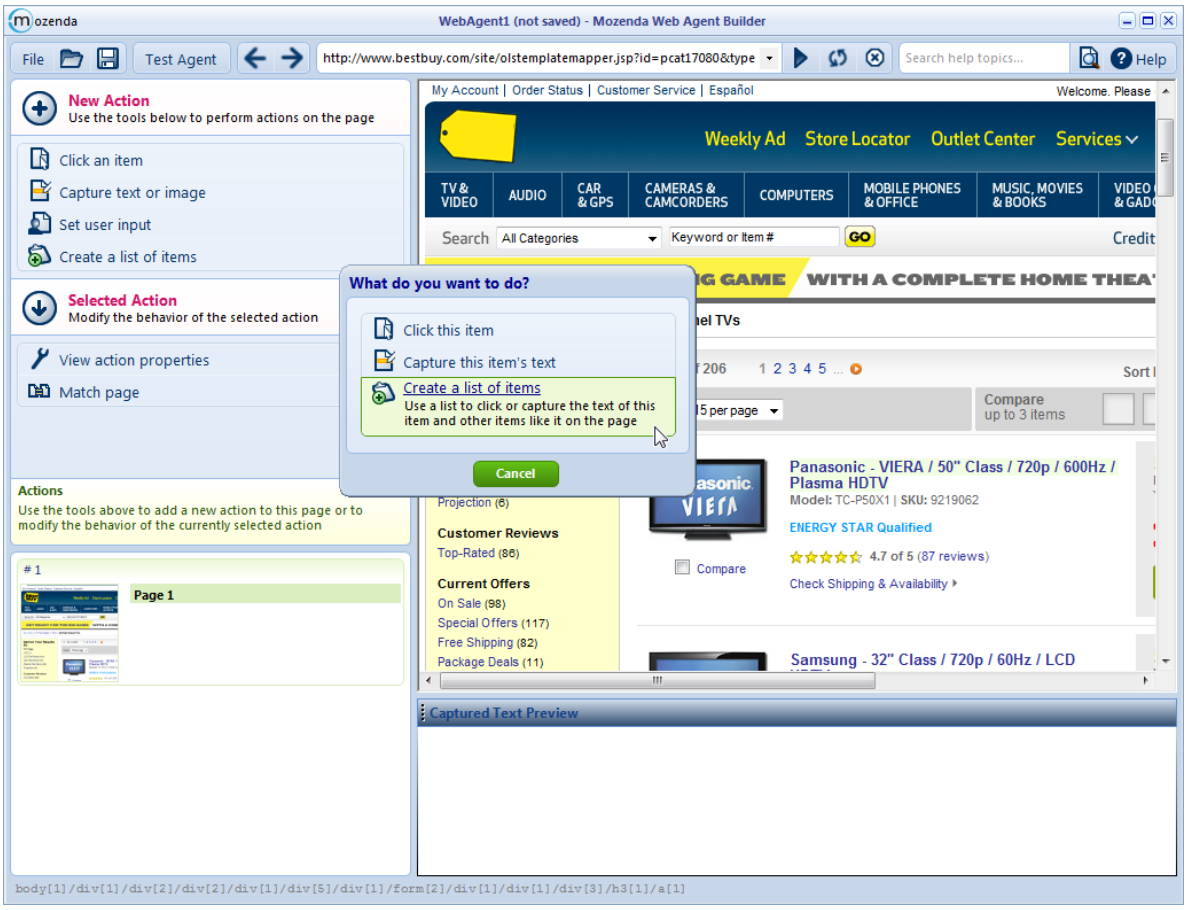

Abbildung 4.4.: Mozenda

[7] https://www.mozenda.com/

Die Vorteile von Mozenda sind der Zeitplan für die Ausführung der Extraktion, das einfache Bedienen durch die Interaktion mit dem Browser und der Export in viele Dateiformate (siehe Abbildung 4.4)[8].

Zu den Nachteilen von Mozenda zählt vor allem die eingeschränkte Navigation. Zum Beispiel wenn man eine Liste von Eingaben für ein Formular in Mozenda definieren möchte, bekommt man einige Schwierigkeiten. Außerdem ist die iterative Extraktion von Daten problematisch.

Web-Harvest

Web-Harvest [9] ist ein Open Source Tool für die Webdatenextraktion. Er ist in Java geschrieben und ist plattformunabhängig.

Web-Harvest konzentriert sich hauptsächlich auf HTML/XML-basierte Webseiten, die immer noch eine überwiegende Mehrheit der Webinhalte sind. Mit XSLT, XQuery und regulären Ausdrücken können die HTML- und XML-Dokumente in Web-Harvest verarbeitet werden.

Nach dem Erstellen der Konfiguration wird Web-Harvest in den folgenden Schritten ausgeführt werden(siehe Abbildung **??**[10]):

a) Ein Http Prozessor lädt die Inhalte aus der angegebenen URL herunter.

b) Mit dem HTML-to-XML-Prozessor werden HTML-Inhalte zu XML transformiert.

c) Eine interne Liste der mit XPath extrahierten XML-Knoten wird erstellt.

[8] `http://www.mozenda.com/tour04-click-capture`
[9] `http://web-harvest.sourceforge.net/`
[10] `http://web-harvest.sourceforge.net/overview.php`

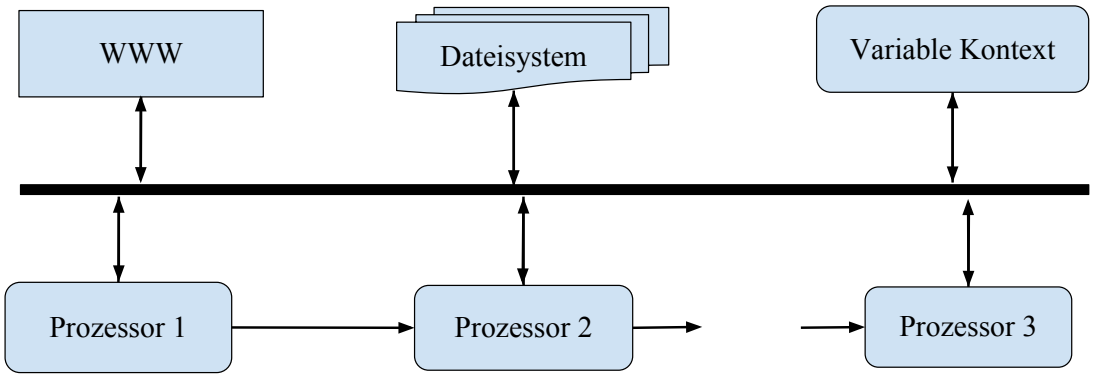

Abbildung 4.5.: Drei Phasen von Web-Harvest

Zu den Vorteilen von Web-Harvest zählen vor allem die klare Struktur der Konfiguration für die Datenextraktion sowie die Erweiterbarkeit und eine gute Dokumentation. Außerdem ist Web-Harvest kostenlos und plattformunabhängig. Deshalb nutzen wir ihn für die Extraktion der Produkttexte aus Yahoo Shopping.

4.1.2. Sammlung von Texten über Produkte aus den Webseiten

Für den Bau des Trainings-Korpus brauchen wir Texte über Produkte. Diese Texte sollten eine relativ vollständige grammatische Struktur besitzen und einige PNEs enthalten. Zum Beispiel die kurzen Beschreibungstexte über Kameraprodukte.

Abbildung 4.6[11] zeigt drei Beispiele über die Canon Kamera aus der Webseite von Yahoo Shopping:

[11] `http://shopping.yahoo.com/digital-cameras/canon--brand/`

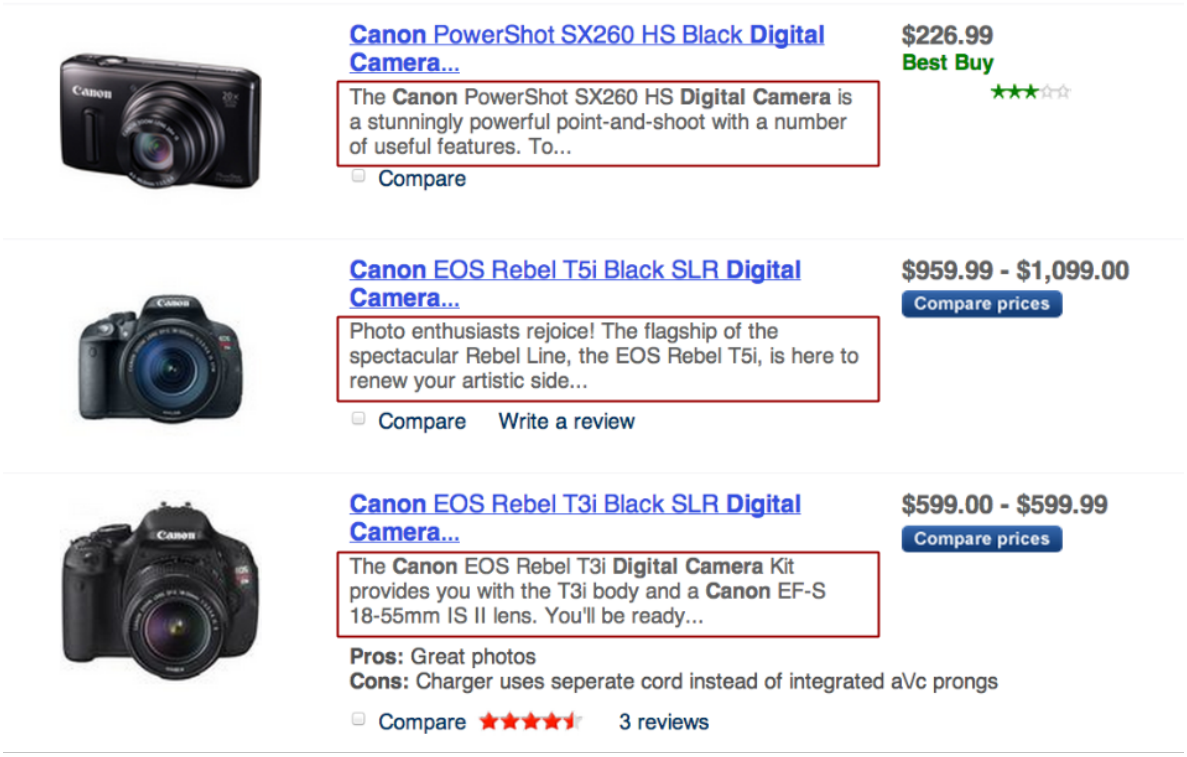

Abbildung 4.6.: Kurze Beschreibungstexte über Canon-Produkte auf Yahoo Shopping

Beschreibungstext 1 *„The Canon PowerShot SX260 HS Digital Camera is a stunningly powerfull point-and-shoot with a number of useful features.“*

Beschreibungstext 2 *„Photo enthusiasts rejoice! The flagship of the spectacular Rebel Line, the EOS Rebel T5i, is here to renew your artistic side.“*

Beschreibungstext 3 *„The Canon EOS Rebel T3i Digital Camera Kit provides you with the T3i body and a Canon EF-S 18-55mm IS II lens.“*

Wir nutzen Web-Harvest, um solche kurzen Beschreibungen von Kameraprodukten aus Yahoo Shopping zu extrahieren. Die Konfiguration von Web-Harvest für die Extraktion von Canon Kameras ist wie in Abbildung 4.7 dargestellt. Durch diese Einstellung werden zwei Seiten von insgesamt 30 kurzen Beschreibungen der Produkte von Canon gecrawlt. Wir wiederholen dieses Crawlen für die Kameramarke Nikon, Fujifilm, Olympus usw. Schließlich erhalten wir 461 Beschreibungstexte aus dem Bereich Kamera.

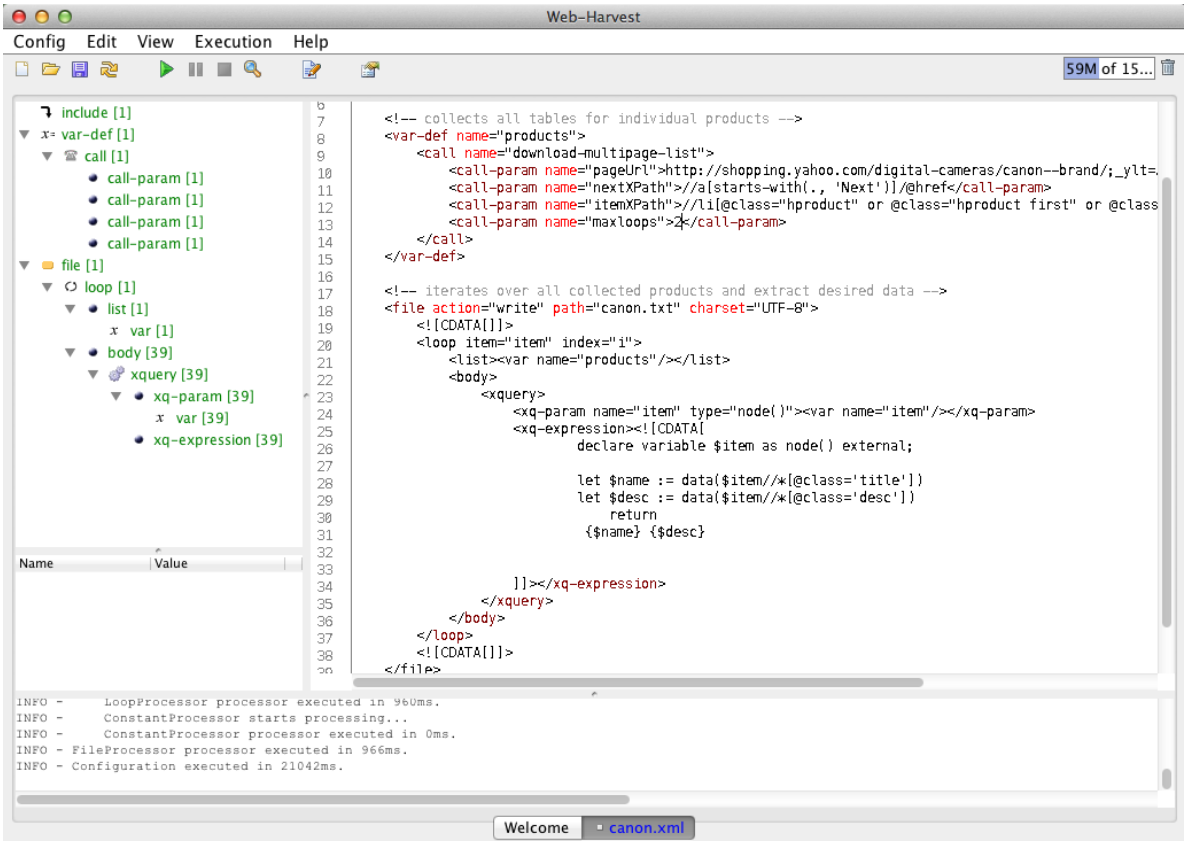

Abbildung 4.7.: Konfiguration von Web-Harvest

4.2. Taggen von Wortarten für den Trainings-Korpus

Die Beschreibungstexte im Bereich Kamera, die wir im Kapitel 4.1 aus Yahoo Shopping extrahiert haben, werden für den Trainings-Korpus mit Wortarten gelabelt. Wir können diese Arbeit manuell machen. Aber es ist sehr zeitaufwendig. Deshalb verwenden wir ein Tool „Stanford POS Tagger[12]“.

Stanford POS Tagger ist eine Software, die den Text in einer Sprache liest und jedem Wort eine Wortart (z.B. Substantiv, Verb, Adjektiv usw.) gibt. Die Software ist eine Java-Implementierung von „log-linear part-of-speech taggers“, die Kristina Toutanova u.a. [Tou+03] in ihrer Arbeit im Jahre 2003 beschrieben hat. Der Tagger verwendet das ME Model, um Sequenz-Tagging zu ermöglichen.

[12] `http://nlp.stanford.edu/software/tagger.shtml`

Der Tagger wurde ursprünglich von Kristina Toutanova entwickelt. Seither haben Dan Klein und andere Autoren die Leistung und die Benutzerfreundlichkeit des Taggers verbessert. Der Tagger wurde auch geändert, um andere Sprachen zu unterstützen.

Der Tagger ist unter der GNU General Public License (v2 oder höher) lizenziert. Das Paket umfasst die Komponenten für den Kommandozeilenaufruf, das Laufen als Server und ein Java API.

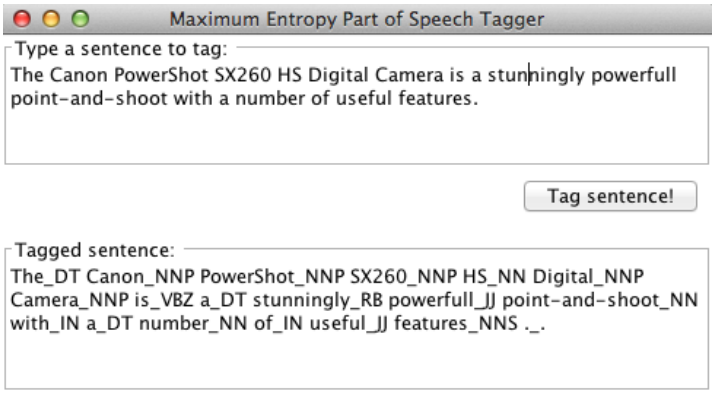

Abbildung 4.8.: Standford POS Tagger

Die Abbildung 4.8 stellt die grafische Benutzeroberfläche des Standford POS Tagger dar. Allerdings verwenden wir den Kommandozeilenaufruf (siehe Abbildung 4.9), um die Stapelverarbeitung von mehreren Sätzen in einem Text zu ermöglichen. Ein anderer Vorteil des Kommandozeilenaufrufs ist, dass wir mit ihm unsere eigenen Trennzeichen statt „“ in einer grafischen Benutzeroberfläche definieren können.

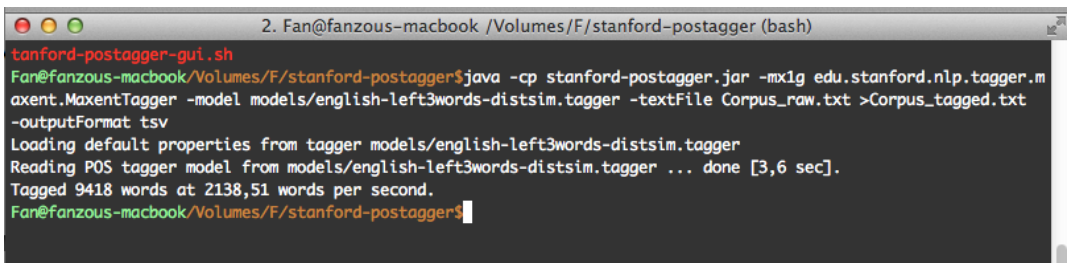

Abbildung 4.9.: Kommandozeilenaufruf von Standford POS Tagger

Wir nehmen den Beschreibungstext 1 als ein Beispiel:

„The Canon PowerShot SX260 HS Digital Camera is a stunningly powerfull point-and-shoot with a number of useful features.“.

Nach dem Tagging wird folgender Text mit den Tags ausgebeben:

The DT
Canon NNP
PowerShot NNP
SX260 NNP
HS NN
Digital NNP
Camera NNP
is VBZ
a DT
stunningly RB
powerfull JJ
point-and-shoot NN
with IN
a DT
number NN
of IN
useful JJ
features NNS
. .

Schließlich haben wir im Kapitel 4.1.2 461 Beschreibungstexte extrahiert (insgesamt 9418 Wörter) und mit dem Standford POS Tagger getaggt.

4.3. Taggen von PNEs für den Trainings-Korpus

Wir fügen jetzt manuell die Tags (B-BRA, I-BRA, B-SER, I-SER, B-TYP, I-TYP, B-PRO, I-PRO und O) für einige Trainingstexte (44 Beschreibungstexte) hinzu, die wir vorher mit Wortarten getaggt haben. Im Kapitel 3.2 wird PNER in zwei Phasen geteilt (siehe Abbildung 3.5). Wir taggen die Trainingstexte auch in zwei Phasen.

4.3.1. Taggen von Produktmarke, Produktserie und Produkttyp

Wir identifizieren und taggen manuell die Wörter mit der Produktmarke (B-BRA / I-BRA), der Produktserie (B-SER / I-SER), dem Produkttyp (B-TYP / I-TYP) oder Produkt (B-PRO / I-PRO) in den Trainingstexten. Andere Arten von Wörtern taggen wir mit „O“ (dieses Wort ist kein PNE).

Ein Beispiel.Beschreibungstexte nach dem Taggen der ersten Phase:

The DT O
Canon NNP B-BRA
PowerShot NNP B-SER
SX260 NNP B-TYP

HS NN I-TYP
Digital NNP B-PRO
Camera NNP I-PRO
is VBZ O
a DT O
stunningly RB O
powerfull JJ O
point-and-shoot NN O
with IN O
a DT O
number NN O
of IN O
useful JJ O
features NNS O
. . O

4.3.2. Taggen von Produkten

Basierend auf der ersten Phase nach der Regeln von Kapitel 3.1.3 taggen wir die Wörter mit B-PRO und I-PRO. Andere Arten von Wörtern taggen wir mit „O“.

Ein Beispiel: Beschreibungstexte nach dem Taggen in der zweiten Phase:

The DT O O
Canon NNP B-BRA B-PRO
PowerShot NNP B-SER I-PRO
SX260 NNP B-TYP I-PRO
HS NN I-TYP I-PRO
Digital NNP B-PRO I-PRO
Camera NNP I-PRO I-PRO
is VBZ O O
a DT O O
stunningly RB O O
powerfull JJ O O
point-and-shoot NN O O
with IN O O
a DT O O
number NN O O
of IN O O *useful* JJ O O
features NNS O O
. . O O

4.4. CRFs Training nach dem getaggten Trainings-Korpus

Als nächstes wird das Recognition-Modell für PNER trainiert. Im Kapitel 3.2 haben wir die Vorteile und Nachteile der drei Recognition-Modelle HMM, MEMM, und CRFs beschrieben und wir haben uns für das CRFs Modell für PNER entschieden.

4.4.1. Auswahl eines geeigneten CRF-Tools

Es gibt schon viele Tools von CRFs. Hier sind einige aktuell vorhandene CRFs-Tools. Hier werden drei CRFs-Tools verglichen.

Das CRFs-Paket von Sunita Sarawagi

Sunita Sarawagi hat ein CRFs-Paket[13] für das Sequenz-Tagging entwickelt. Das Paket ist eine Java-Implementierung, die die ungerichteten grafischen Modelle verwendet. Der Code folgt den Notationen und dem Algorithmus, die F. Sha und F. in ihrer Arbeit „Shallow parsing with conditional random fields“ [SP03] beschrieben haben. Das Paket wird in einer Weise gebaut, um die IE, die Segmentierung und Sequenz-Klassifizierung zu ermöglichen.

Die verschiedenen Verzeichnisse im Paket sind wie folgt:

- ◇ build/ : Speichert alle die kompilierten Java-Class-Dateien
- ◇ doc/ : Javadoc und die Dokumentation für das Paket
- ◇ samples/: Eine Probe von Datensatz und die entsprechende Konfigurationsdatei
- ◇ src/ : Java-Quelldateien
- ◇ lib/ : jar-Dateien

Der Code der Verteilung ist in verschiedenen Paketen organisiert. Der Quellcode kann im src / Verzeichnis der Distribution gefunden werden. Eine Zusammenfassung der verschiedenen Pakete ist unten angegeben.

- ◇ iitb.CRF: Kernpaket; enthält Durchführung der Trainings- und Inferenzalgorithmen und definiert verschiedene Schnittstellen, die von dem Benutzer der Verteilung umgesetzt werden können.
- ◇ iitb.Model: speichert die Umsetzung der verschiedenen Graphen, Funktionen und enthält den Feature-Generator.
- ◇ iitb.Utils: gemeinsame Klassen, die von anderen Paketen verwendet werden.

[13] `http://crf.sourceforge.net/`

- ◇ iitb.MaxentClassifier: eine Anwendung von CRFs zu einer ME-Klassifikations-Aufgabe.
- ◇ iitb.Segment: eine Anwendung von CRF zur einer Text-Segmentierungs-Aufgabe.

Dieses CRFs-Tool ist kostenlos und braucht J2SE 1.4[14] oder höher. Das Apache Ant-Paket wird auch gebraucht, um den Code zu kompilieren.

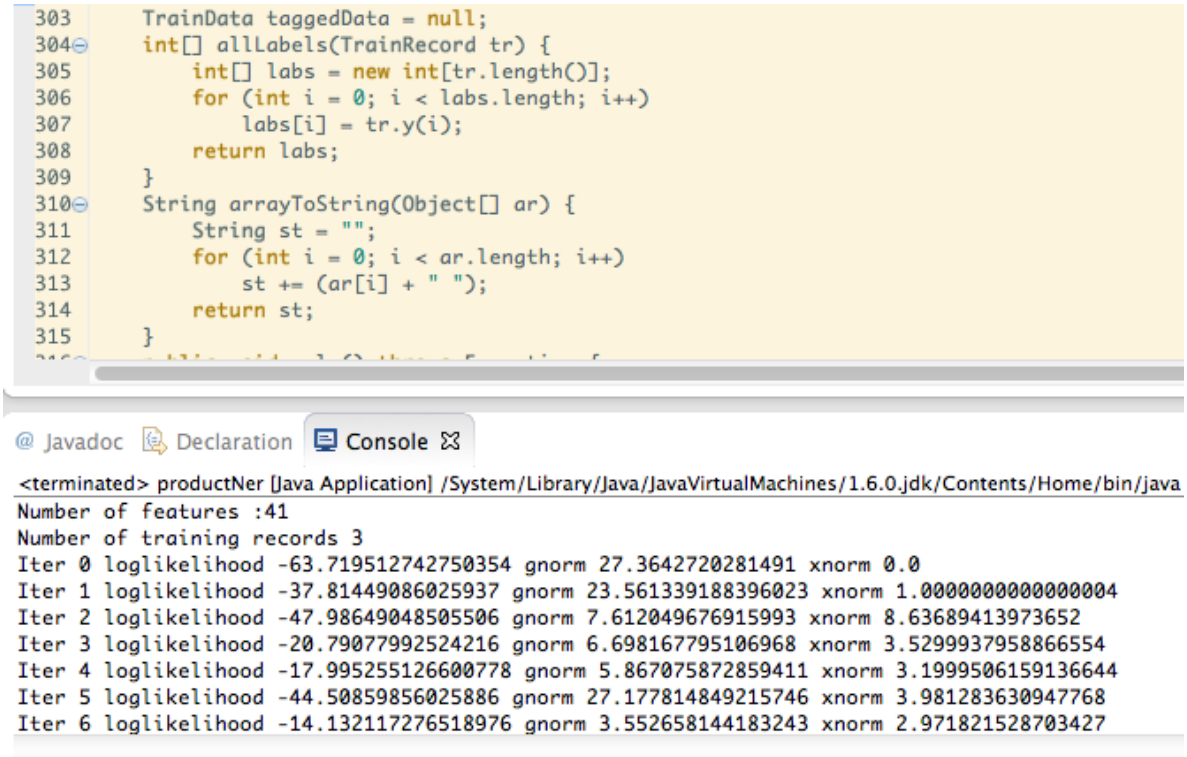

Abbildung 4.10.: Trainingstest mit CRFs-Paket von Sunita Sarawagi

MALLET

Machine Learning for Language Toolkit (kurz MALLET[15]) ist ein Java-basiertes Paket für die statistische Verarbeitung natürlicher Sprache, Dokumenten-Klassifikation, Extraktion von Informationen und andere Anwendungen des maschinellen Lernens zum Text.

Zusätzlich zur Klassifikation enthält MALLET Tools für Sequenz-Tagging für Anwendungen wie NER aus Text. Die Algorithmen, die bei MALLET für Sequenz-Tagging

[14] http://www.oracle.com/technetwork/java/javase/index.html
[15] http://mallet.cs.umass.edu/

verwendet wird, sind HMM, ME und CRFs. Diese Methoden werden in ein erweiterbares System für Finite State Transducers (kurz FST) implementiert.

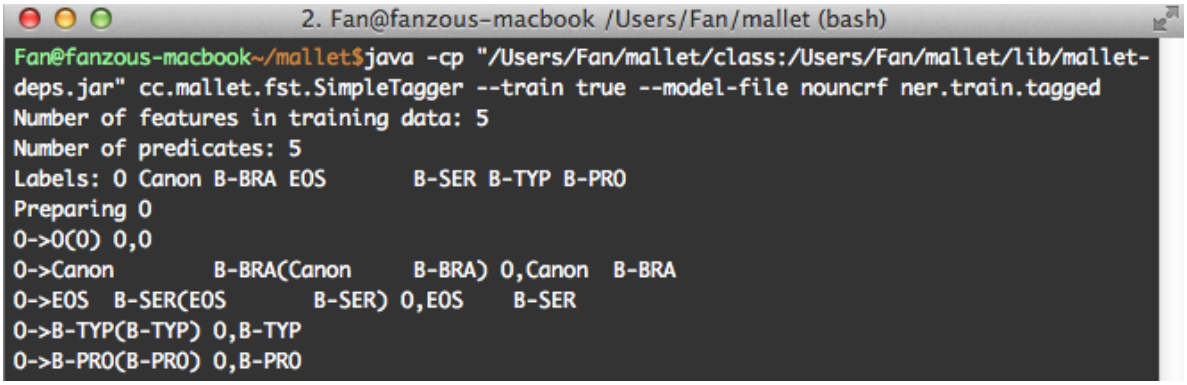

Abbildung 4.11.: Trainingstext mit MALLET

CRF++

CRF++[16] ist eine einfache, anpassbare Open-Source-Implementierung von CRFs zum Sequenz-Tagging für NER, Information Extraction und Text Chunking.

Die Eigenschaften von CRF++ sind:

- ◇ Kann Feature-Sets definieren
- ◇ Geschrieben in C + + mit STL
- ◇ Schnelles Training basierend auf LBFGS
- ◇ Weniger Speicherverbrauch beim Training und Testen
- ◇ Codierung / Decodierung in akzeptabler Zeit
- ◇ Kann n-best-Ausgänge führen
- ◇ Kann Single-beste MIRA-Trainings durchführen
- ◇ Kann marginale Wahrscheinlichkeiten für alle Kandidaten ausgeben
- ◇ Verfügbar als Open-Source-Software

[16] http://crfpp.googlecode.com/

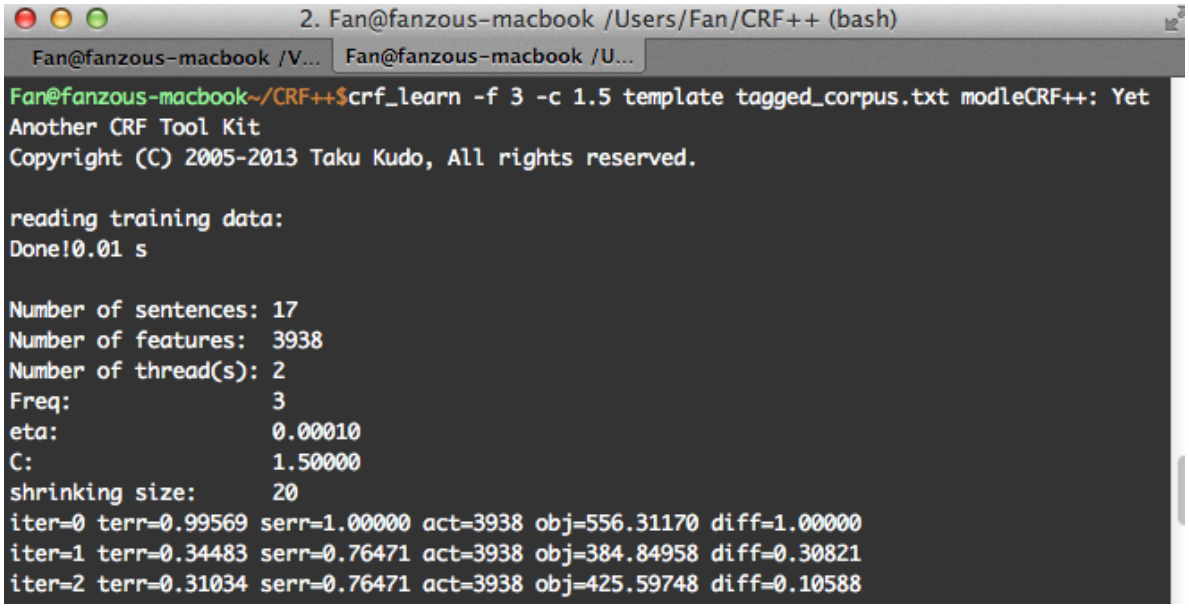

Abbildung 4.12.: Trainingstest mit CRF++

Vergleich der Tools

Die drei oben genannten Tools werden in Tabelle 4.1 noch einmal aufgeführt, um den Vergleich der drei Tools zu ermöglichen. Die Vorteile, die Nachteile und die Verfügbarkeit einer kostenlosen Version werden in dieser Tabelle dargestellt.

Tools	Lizenz	Verfügbarkeit	Vorteile	Nachteile
CRFs von Sunita Sarawagi	University of Illinois/NCSA Open Source License(NCSA)	kostenlos	ausführliche Dokumentationenmehrere Beispiele, plattformunabhängig und Erweiterbarkeit	relativ langsam, schwierige Definition von Features, großer Speicherverbrauch bei Training und Testen
MALLET	Common Public License, version 1.0 (CPL-1.0)	kostenlos	plattformunabhängig, Erweiterbarkeit und mächtig	kompliziert, sehr langsam, viel Speicherverbrauch beim Training und Testen, wenige Dokumentationen
CRF++	GNU Lesser General Public License (LGPL)	kostenlos	einfach zu nutzen, schnell, einfache Definition von Features, geringer Speicherverbrauch beim Training und Testen	plattformabhängig, geringe Erweiterbarkeit

Tabelle 4.1.: Eigenschaften der aktuell verfügbaren CRFs-Tools

Nach der Untersuchung wird CRF++ für die weitere Arbeit ausgewählt, da es beim Training und Testen wenig Speicher braucht und sehr schnell ist. Bei Implementierung sind diese Eigenschaften die großen Vorteile.

4.4.2. Training mit CRF++

Die Kontexte von PNEs spielen eine wichtige Rolle für die Verbesserung der Ergebnisse von PNER. CTF++ Pakete können sehr leicht die Beliebigen Features einer beobachtbaren Sequenz dem CRF Modell hinzugefügt werden, um die Abhängigkeitsinformationen im Kontext auszudrucken.

Dieser „Kontext“ bedeutet ein „Beobachtungsfenster“ $(\omega_{-n}, \omega_{-(n-1)}, ..., \omega_0, ..., \omega_{(n-1)}, \omega_n)$, das ein aktuelles Wort ω_0 und einige Wörter vor und nach dem aktuellen Wort enthält. Theoretisch je größer das „Beobachtungsfenster“ ist, desto mehr Kontextinformation stehen für die Recognition zur Verfügung. Wenn allerdings das Beobachtungsfenster zu groß ist, wird nicht nur die Performance der Recognition erheblich beeinträchtigt, sondern es erscheint auch das Overfitting (Überanpassung). Das Beobachtungsfenster kann auch nicht zu klein sein. Ansonsten können die Features nicht ausgenutzt werden. Dann gehen viele wichtige Informationen verloren. Nach der Analyse wählen wir eine Beobachtungsfenstergröße von 2. D.h. $(\omega_{-2}, \omega_{-1}, \omega_0, \omega_1\omega_2)$. Die Atomic Feature-Templates sind in Tabelle 4.2 (für BRA , SER, TYP Tagging und Recognition) und 4.3 (für PRO Tagging und Recognition) dargestellt.

Nr.	Atomic Feature-Templates	Bedeutung
U00	Cur_Char	aktuelles Wort
U01	Cur_Char_State	das Label des aktuellen Worts
U02	Cur_Char_FirstLeft	das erste Wort vor dem aktuellen Wort
U03	Cur_Char_FirstLeft_State	das Label des ersten Worts vor dem aktuellen Wort
U04	Cur_Char_FirstRight	das erste Wort nach dem aktuellen Wort
U05	Cur_Char_FirstRight_State	das Label des ersten Worts nach dem aktuellen Wort
U06	Cur_Char_SecondLeft	das zweite Wort vor dem aktuellen Wort
U07	Cur_Char_SecondLeft_State	das Label des zweiten Worts vor dem aktuellen Wort
U08	Cur_Char_SecondRight	das zweite Wort nach dem aktuellen Wort
U09	Cur_Char_SecondRight_State	das Label des zweiten Worts nach dem aktuellen Wort

Tabelle 4.2.: CRF Atomic Feature-Templates für PNER in der ersten Phase (für Produktkomponenten

Nr.	Atomic Feature-Templates	Bedeutung
U00	Cur_Char_State	das Label des aktuellen Worts
U01	Cur_Char_FirstLeft_State	das Label des ersten Worts vor dem aktuellen Wort
U02	Cur_Char_FirstRight_State	das Label des ersten Worts nach dem aktuellen Wort
U03	Cur_Char_SecondLeft_State	das Label des zweiten Worts vor dem aktuellen Wort
U04	Cur_Char_SecondRight_State	das Label des zweiten Worts nach dem aktuellen Wort

Tabelle 4.3.: CRF Atomic Feature-Templates für PNER in der zweiten Phase (für Produkte)

Die Templates in Tabelle 4.2 betrachten nur eine Beobachtungseinheit. Es reicht nicht aus, alle Phänomene im Kontext zu beschreiben. Deshalb werden einige neue komplexe Features durch die Kombination der oben genannten Atom-Features gebildet (siehe Tabelle 4.4 und 4.4).

Nr.	komplexe Templates
U10	Cur_Char & Cur_Char_FirstLeft
U11	Cur_Char & Cur_Char_FirstRight
U12	Cur_Char_State & Cur_Char_FirstLeft_State
U13	Cur_Char_State & Cur_Char_FirstRight_State
U14	Cur_Char_SecondLeft_State & Cur_Char_FirstLeft_State
U15	Cur_Char_SecondRight_State & Cur_Char_FirstRight_State
U16	Cur_Char_SecondLeft_State & Cur_Char_FirstLeft_State & Cur_Char_State
U17	Cur_Char_FirstLeft_State & Cur_Char_State & Cur_Char_FirstRight_State
U18	Cur_Char_ State & Cur_Char_FirstRight_State & Cur_Char_SecondRight_State

Tabelle 4.4.: CRF komplexe Feature-Templates für PNER in der ersten Phase (für Produktkomponenten)

Nr.	komplexe Templates
U05	Cur_Char_State & Cur_Char_FirstLeft_State
U06	Cur_Char_State & Cur_Char_FirstRight_State
U07	Cur_Char_SecondLeft_State & Cur_Char_FirstLeft_State & Cur_Char_State
U08	Cur_Char_FirstLeft_State & Cur_Char_State & Cur_Char_FirstRight_State
U09	Cur_Char_ State & Cur_Char_FirstRight_State & Cur_Char_SecondRight_State

Tabelle 4.5.: 5 CRF komplexe Feature-Templates für PNER in der zweiten Phase (für Produkte)

Die Konfigurationen von Feature-Templates in CRF++ sind wie folgt:

Template für PNER in der ersten Phase:

```
# Unigram


U00:%x[-2,0]
U01:%x[-1,0]
U02:%x[0,0]
U03:%x[1,0]
U04:%x[2,0]
U10:%x[-1,0]/%x[0,0]
U11:%x[0,0]/%x[1,0]
```

U05:%x[-2,1]

U06:%x[-1,1]

U07:%x[0,1]

U08:%x[1,1]

U09:%x[2,1]

U12:%x[-2,1]/%x[-1,1]

U13:%x[-1,1]/%x[0,1]

U14:%x[0,1]/%x[1,1]

U15:%x[1,1]/%x[2,1]

U16:%x[-2,1]/%x[-1,1]/%x[0,1]

U17:%x[-1,1]/%x[0,1]/%x[1,1]

U18:%x[0,1]/%x[1,1]/%x[2,1]

Template für PNER in der zweiten Phase:

Unigram

U00:%x[-2,2]

U01:%x[-1,2]

U02:%x[0,2]

U03:%x[1,2]

U04:%x[2,2]

U05:%x[-1,2]/%x[0,2]

U06:%x[0,2]/%x[1,2]

U07:%x[-2,2]/%x[-1,2]/%x[0,2]

U08:%x[-1,2]/%x[0,2]/%x[1,2]

U09:%x[0,2]/%x[1,2]/%x[2,2]

Mit den zwei Templates und dem getaggten Trainings-Korpus kann das CRFs Modell in CRF++ trainiert werden. Nach dem Training bekommen wir zwei CRFs-Modellfiles (für zwei Phasen), die wir später für die Recognition von PNEs verwenden können.

4.5. Implementierung des Recognition-Systems auf der Google App Engine

Die Anwendung „productextraction“ wird auf der Google App Engine implementiert. Da die Google App Engine keine C++ Programme unterstützt, verwenden wir noch einen extra Linux Server, auf dem CRF++ laufen kann.

4.5.1. Architektur der Anwendung

Die Implementierung der Anwendung „productextraction“ folgt dem Architekturmuster Model-View-Controller 2 Modell (MVC 2), das eine für Webanwendungen optimierte Variante des Model-View-Controllers (kurz MVC) Modells ist. Durch den Einsatz dieses Architekturmusters kann die Logik von der Benutzeroberfläche einer Anwendung getrennt werden.

Abbildung 4.13 stellt die Architektur der Anwendung „productextraction“ dar. Das ganze „productextraction“ System ist in zwei Blöcke geteilt. Ein Teil ist in Java geschrieben und läuft auf der Google App Engine. In diesem Teil gibt es Funktionen für das Validierungsprüfen, das Webcrawlen, das Textverarbeiten, das Tagging von Wortarten und das Markieren von Endergebnissen. Der zweite Teil läuft auf einem Linux-Server. Auf dem Server wird die Software von Apache[17] , PHP[18] und CRF++ installiert. Diese Anwendungen sind für die Identifizierung der PNEs verantwortlich.

[17] http://www.apache.org/
[18] http://php.net/

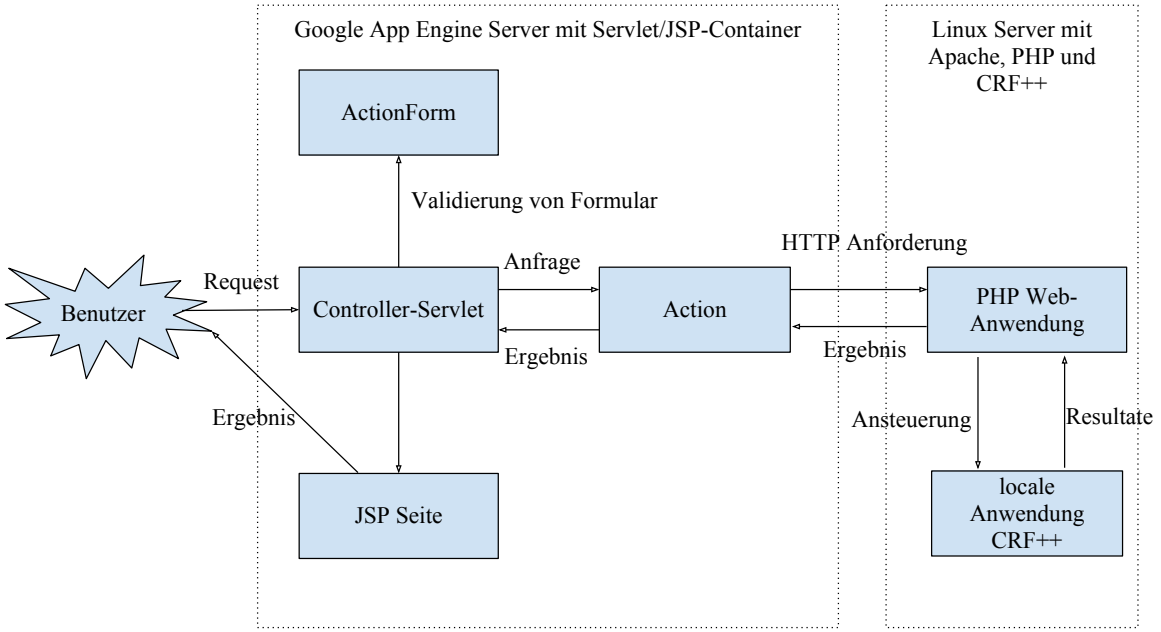

Abbildung 4.13.: Architektur des PNER-Systems „productextraction“

Das gesamte Verfahren der Anwendung ist wie folgt.

a) Der Benutzer gibt eine URL von einer Webseite ein, aus der er die Namen, die Spezifikationen und die Restriktionen von Produkten extrahieren möchte.

b) Der Browser überprüft die Gültigkeit der URL, gibt Fehlermeldungen aus oder leitet die Anfrage an Programme auf GAE weiter.

c) Die Crawler-Funktion GAE crawlt die entsprechende Webseite und bekommt ein HTML-Dokument.

d) Das HTML-Dokument wird zu einem reinen TEXT-Dokument konvertiert (auf GAE).

e) Das TEXT-Dokument wird segmentiert und mit Wortarten getaggt (auf GAE).

f) Der getaggte Text wird zum externen Linux-Server gesendet und dort weiter verarbeitet.

g) Das PHP-Programm auf dem Linux-Server behandelt die Anfrage und ruft das CRF++ Programm auf, das auf dem Linux-Server läuft.

h) CRF++ identifiziert zuerst die Komponente von PNEs (BRA, TYP, SER). Basierend auf dem Ergebnis werden PNEs durch CRF++ weiter identifiziert und getaggt. Dieser getaggte Text wird auf dem Linux-Server gespeichert.

i) Das PHP-Programm sendet den getaggten Text an GAE zurück.

j) Die Extraktionsfunktion auf GAE extrahiert die PNEs, ihre Spezifikationen und ihre Restriktionen nach dem getaggten Text.

k) Die Ergebnisse werden für den Benutzer markiert und anschaulich dargestellt.

Abbildung 4.14 zeigt das gesamte Sequenzdiagramm der Anwendung „productextraction“.

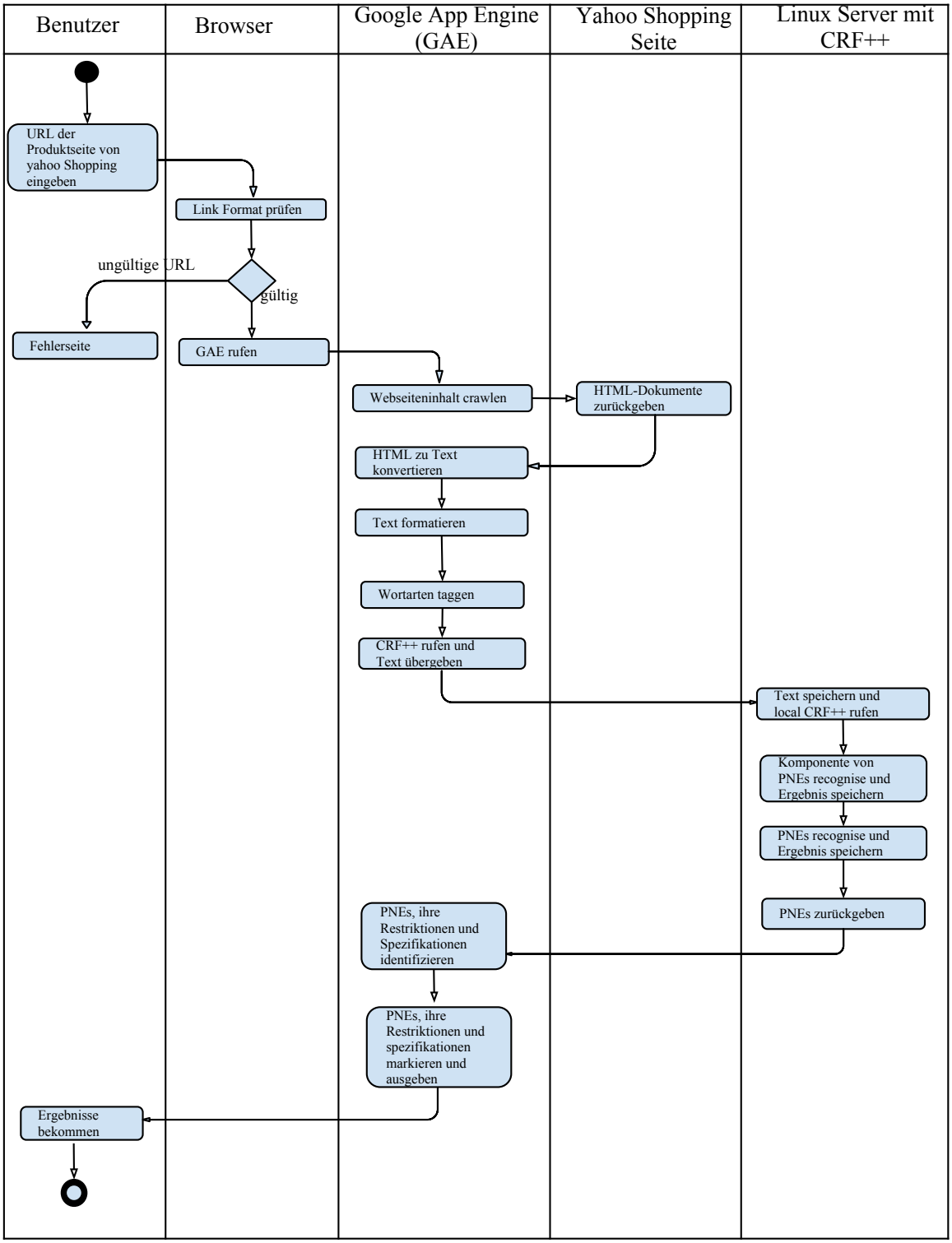

Abbildung 4.14.: Sequenzdiagramm des PNER-Systems „productextraction“

4.5.2. Funktionen der Anwendung

Die Beziehung zwischen der Java-Klasse „ProductExtractionServlet" und anderen Java-Klassen wird in Abbildung 4.15 gezeigt.

In der Klasse ProductExtractionServlet wird statt doGet()die doPost() Methode verwendet, damit werden die Daten der URL nicht angegeben. Die Funktion doPost() nimmt die Daten des entsprechenden HTML-Formulars an und gibt die Daten an andere Funktionen weiter.

Die anderen Klassen, die von der Klasse ProductExtractionServlet aufgerufen werden:

a) Die Klasse getLinkContent crawlt die entsprechende Webseite mit der angegebenen URL. Nach dem Crawlen wird das HTML-Dokument dieser Webseite durch diese Klasse zu reinem Text verwandelt. Nur der Text zwischen den HTML-Tags „<body>" und „</body>" wird extrahiert und verarbeitet. Einige Zeichen im Text werden ersetzt (z.b. wird „_" durch „ " ersetzt), um das künftige Tagging von Wortarten anzupassen. Nach der Verarbeitung wird der Text an die Klasse ProductExtractionServlet zurückgegeben.

b) Die Klasse postTagger verwendet Stanford POS Tagger Java-Paket. Sie ruft die MaxentTagger Funktion auf, die das „english-left3words-distsim.tagger" Modell verwendet. Durch den Tagger wird der Text mit Wordarten gelabelt.

c) Die Klasse postData sendet den getaggten Text zu einem anderen PHP Programm auf einem externen Linux-Server und löst die Identifizierung aus. Die Ergebnisse werden von ProductExtractionServlet geholt.

d) Die Klasse Product beschreibt das Objekt von Produkt und bietet zwei Methoden, um auf die Produkte im Text zuzugreifen (getProduct()) oder die Menge von Produkten im Text zu berechnen (geProductQty()).

e) Die Klasse markProduct bietet eine Methode, um die Produktnamen, die Spezifikationen und die Restriktionen von Produkten zu markieren und mit der originalen Seite zu vergleichen.

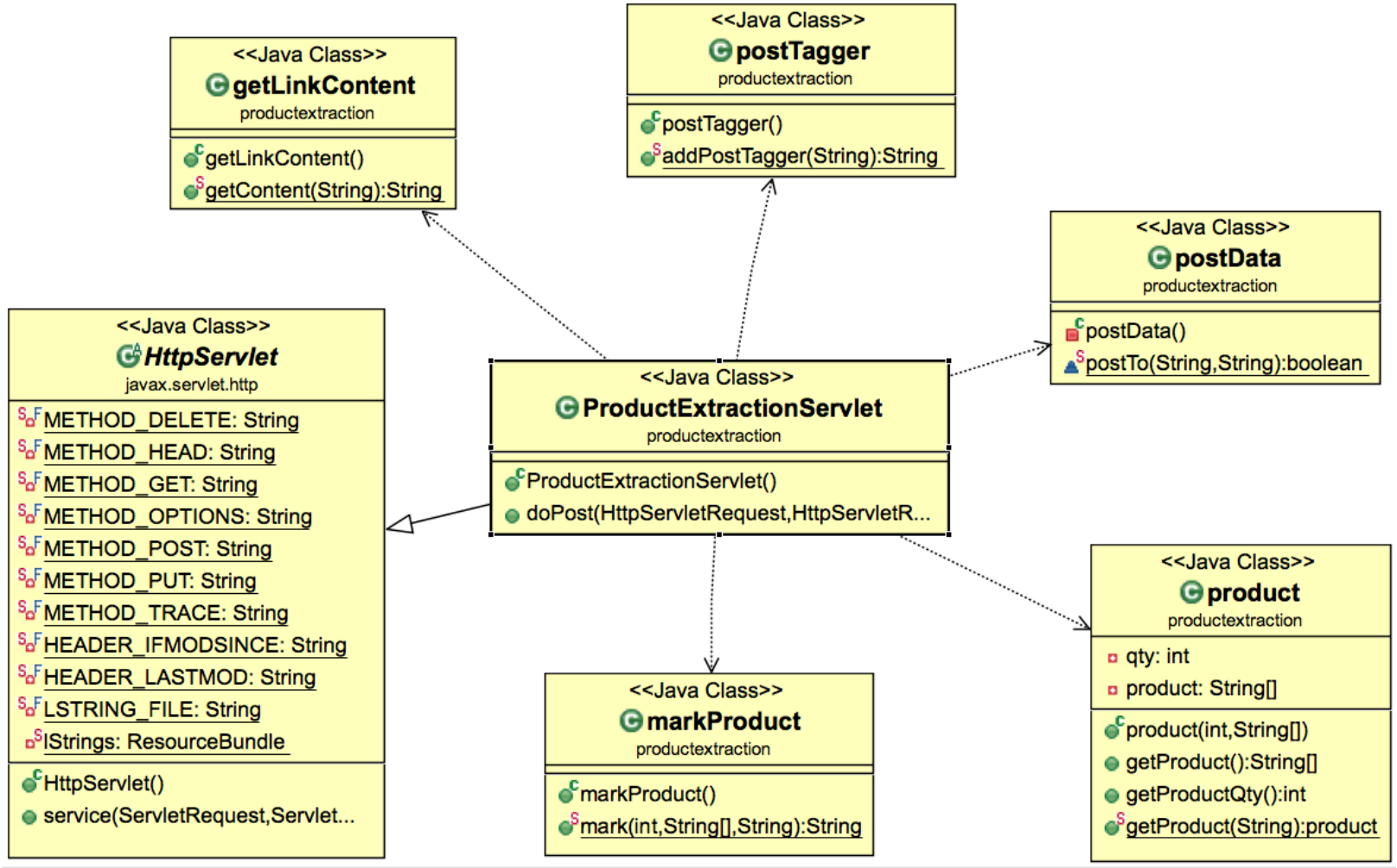

Abbildung 4.15.: Klassendiagramm von „productExtraction“

Außerdem gibt es auf dem GAE-Server noch die getUrl.jsp und extract.jsp Seiten. Die getUrl.jsp Seite ist die Startseite der Anwendung. Sie nimmt die URL durch das Formular an und überprüft die Gültigkeit der URL mit regulären Ausdrücken. Die exract.jsp ist verantwortlich für die Darstellung der Ergebnisse. Die Liste von Dateien in Eclipse für GAE siehe Abbildung 4.16.

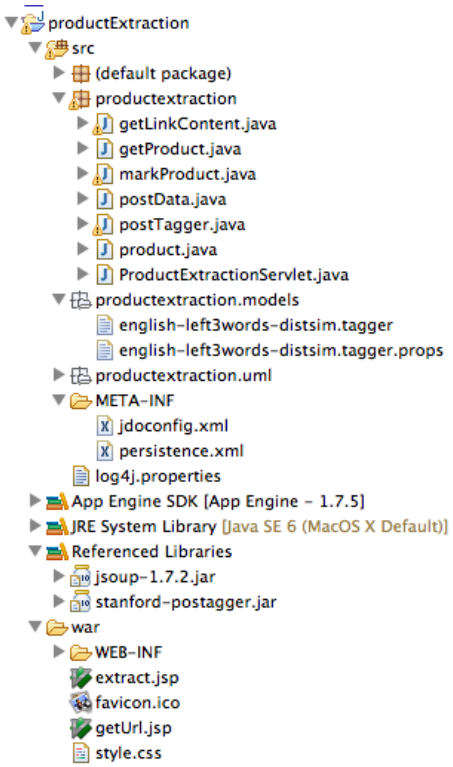

Abbildung 4.16.: Dateiliste von „productExtraction“ im Eclipse für GAE

Abbildung 4.17 zeigt die Dateiliste Dateiliste von „productExtraction“ auf dem Linux-Server an. Das crfprocess.php ist eine „Kupplung“ zwischen Java-Programmen auf dem GAE-Server und dem CRF++ Tool auf dem externen Linux-Server. Die Datei „tag-gedData.txt“ ist eine mit Wortarten getaggte Text-Datei, die durch CRF++ identifiziert werden wird. Nach der ersten Phase der Identifizierung wird cecognisedBrand.txt generiert. Dieser Text wird wieder mit PNEs getaggt. Dann bekommen wir eine Datei „recognisedData.txt“.

```
root@li280-89:/home/wwwroot/dogerman.de# ls -la
total 196
drwxr-xr-x  3 www www    4096 Aug  8 19:59 .
drwxr-xr-x 21 www root   4096 Jan  8  2013 ..
drwxr-xr-x  2 www www    4096 Aug  8 19:59 CRF++
-rw-r--r--  1 www www      20 Aug  3 14:39 crf.log.txt
-rw-r--r--  1 www www    1342 Jun 11 02:56 crfprocess.php
-rw-r--r--  1 www www  110488 Jun 25 15:01 model_file.brand
-rw-r--r--  1 www www    9676 Jun 11 02:55 model_file.product
-rw-r--r--  1 www www   16358 Aug  3 14:39 recognisedBrand.txt
-rw-r--r--  1 www www   19916 Aug  3 14:39 recognisedData.txt
-rw-r--r--  1 www www   12799 Aug  3 14:39 taggedData.txt
```

Abbildung 4.17.: Dateiliste von „productExtraction“ auf dem Linux-Server

4.5.3. Benutzeroberfläche

Der Screenshot der Startseite wird in Abbildung 4.18 angezeigt. In die Textbox ist schon eine URL durch den Benutzer eingegeben.

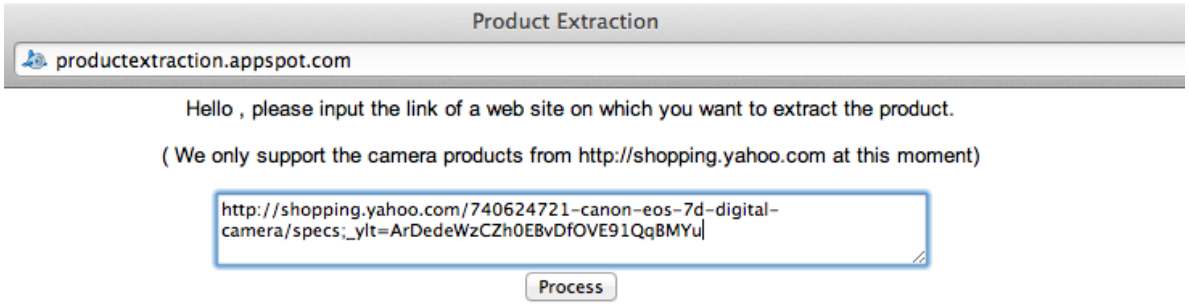

Abbildung 4.18.: Screenshot der Startseite

Abbildung 4.19 zeigt die Fehlermeldung, wenn der Benutzer eine ungültige URL eingegeben hat.

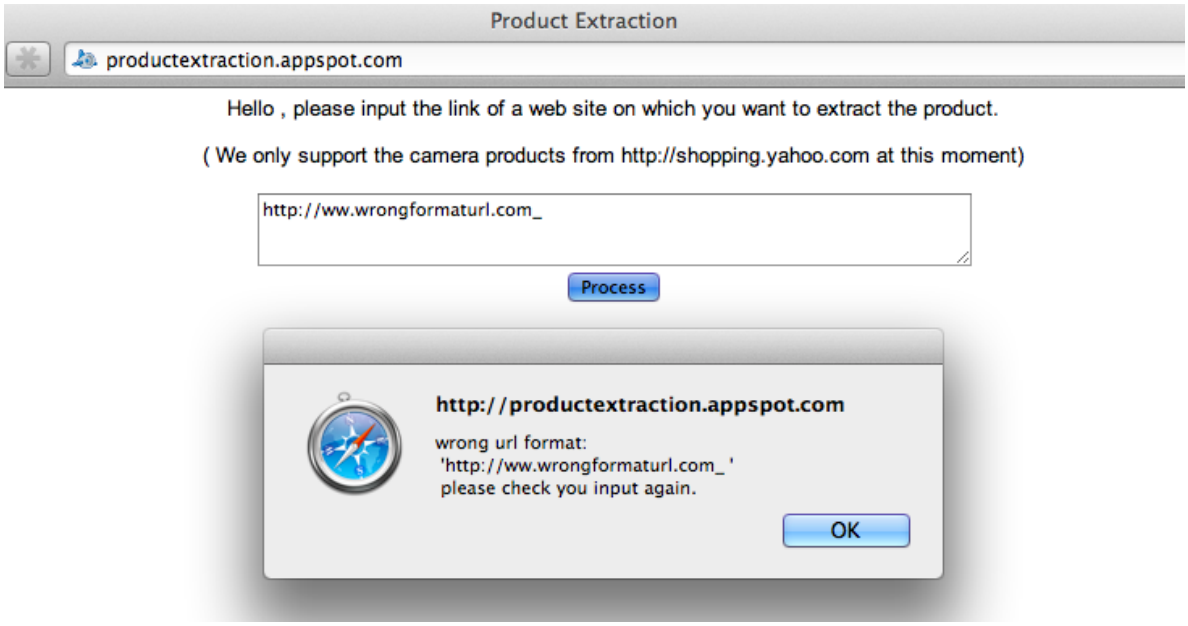

Abbildung 4.19.: Fehlermeldung bei ungültiger URL

Abbildung4.20 gibt ist ein Screenshot der markierten Ergebnisse. Auf dieser Webseite gibt es zwei Spalten. In der linken Spalte stehen die Ergebnisse der Identifizierung. Produktnamen werden in blau markiert. Grün steht für die Spezifikationen von Produkten. Die Farbe der Restriktionen des Produkts ist rot. In der rechten Spalte wird die ursprüngliche Seite gezeigt.

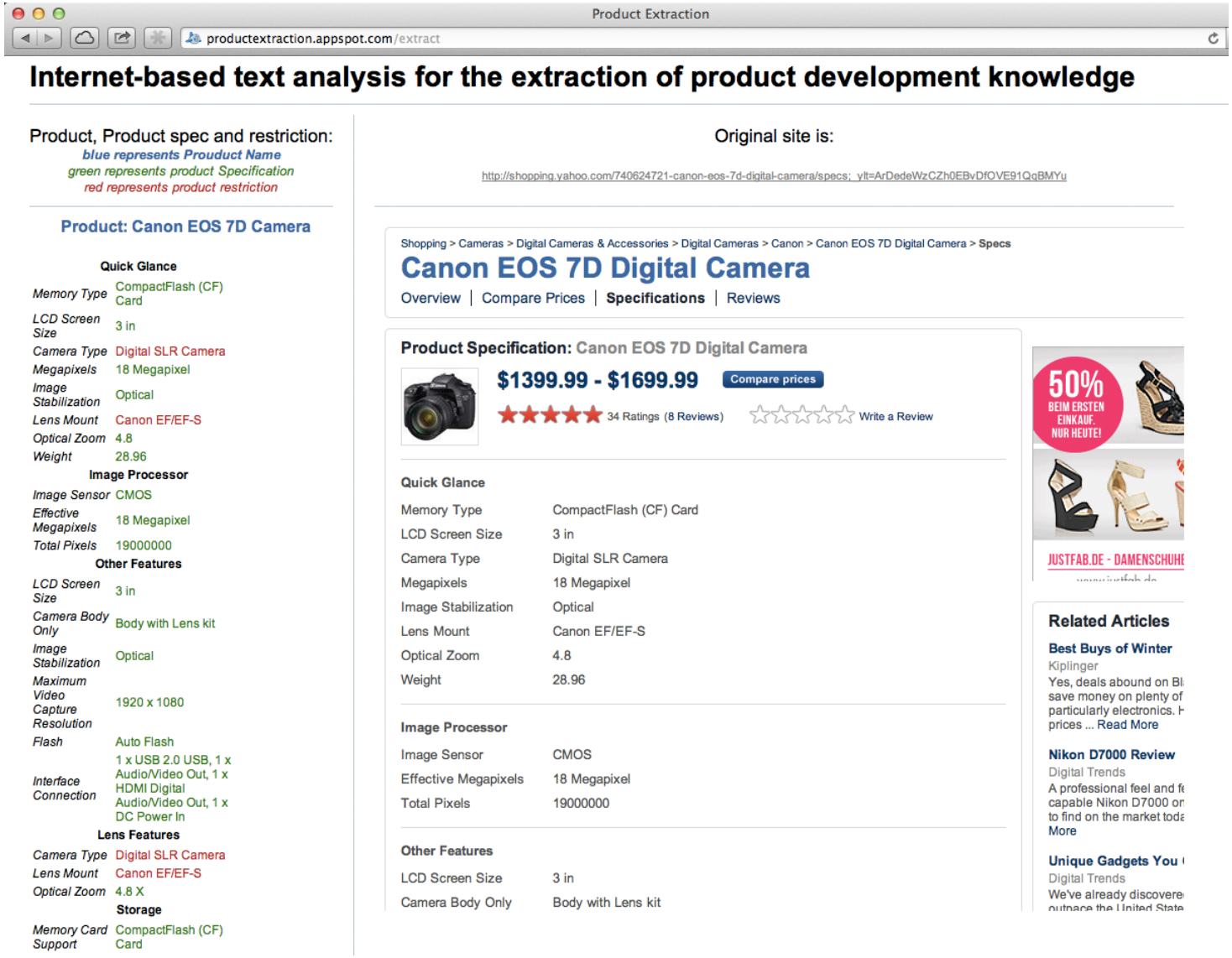

Abbildung 4.20.: Screenshot der markierten Ergebnisse

Kapitel 5.

Evaluation

Im Kapitel 4.2 nach der Extraktion haben wir 461 Beschreibungstexte im Bereich der Kameras auf der Yahoo Shopping Seite durch den Stanford POS Tagger getaggt. Aus diesen 461 Beschreibungstexten wählen wir 10% aus für das CRF-Training. Die anderen 90% verwenden wir als Testproben für den Test.

Im dem Experiment testen wir die Performance des PNER-Systems. Die Identifizierung von vier NEs wie PRO , BRA, SER und TYP werden getestet.

Wie in Kapitel 2.4 beschrieben verwenden wir die Korrektklassifikationsrate, die Genauigkeit, die Trefferquote und das F-Maß für die Evaluierung.

Tabelle 5.1 zeigt das Ergebnis der Tests in der ersten Phase bei der Identifizierung von BRA, SER, TYP und PRO ohne Präfixe „B-“ und „I-“. In diesem Fall werden zum Beispiel der B-TYP und der I-TYP zusammen als eine Einheit gezählt. Insgesamt 8376 Tokens sind verarbeitet. 1449 Phrasen von 1559 wurden gefunden. Davon sind 1236 Phrasen richtig getaggt. Die gesamte Korrektklassifikationsrate beträgt 96,20%. Die Genauigkeit ist 91,37%. Die Trefferquote beträgt 79,28% und das F-Maß ist 82,18%. Detaillierte Daten sind in der Tabelle 5.1 beschrieben.

	Genauigkeit	Trefferquote	$F_{\beta=1}$
BRA	94,04%	90,91%	92,45
PRO	82,37%	77,12%	79,66
SER	75,63%	67,42%	71,29
TYP	84,39%	76,50%	80,25
Overall	85,30%	79,28%	82,18

Tabelle 5.1.: Ergebnisse der Evaluation ohne Präfixe „B-“ und „I-“ für die Identifizierung in der ersten Phase

Nach den Ergebnissen von Tabelle 5.1 können wir feststellen, dass die Identifizierung der Produktmarke die beste Performance hat. Denn im Vergleich zu anderen Produktkomponenten sind die Produktmarken relativ statisch. Deren Aktualisieren erfolgt relativ langsam. Deshalb ist die Identifizierung relativ gesehen leichter. Die Produktserien und die Produkttypen ändern sich schnell und haben viele verschiedene

Formen. Diese Eigenschaften haben eine Menge Schwierigkeiten bei der Identifizieren gebracht. Deshalb sind die Genauigkeit, die Trefferquote und das F-Maß bei SER und TYP relativ gesehen niedriger.

	Genauigkeit	Trefferquote	$F_{\beta=1}$
B-BRA	94,04%	90,91%	92,45
B-PRO	85,46%	80,34%	82,82
B-SER	81,51%	72,93%	76,98
B-TYP	85,64%	77,22%	81,21
I-PRO	90,42%	88,22%	89,31
I-SER	0,00%	0,00%	0,00
I-TYP	66,67%	43,75%	52,83
Overall	88,48%	82,68%	85,48

Tabelle 5.2.: Ergebnisse der Evaluation mit den Präfixen „B-“ und „I-“ für die Identifizierung in der ersten Phase

Tabelle 5.2 zeigt die Ergebnisse der Evaluation mit den Präfixen „B-“ und „I-“. D.h. B-SER und I-SER werden als zwei Einheiten betrachtet und bewertet. Wir können feststellen, dass die Performance der Identifizierung von I-PRO deutlich besser als von B-PRO ist. Denn in vielen Fällen gibt es beim Ende des Produktnamens eine Produktkategorie, die einen Produktnamen anzeigt. Zum Beispiel in „Fujifilm X100 digital Camera“ ist „digital Camera“ ein Zeichen für den Produktnamen. Solche Einheiten können relativ leicht durch das PENR-System als „I-PRO“ identifiziert werden.

Im Gegensatz zum „I-PRO“ hat „I-TYP“ eine geringe Genauigkeit, Trefferquote und F-Maß. Denn normalerweise besteht der Produkttyp aus Buchstaben und Zahlen und es gibt sehr unterschiedliche Kombinationen. Deshalb ist die Grenze des Endes des Produkttyps schwierig zu erkennen. Der Anfang von Produkttyp (B-TYP) ist besser zu identifizieren, da er normalerweise direkt der Produktserie folgt. Die Noten von „I-SER“ sind null, weil es insgesamt 6 „I-SER“ Einheiten im Testkorpus gibt, aber das PNER-System keine erkannt hat.

Nach der zweiten Phase bekommen wir das Testergebnis für die gesamte Identifizierung von Produktenamen. Insgesamt gibt es 540 Produkte im Testkorpus. 498 Einheiten werden als Produkte getaggt. Davon sind 455 richtig (siehe Tabelle 5.3).

	Genauigkeit	Trefferquote	$F_{\beta=1}$
B-PRO	98,35%	89,16%	93,53
I-PRO	98,62%	88,73%	93,41
Overall	98,57%	88,81%	93,43

Tabelle 5.3.: Ergebnisse der Evaluation für die gesamte Identifizierung

Die Genauigkeit, die Trefferquote und das F-Maß sind alle höher als in der ersten Phase. Denn in vielen Fällen können die Produktnamen in der zweiten Phase richtig erkannt werden, auch wenn ihre Produktkomponenten in der ersten Phase mit falschen Labeln getaggt sind.

Die hohen Werte der totalen Genauigkeit und Trefferquote sind dadurch zu erklären, dass der Trainingskorpus und der Testkorpus aus den gleichen Webseiten und im geleichen Bereich sind. Durch den Einsatz von mehreren Trainingskorpora aus anderen Webseiten und anderen Bereichen kann ein weiteres PNER-System trainiert werden, um dieses in einem weiteren Bereich anzuwenden.

Insgesamt bestätigen die Ergebnisse der Evaluation die Wirksamkeit des PNER-Systems und geben Hinweise auf weitere Verbesserungsmöglichkeiten.

Kapitel 6.

Zusammenfassung und Ausblick

In diesem letzten Kapitel werden zunächst die Ergebnisse der Arbeit zusammengefasst. Für den Überblick über die Arbeit werden die einzelnen Abschnitte vorgestellt. Danach wird ein Ausblick gegeben.

Im Kapitel 1 geht es um die Einleitung in das Thema dieser Arbeit. Zuerst werden der Hintergrund und das Ziel der Arbeit dargestellt. Danach erfolgt eine kurze Angabe der Inhalte der nachfolgenden Kapitel. Nach den Anforderungen der Aufgabe der Arbeit wird NER als eine Lösung vorgestellt.

Die für diese Arbeit notwendigen Grundlagen werden im Kapitel 2 dargelegt. Detailliert werden zunächst die Aufgaben und Herausforderungen von NER beschrieben. Danach werden die drei Lernmethoden, drei Modelle für maschinelles Lernen von NER, sowie die Evaluationsmethode für NER vorgestellt. Schließlich wird auf PNER eingegangen.

Nach der Grundlage wird ein Konzept im Kapitel 3 für PNER im Bereich von Elektronik und Technik entwickelt. Zuerst werden die Komponenten des Produktnamens definiert. Danach wird durch die Analyse und den Vergleich der verschiedenen Methoden das CRFs Modell für das PNER-System ausgewählt. Dazu wird ein Trainingskorpus für das PNER-System gebraucht. Deshalb werden als nächstes einige Regeln zum Markieren des Trainingskorpus erstellt. Die zwei Erkennungsphasen werden auch in diesem Kapitel eingegeben.

Im Kapitel 4 wird die prototypische Anwendung „productextraction“ auf die GAE implementiert. Einige Tools für die Datenextraktion aus den Web-Seiten für den Bau des Trainingskorpus werden verglichen und analysiert. Für die Implementierung wird ein Tool „CRF++“ eingesetzt. Die Architektur, die Funktionen und die Benutzeroberfläche der Anwendung werden auch in diesem Kapitel beschrieben.

In Kapitel 5 wird auf die Bewertung der erarbeiteten Ergebnisse eingegangen. Dabei wurden die Produktkomponenten und die Produktnamen aus einem Testkorpus durch das PNER-System extrahiert und klassifiziert. Die Ergebnisse der Evaluation haben die Wirksamkeit des PNER-Systems bestätigt.

Zum Zeitpunkt dieser Arbeit unterstützt dieses PNER-System nur die Yahoo Shopping Webseite im Bereich Kameras. Durch die Sammlung von mehreren Textkorpusen aus

anderen Webseiten und Bereichen kann das PNER-System weiter trainiert und weiter angesetzt werden.

Akronyme und Abkürzungen

BRA Product Brand

CityU City University of Hong Kong

CoNLL Conference on Computational Natural Language Learning

CPL Common Public License

CRF Conditional Random Field

CRFs Conditional Random Fields

BL betreute Lernmethode

FST Finite State Transducers

GAE Google App Engine

GNU General Public License

HBL halb-betreute Lernmethode

HHMM Hierarchical Hidden Markov Model

HMM Hidden Markov Model

HTML HyperText Markup Language

IE Information Extraction

LBFGS Limited-memory Broyden-Fletcher-Goldfarb-Shanno

LDC Linguistic Data Consortium

LGPL Lesser General Public License

MALLET Machine Learning for Language Toolkit

ME Maximum Entropy Model

MEMM Maximum Entropy Markov Modelle

MIRA Margin Infused Relaxed Algorithm

MSRA Microsoft Research

MVC Model-View-Controller

MUC Message Understanding Conference

NCSA National Center for Supercomputing Applications

NE Named Entity

NER Named Entity Recognition

NEs Named Entities

NLP Natural Language Processing

NERC Named Entity Recognition and Classification

PDF Portable Document Format

PHP Hypertext Preprocessor

PNER Product Named Entity Recognition

PNE Product Named Entity

POS Part-Of-Speech

UBL unbetreute Lernmethode

URL Uniform Resource Locator

SVM Support Vector Machine

SER Product Series

STL Standard Template Library

TYP Product Type

WS-LDA Weakly Supervised Latent Dirichlet Allocation

XML Extensible Markup Language

Literatur

[AM02] E. Alfonseca und S. Manandhar.
„An Unsupervised Method for General Named Entity Recognition and Automated Concept Discovery“.
In: *Proceedings of the 1st International Conference on General WordNet*.
India, 2002 (siehe S. 17).

[AM03] Masayuki Asahara und Yuji Matsumoto. „Japanese Named Entity extraction with redundant morphological analysis“. In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*.
NAACL '03.
Edmonton, Canada: Association for Computational Linguistics, 2003, S. 8–15. DOI: 10.3115/1073445.1073447.
URL: http://dx.doi.org/10.3115/1073445.1073447 (siehe S. 15).

[Bic04] Eckhard Bick. „A Named Entity Recognizer for Danish.“ In: *LREC*.
European Language Resources Association, 2004. URL:
http://dblp.uni-trier.de/db/conf/lrec/lrec2004.html#Bick04
(siehe S. 26).

[Bik+97] Daniel M. Bikel u. a.
„Nymble: a high-performance learning name-finder“.
In: *Proceedings of the fifth conference on Applied natural language processing*.
ANLC '97.
Washington, DC: Association for Computational Linguistics, 1997, S. 194–201. DOI: 10.3115/974557.974586.
URL: http://dx.doi.org/10.3115/974557.974586 (siehe S. 15).

[Bor+98] Andrew Borthwick u. a. „NYU: Description of the MENE Named Entity System as Used in MUC-7“. In: *In Proceedings of the Seventh Message Understanding Conference (MUC-7*. 1998 (siehe S. 15).

[BP66] L. E. Baum und T. Petrie. „Statistical inference for probabilistic functions of finite state Markov chains“.
In: *Annals of Mathematical Statistics* 37 (1966), S. 1554–1563 (siehe S. 19).

[Bri99] Sergey Brin.
„Extracting Patterns and Relations from the World Wide Web“.
In: *Selected papers from the International Workshop on The World Wide Web and Databases*. WebDB '98. London, UK, UK: Springer-Verlag, 1999, S. 172–183. ISBN: 3-540-65890-4.
URL: http://dl.acm.org/citation.cfm?id=646543.696220
(siehe S. 16).

[BSW99] Daniel M. Bikel, Richard Schwartz und Ralph M. Weischedel.
„An Algorithm that Learns Whats in a Name“.
In: *Mach. Learn.* 34.1-3 (Feb. 1999), S. 211–231. ISSN: 0885-6125.
DOI: 10.1023/A:1007558221122.
URL: http://dx.doi.org/10.1023/A:1007558221122 (siehe S. 19).

[CN03] Hai Leong Chieu und Hwee Tou Ng.
„Named entity recognition with a maximum entropy approach“.
In: *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*. CONLL '03.
Edmonton, Canada: Association for Computational Linguistics, 2003, S. 160–163. DOI: 10.3115/1119176.1119199.
URL: http://dx.doi.org/10.3115/1119176.1119199 (siehe S. 15).

[Con] *Conference on Natural Language Learning, Language-Independent Named Entity Recognition (II)* (siehe S. 16).

[CR97] Nancy Chinchor und Patty Robinson.
MUC-7 Named Entity Task Definition. 1997. URL: http://www-nlpir.nist.gov/related_projects/muc/proceedings/ne_task.html
(siehe S. 14).

[CV01] Alessandro Cucchiarelli und Paola Velardi. „Unsupervised named entity recognition using syntactic and semantic contextual evidence“.
In: *Comput. Linguist.* 27.1 (März 2001), S. 123–131. ISSN: 0891-2017.
URL: http://dl.acm.org/citation.cfm?id=972778.972784
(siehe S. 17).

[CZI06] Wenliang Chen, Yujie Zhang und Hitoshi Isahara.
„Chinese Named Entity Recognition with Conditional Random Fields“.
In: *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*.
Sydney, Australia: Association for Computational Linguistics, 2006, S. 118–121. URL: http://www.aclweb.org/anthology/W/W06/W06-0116
(siehe S. 16).

[DP+92] S. Della Pietra u. a. „Adaptive language modeling using minimum discriminant estimation“.
In: *Proceedings of the workshop on Speech and Natural Language*. HLT '91.
Harriman, New York: Association for Computational Linguistics, 1992, S. 103–106. ISBN: 1-55860-272-0. DOI: 10.3115/1075527.1075551.
URL: http://dx.doi.org/10.3115/1075527.1075551 (siehe S. 21).

[GS96] Ralph Grishman und Beth Sundheim.
„Message Understanding Conference-6: a brief history“. In: *Proceedings of the 16th conference on Computational linguistics - Volume 1*. COLING '96.
Copenhagen, Denmark: Association for Computational Linguistics, 1996, S. 466–471. DOI: 10.3115/992628.992709.
URL: http://dx.doi.org/10.3115/992628.992709 (siehe S. 13).

[Guo+09] Jiafeng Guo u. a. „Named entity recognition in query“.
In: *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '09.
Boston, MA, USA: ACM, 2009, S. 267–274. ISBN: 978-1-60558-483-6.
DOI: 10.1145/1571941.1571989.
URL: http://doi.acm.org/10.1145/1571941.1571989 (siehe S. 17).

[Jay57] E. T. Jaynes. „Information Theory and Statistical Mechanics“.
In: *Physical Review* 106(4) (1957), S. 620–630 (siehe S. 21).

[JM08] Daniel Jurafsky und James H. Martin. *Speech and Language Processing (2nd Edition) (Prentice Hall Series in Artificial Intelligence)*. 2. Aufl.
Prentice Hall, 2008. ISBN: 0131873210.
URL: http://www.amazon.com/Language-Processing-Prentice-Artificial-Intelligence/dp/0131873210%3FSubscriptionId%3D13CT5CVB80YFWJEPWS02%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0131873210
(siehe S. 19).

[Kim+04] Jin-Dong Kim u. a.
„Introduction to the bio-entity recognition task at JNLPBA“.
In: *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*. JNLPBA '04.
Geneva, Switzerland: Association for Computational Linguistics, 2004, S. 70–75.
URL: http://dl.acm.org/citation.cfm?id=1567594.1567610
(siehe S. 9).

[KT07] Roman Klinger und Katrin Tomanek.
Classical Probabilistic Models and Conditional Random Fields.
Techn. Ber. TR07-2-013. Department of Computer Science, Dortmund University of Technology, Dez. 2007 (siehe S. 23).

[Kun] Maurice de Kunder. *The size of the World Wide Web* (siehe S. 9).

[LMP01] John D. Lafferty, Andrew McCallum und Fernando C. N. Pereira.
„Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data“. In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01.
San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, S. 282–289. ISBN: 1-55860-778-1.
URL: http://dl.acm.org/citation.cfm?id=645530.655813
(siehe S. 23, 29, 37).

[Lu+07] Peng Lu u. a. „Hierarchical Conditional Random Fields (HCRF) for Chinese Named Entity Tagging“. In: *2013 International Conference on Computing, Networking and Communications (ICNC)* 5 (2007), S. 24–28. DOI: http://doi.ieeecomputersociety.org/10.1109/ICNC.2007.415 (siehe S. 23).

[May+01] D. Maynard u. a. „Named Entity Recognition from Diverse Text Types“. In: *Proceedings of the Recent Advances in Natural Language Processing 2001 Conference*. Tzigov Chark, Bulgaria, 2001, S. 257–274. URL: http://gate.ac.uk/sale/ranlp2001/maynard-etal.pdf (siehe S. 15).

[ML03] Andrew McCallum und Wei Li. „Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons“. In: *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*. CONLL '03. Edmonton, Canada: Association for Computational Linguistics, 2003, S. 188–191. DOI: 10.3115/1119176.1119206. URL: http://dx.doi.org/10.3115/1119176.1119206 (siehe S. 15, 16).

[MWC05] Einat Minkov, Richard C. Wang und William W. Cohen. „Extracting personal names from email: applying named entity recognition to informal text“. In: *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. HLT '05. Vancouver, British Columbia, Canada: Association for Computational Linguistics, 2005, S. 443–450. DOI: 10.3115/1220575.1220631. URL: http://dx.doi.org/10.3115/1220575.1220631 (siehe S. 15).

[Nin+09] Ning u. a. „A Method of Chinese Named Entity Recognition Based on Maximum Entropy Model“. In: *the 2009 of IEEE ‚International Conference on Mechatronics and Automation*. chuangchun, China, 2009 (siehe S. 21).

[Niu+03] Cheng Niu u. a. „A Bootstrapping Approach to Named Entity Classification Using Successive Learners“. In: *In Proceedings of the 41st Annual Meeting of the ACL*. 2003, S. 335–342 (siehe S. 26).

[NS07] David Nadeau und Satoshi Sekine. „A survey of named entity recognition and classification“. In: *Linguisticae Investigationes* 30.1 (2007). Publisher: John Benjamins Publishing Company, S. 3–26. URL: http://www.ingentaconnect.com/content/jbp/li/2007/00000030/00000001/art00002 (siehe S. 13, 16).

[Pasimtech7] Marius Paşca. „Weakly-supervised discovery of named entities using web search queries“. In: *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. CIKM '07. Lisbon, Portugal: ACM, 2007, S. 683–690. ISBN: 978-1-59593-803-9.

DOI: 10.1145/1321440.1321536.
URL: http://doi.acm.org/10.1145/1321440.1321536 (siehe S. 17).

[Pas+06] Marius Pasca u. a. „Organizing and searching the world wide web of facts - step one: the one-million fact extraction challenge“.
In: *proceedings of the 21st national conference on Artificial intelligence - Volume 2*. AAAI'06. Boston, Massachusetts: AAAI Press, 2006, S. 1400–1405. ISBN: 978-1-57735-281-5.
URL: http://dl.acm.org/citation.cfm?id=1597348.1597411 (siehe S. 17).

[Pie02] John M. Pierre. „Mining Knowledge from Text Collections Using Automatically Generated Metadata“. In: *Proceedings of the 4th International Conference on Practical Aspects of Knowledge Management*. PAKM '02. London, UK, UK: Springer-Verlag, 2002, S. 537–548.
ISBN: 3-540-00314-2.
URL: http://dl.acm.org/citation.cfm?id=645775.668127 (siehe S. 26).

[Pin+03] David Pinto u. a. „Table extraction using conditional random fields“.
In: *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. SIGIR '03.
Toronto, Canada: ACM, 2003, S. 235–242. ISBN: 1-58113-646-3.
DOI: 10.1145/860435.860479.
URL: http://doi.acm.org/10.1145/860435.860479 (siehe S. 37).

[RJ99] Ellen Riloff und Rosie Jones. „Learning dictionaries for information extraction by multi-level bootstrapping“.
In: *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*. AAAI '99/IAAI '99.
Orlando, Florida, USA: American Association for Artificial Intelligence, 1999, S. 474–479. ISBN: 0-262-51106-1.
URL: http://dl.acm.org/citation.cfm?id=315149.315364 (siehe S. 17).

[Sek98] Satoshi Sekine.
„NYU: Description of the Japanese NE system used for MET-2“.
In: *Proc. of the Seventh Message Understanding Conference (MUC-7*. 1998 (siehe S. 15).

[Set04] Burr Settles. „Biomedical named entity recognition using conditional random fields and rich feature sets“.
In: *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*. JNLPBA '04.
Geneva, Switzerland: Association for Computational Linguistics, 2004, S. 104–107.
URL: http://dl.acm.org/citation.cfm?id=1567594.1567618 (siehe S. 16).

[SP03] Fei Sha und Fernando Pereira.
„Shallow parsing with conditional random fields“. In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. NAACL '03.
Edmonton, Canada: Association for Computational Linguistics, 2003, S. 134–141. DOI: 10.3115/1073445.1073473.
URL: http://dx.doi.org/10.3115/1073445.1073473 (siehe S. 37, 51).

[SS04] Yusuke Shinyama und Satoshi Sekine.
„Named entity discovery using comparable news articles“. In: *Proceedings of the 20th international conference on Computational Linguistics*. COLING '04.
Geneva, Switzerland: Association for Computational Linguistics, 2004.
DOI: 10.3115/1220355.1220477.
URL: http://dx.doi.org/10.3115/1220355.1220477 (siehe S. 17).

[Tou+03] Kristina Toutanova u. a. „Feature-rich part-of-speech tagging with a cyclic dependency network“. In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. NAACL '03.
Edmonton, Canada: Association for Computational Linguistics, 2003, S. 173–180. DOI: 10.3115/1073445.1073478.
URL: http://dx.doi.org/10.3115/1073445.1073478 (siehe S. 47).

[ZG10] Zhang und Guo. „Named Entity Recognition of the Products with English Based on Conditional Random Fields“.
In: *Computer Engineering and Science*.
Kunming, China: European Language Resources Association, 2010 (siehe S. 26).

[ZK06] Hai Zhao und Chunyu Kit.
„Unsupervised Segmentation Helps Supervised Learning of Character Tagging for Word Segmentation and Named Entity Recognition“.
In: *SIGHAN-5*. 2006, S. 162–165 (siehe S. 24).

[ZL10] Jun Zhao und Feifan Liu.
„Product named entity recognition in Chinese text.“
In: *Language Resources and Evaluation* 42.2 (11. Mai 2010), S. 197–217.
URL:
http://dblp.uni-trier.de/db/journals/lre/lre42.html#ZhaoL08
(siehe S. 26, 29).

Anhang A.

Ein Anhang

A.1. Verwendete Software und Bibliotheken

Software	Version
Apache	2.3.7
CRF++	0.58
Google APP Engine SDK	1.7.5
JDK	1.6.0
Jsoup	1.7.2
Php	5.4.0
Stanford POS Tagger	3.1.5
Web-Harvest	2.0

Tabelle A.1.: Verwendete Software und Bibliotheken

Alle URLs wurden zuletzt am 2013-08-21 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift